Estimation of Distribution Algorithms for Reservoir History-Matching Optimisation

Imperial College London
Department of Earth Science and Engineering

Iryna Petrovska

Thesis submitted for the fulfilment of the requirements for the degree of Doctor of Philosophy
January 2009
Reservoir modelling is widely used in the oil and gas industry to quantify the risk associated with alternative production scenarios. However, reservoir models themselves still contain a high level of uncertainty because of the typically very limited, sparse and multiscale field knowledge available. History-matching (HM) reduces this uncertainty by constraining the reservoir model to the available dynamic field data.

History-matching is an example of a typical non-linear inverse problem which yields the existence of not one but multiple solutions, which all satisfy available data constraints. In inverse problem theory Monte Carlo methods are regarded as the most accurate methods for generating a family of problem solutions and capturing posterior distributions of model parameters by exhaustive exploration of parameter space. However these methods are very rarely applicable to HM problems because they are too time and cost consuming.

While other stochastic inversion techniques have successfully overcome the runtime issue Monte Carlo methods have, none of them has provided a deliberate estimation of the posterior probabilities one would expect from Monte Carlo methods.

This thesis introduces an innovative application of a member of a class of Estimation of Distribution Algorithms - a histogram-based Population-Based Incremental Learning algorithm, to the problem of reservoir history-matching optimisation. It is shown that while avoiding an exhaustive exploration of parameter space the proposed algorithm is capable of producing the approximations of the marginal posterior distributions of model parameters which can be interpreted as their uncertainty estimates.

We also suggest a new extension of histogram-based PBIL for pair-wise conditional probabilities sampling. The developed extended version of the histogram-based PBIL is the first attempt to explicitly capture possible dependencies between reservoir model parameters and use them to perform conditional sampling of the solution space. None of the currently used algorithms and techniques for reservoir history-matching optimisation explicitly utilizes this dependency information.
# Contents

## Acknowledgements

## 1 Introduction

1.1 Inverse problem theory ........................................... 28
1.2 Reservoir history-matching as an example of a real world inverse problem ........................................... 31
1.3 Thesis aims and goals ........................................... 35
1.4 Thesis outline ........................................... 36

## 2 Reservoir History-Matching: the story so far

2.1 Classification of optimisation techniques ......................... 40
2.2 Deterministic methods ........................................... 47
2.3 Stochastic methods ........................................... 52
  2.3.1 Simulated Annealing ........................................... 53
  2.3.2 Evolutionary Algorithms ........................................... 56
  2.3.3 Neighbourhood Algorithm ........................................... 62
2.4 Geologically consistent methods ........................................... 64
  2.4.1 Randomized Maximum Likelihood ........................................... 65
  2.4.2 Ensemble Kalman Filter ........................................... 66
  2.4.3 Gradual Deformation Method ........................................... 68
  2.4.4 Probability Perturbation Method ........................................... 71
  2.4.5 Streamline-based Optimisation ........................................... 72
2.5 Experimental Design and Response Surface Methodology ............... 74
2.6 Summary ........................................... 78
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Introduction to Estimation of Distribution Algorithms</td>
<td>81</td>
</tr>
<tr>
<td>3.1</td>
<td>EDAs background</td>
<td>82</td>
</tr>
<tr>
<td>3.2</td>
<td>Discrete EDAs</td>
<td>87</td>
</tr>
<tr>
<td>3.2.1</td>
<td>No interactions</td>
<td>87</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Pairwise interactions</td>
<td>89</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Multivariate interactions</td>
<td>91</td>
</tr>
<tr>
<td>3.3</td>
<td>Continuous EDAs</td>
<td>92</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Gaussian models</td>
<td>92</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Non-Gaussian models</td>
<td>97</td>
</tr>
<tr>
<td>3.4</td>
<td>Summary</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>Population-Based Incremental Learning algorithm</td>
<td>103</td>
</tr>
<tr>
<td>4.1</td>
<td>Binary PBIL</td>
<td>103</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Discrete problem optimisation with PBILb</td>
<td>106</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Continuous problem optimisation with PBILb</td>
<td>107</td>
</tr>
<tr>
<td>4.2</td>
<td>Histogram-Based PBIL</td>
<td>110</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Discrete problem optimisation with PBILh</td>
<td>115</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Continuous problem optimisation with PBILh</td>
<td>115</td>
</tr>
<tr>
<td>4.3</td>
<td>Comparison of PBILb and PBILh</td>
<td>121</td>
</tr>
<tr>
<td>4.4</td>
<td>PBIL control parameters</td>
<td>125</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Learning rate</td>
<td>125</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Search rate</td>
<td>135</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Population size</td>
<td>140</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Size of the binary representation of discrete variables: PBILb</td>
<td>153</td>
</tr>
<tr>
<td>4.4.5</td>
<td>Histogram discretisation: PBILh</td>
<td>157</td>
</tr>
<tr>
<td>4.4.6</td>
<td>Fitness function: PBILh</td>
<td>163</td>
</tr>
<tr>
<td>4.5</td>
<td>Summary</td>
<td>172</td>
</tr>
<tr>
<td>5</td>
<td>Algorithm testing: synthetic model</td>
<td>175</td>
</tr>
<tr>
<td>5.1</td>
<td>Parallelisation of EDAs</td>
<td>176</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Overview</td>
<td>176</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Experimental parallel setup</td>
<td>178</td>
</tr>
<tr>
<td>5.2</td>
<td>Misfit function definition</td>
<td>180</td>
</tr>
<tr>
<td>5.3</td>
<td>Results</td>
<td>183</td>
</tr>
<tr>
<td>5.3.1</td>
<td>TEST 1 – Optimisation of a large parameter space based only on oil and water production match</td>
<td>186</td>
</tr>
<tr>
<td>5.3.2</td>
<td>TEST 2 – Influence of different learning rate values on the performance of PBILh</td>
<td>195</td>
</tr>
</tbody>
</table>
6 Algorithm testing: real field application
   6.1 FIELD 1 case study ......................................... 220
      6.1.1 General field description .............................. 221
      6.1.2 Test setup ........................................... 222
      6.1.3 Results. High vs. low learning rate ($\alpha$) ....... 223
      6.1.4 Results. Realistic vs. unrealistic parameter ranges . 232
   6.2 FIELD 2 case study ......................................... 241
      6.2.1 General field description .............................. 241
      6.2.2 Test setup ........................................... 242
      6.2.3 Results. Histogram-based PBIL vs. standard GA .... 245
   6.3 Summary .................................................. 263

7 PBILh with pairwise conditional sampling .......................... 265
   7.1 Previous work ............................................. 266
   7.2 Suggested algorithm modification ........................... 269
   7.3 Testing results ............................................ 277
   7.4 Summary .................................................. 292

8 Conclusions .................................................... 295
   8.1 Discussion of results ...................................... 295
   8.2 Main research contributions ................................ 301
   8.3 Future research directions .................................. 302
      8.3.1 What’s next for the application of EDAs? ............ 302
      8.3.2 What’s next for reservoir history-matching optimisation? . 303

Bibliography ......................................................... 306

Appendix A: Example of a single iteration of PBIL ................. 318

Appendix B: OneMax and PEAKS model description ............... 328

Appendix C: IC Fault model description .......................... 331

Nomenclature ....................................................... 334
List of Figures

1.1 Conventional and improved representation of forward-inverse problem relationship ........................................ 29
1.2 The least-squares misfit function of a linear and non-linear problem ........................................... 30
1.3 ................................................................................................................................................. 34

2.1 Six main and four additional history-matching specific categories of optimisation algorithms ........................................................................................................ 41
2.2 A schematic representation of various search/optimisation algorithms in terms of the degrees to which they explore the parameter space and exploit information ........................................................................................................ 43
2.3 Example of a solution space exploration by a gradient-based method of steepest descent ........................................................................................................................................ 48
2.4 Example of a solution space exploration by a tunnelling method ............................................................ 50
2.5 Pseudo-code of Simulated Annealing ....................................................................................................... 54
2.6 Example of a solution space exploration by simulated annealing ............................................................ 55
2.7 Pseudo-code of Genetic Algorithm ......................................................................................................... 58
2.8 Example of a solution space exploration by genetic algorithm ............................................................... 59
2.9 Pseudo-code of Evolutionary Strategy .................................................................................................... 60
2.10 Example of a solution space exploration by (1, 3) – ES evolutionary strategy ............................................. 61
2.11 Pseudo-code of Neighbourhood Algorithms ......................................................................................... 62
2.12 Parameter space discretisation and sampling principles of Neighbourhood Algorithm .................................... 63
2.13 Pseudo-code of the Randomized Likelihood Algorithm ........................................................................ 65
2.14 Basic framework of optimisation with Ensemble Kalman Filter ............................................................... 67
2.15 Influence of the gradual deformation parameter $\rho$ ............................................................................... 69
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.16 Gradual deformation strategy types</td>
<td>70</td>
</tr>
<tr>
<td>2.17 Identification of model flow-related zonation in Streamline-based history-matching</td>
<td>74</td>
</tr>
<tr>
<td>2.18 Pseudo-code of Response Surface methodology coupled with Experimental Design</td>
<td>75</td>
</tr>
<tr>
<td>2.19 Various types of experiment designs for response surface methodology</td>
<td>76</td>
</tr>
<tr>
<td>2.20 Time-line of techniques/algorithms used for reservoir history-matching optimisation</td>
<td>79</td>
</tr>
<tr>
<td>3.1 Main principle of EDAs operation</td>
<td>83</td>
</tr>
<tr>
<td>3.2 Pseudo-code of an Estimation of Distribution Algorithm</td>
<td>84</td>
</tr>
<tr>
<td>3.3 Classification of EDAs</td>
<td>85</td>
</tr>
<tr>
<td>3.4 Individual representation for discrete binary coded EA</td>
<td>85</td>
</tr>
<tr>
<td>3.5 Individual representation for discrete binary coded EA, history-matching specific example</td>
<td>85</td>
</tr>
<tr>
<td>3.6 Pseudo-code of Univariate Marginal Distribution Algorithm</td>
<td>89</td>
</tr>
<tr>
<td>3.7 Classification of continuous EDAs</td>
<td>93</td>
</tr>
<tr>
<td>4.1 General workflow of the binary PBIL</td>
<td>105</td>
</tr>
<tr>
<td>4.2 OneMax function optimisation with PBILb - Individual-wise (a) and generation-wise (b) evolution of the model quality with time</td>
<td>108</td>
</tr>
<tr>
<td>4.3 OneMax function optimisation with PBILb - Evolution of the prototype vector in time</td>
<td>108</td>
</tr>
<tr>
<td>4.4 PEAKS function optimisation with PBILb - Individual-wise (a) and generation-wise (b) evolution of the model quality with time</td>
<td>109</td>
</tr>
<tr>
<td>4.5 PEAKS function optimisation with PBILb - Evolution of the prototype vector in time</td>
<td>109</td>
</tr>
<tr>
<td>4.6 Histogram as a measure of frequency or fitness</td>
<td>112</td>
</tr>
<tr>
<td>4.7 General workflow of the histogram-based PBIL</td>
<td>113</td>
</tr>
<tr>
<td>4.8 Sampling of discrete and continuous variables in PBILh</td>
<td>114</td>
</tr>
<tr>
<td>4.9 OneMax function optimisation with PBILh - Individual-wise (a) and generation-wise (b) evolution of the model quality with time</td>
<td>116</td>
</tr>
<tr>
<td>4.10 OneMax function optimisation with PBILh - Evolution of the marginal probability distribution estimate (histogram) throughout the course of optimisation</td>
<td>117</td>
</tr>
<tr>
<td>4.11 OneMax function optimisation with PBILh - Estimate of the marginal probability distribution (histogram) for binary vector positions at the end of optimisation cycle</td>
<td>117</td>
</tr>
</tbody>
</table>
4.12 PEAKS function optimisation with PBILh - Individual-wise (a) and generation-wise (b) evolution of the model quality with time. 119
4.13 PEAKS function optimisation with PBILh - Evolution of the marginal probability distribution estimate (histogram) throughout the course of optimisation. 119
4.14 PEAKS function optimisation with PBILh - Estimate of the marginal probability distribution (histogram) for model parameters at the end of optimisation cycle. 120
4.15 PEAKS function optimisation with PBILh - Estimate of the marginal cumulative distribution for model parameters at the end of optimisation cycle. 120
4.16 OneMax function optimisation - Comparison of the cumulative distribution of model misfit for PBILb and PBILh. 122
4.17 PEAKS function optimisation - Comparison of the solution space sampling by PBILb and PBILh. 124
4.18 PEAKS function optimisation - Comparison of the cumulative distribution of model misfit for PBILb and PBILh. 124
4.19 Fraction of information in final posterior preserved from initial generation. 127
4.20 Fraction of information in final posterior preserved from each generation. 127
4.21 OneMax function optimisation with PBILb - Influence of the learning rate parameter $\alpha$ on the quality of sampling. 130
4.22 PEAKS function optimisation with PBILh - Influence of the learning rate parameter $\alpha$ on the quality of sampling. 130
4.23 OneMax function optimisation with PBILb - Influence of the learning rate parameter $\alpha$ on the convergence speed of the algorithm. 131
4.24 PEAKS function optimisation with PBILh - Influence of the learning rate parameter $\alpha$ on the convergence speed of the algorithm. 132
4.25 PEAKS function optimisation with PBILh - Influence of the learning rate parameter $\alpha$ on the stochastic variability in the estimate of posterior marginal probability distribution for variable $X$. 133
4.26 PEAKS function optimisation with PBILh - Influence of the learning rate parameter $\alpha$ on the stochastic variability in the estimate of posterior marginal probability distribution for variable $Y$. 134
4.27 Influence of the search rate parameter on the evolution of a single position of a probability vector. 137
4.28 PEAKS function optimisation with PBILb - Influence of the search rate parameter $s$ on the prototype vector evolution. 138
4.29 PEAKS function optimisation with PBILb - Influence of the search rate parameter $s$ on the quality of sampling .......................... 138
4.30 PEAKS function optimisation with PBILb - Influence of the search rate parameter $s$ on the convergence speed of the algorithm .......... 139
4.31 OneMax function optimisation with PBILb - Influence of the size of initial generation on the quality of sampling ........................ 142
4.32 PEAKS function optimisation with PBILb - Influence of the size of initial generation on the quality of sampling ........................ 142
4.33 OneMax function optimisation with PBILb - Influence of the size of initial generation on the convergence speed of the algorithm ...... 143
4.34 PEAKS function optimisation with PBILh - Influence of the size of initial generation on the convergence speed of the algorithm ...... 144
4.35 PEAKS function optimisation with PBILh - Influence of the size of initial generation on the stochastic variability in the estimate of posterior marginal probability distribution for variable $X$ ........................ 145
4.36 PEAKS function optimisation with PBILh - Influence of the size of initial generation on the stochastic variability in the estimate of posterior marginal probability distribution for variable $Y$ ........................ 146
4.37 OneMax function optimisation with PBILb - Influence of the size of generation $SizeGen$ on the quality of sampling ....................... 148
4.38 PEAKS function optimisation with PBILh - Influence of the size of generation $SizeGen$ on the quality of sampling ....................... 148
4.39 OneMax function optimisation with PBILb - Influence of the size of generation $SizeGen$ on the convergence speed of the algorithm .... 149
4.40 PEAKS function optimisation with PBILh - Influence of the size of generation $SizeGen$ on the convergence speed of the algorithm .... 150
4.41 PEAKS function optimisation with PBILh - Influence of the size of generation $SizeGen$ on the stochastic variability in the estimate of posterior marginal probability distribution for variable $X$ ............ 151
4.42 PEAKS function optimisation with PBILh - Influence of the size of generation $SizeGen$ on the stochastic variability in the estimate of posterior marginal probability distribution for variable $Y$ ............ 152
4.43 Maximal decimal number that can be encoded in a binary string of a given length ................................................................. 155
4.44 PEAKS function optimisation with PBILb - Influence of the size of binary representation of variables on the quality of sampling ......... 155
4.45 PEAKS function optimisation with PBILb - Influence of the size of binary representation of variables on the convergence speed of the algorithm .............................................................. 156
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.46</td>
<td>Influence of the optimal histogram discretisation on the accuracy of the underlying distribution (or solution space) approximation</td>
<td>159</td>
</tr>
<tr>
<td>4.47</td>
<td>PEAKS function optimisation with PBILh - Influence of the histogram discretisation on the quality of sampling</td>
<td>159</td>
</tr>
<tr>
<td>4.48</td>
<td>PEAKS function optimisation with PBILh - Influence of the histogram discretisation on the convergence speed of the algorithm</td>
<td>160</td>
</tr>
<tr>
<td>4.49</td>
<td>PEAKS function optimisation with PBILh - Influence of the histogram discretisation on the stochastic variability in the estimate of posterior marginal probability distribution for variable X</td>
<td>161</td>
</tr>
<tr>
<td>4.50</td>
<td>PEAKS function optimisation with PBILh - Influence of the histogram discretisation on the stochastic variability in the estimate of posterior marginal probability distribution for variable Y</td>
<td>162</td>
</tr>
<tr>
<td>4.51</td>
<td>Influence of the scaling coefficient $C_{scale}$ on the fitness estimate sensitivity</td>
<td>168</td>
</tr>
<tr>
<td>4.52</td>
<td>PEAKS function optimisation with PBILh - Influence of the fitness function scaling coefficient $C_{scale}$ on the quality of sampling</td>
<td>169</td>
</tr>
<tr>
<td>4.53</td>
<td>PEAKS function optimisation with PBILh - Influence of the fitness function scaling coefficient $C_{scale}$ on the stochastic variability in the estimate of posterior marginal probability distribution for variable X</td>
<td>170</td>
</tr>
<tr>
<td>4.54</td>
<td>PEAKS function optimisation with PBILh - Influence of the fitness function scaling coefficient $C_{scale}$ on the stochastic variability in the estimate of posterior marginal probability distribution for variable Y</td>
<td>171</td>
</tr>
<tr>
<td>4.55</td>
<td>Reservoir history-matching optimisation workflow with PBILh</td>
<td>173</td>
</tr>
<tr>
<td>5.1</td>
<td>An asynchronous Master/Slave parallel model setup</td>
<td>179</td>
</tr>
<tr>
<td>5.2</td>
<td>Water production measurement with $\tau_0 = 15.5$</td>
<td>181</td>
</tr>
<tr>
<td>5.3</td>
<td>Plot of the standard objective function against model parameter</td>
<td>182</td>
</tr>
<tr>
<td>5.4</td>
<td>Plot of the proposed objective function against model parameter</td>
<td>182</td>
</tr>
<tr>
<td>5.5</td>
<td>IC Fault Model optimisation - Comparison between the analytically derived marginal distribution estimates based on an entire IC Fault Model database for different misfit function definitions</td>
<td>184</td>
</tr>
<tr>
<td>5.6</td>
<td>IC Fault Model optimisation - Comparison between the analytically derived marginal distribution estimates based on a subset of the IC Fault Model database for different misfit function definitions</td>
<td>184</td>
</tr>
<tr>
<td>5.7</td>
<td>IC Fault Model optimisation with PBILh - TEST 1: Variability in the sizes of processed generations due to asynchronous nature of parallelisation scheme</td>
<td>187</td>
</tr>
<tr>
<td>5.8</td>
<td>IC Fault Model optimisation with PBILh - TEST 1: Individual-wise convergence of the algorithm</td>
<td>188</td>
</tr>
</tbody>
</table>
5.9 IC Fault Model optimisation with PBILh - TEST 1: Generation-wise convergence of the algorithm ................................. 188
5.10 IC Fault Model optimisation with PBILh - TEST 1: Cumulative distribution of sampled model misfit ................................. 189
5.11 IC Fault Model optimisation with PBILh - TEST 1: Simulated production of the 586 fittest models, which represent \( Q_{10} \) quantile of the total number of sampled models with misfit function threshold of \( MF < 127189 \)
5.12 IC Fault Model optimisation with PBILh - TEST 1: Frequency count of the sampled parameter values based on the best 586 history-matched models ................................. 190
5.13 IC Fault Model optimisation with PBILh - TEST 1: Evolution of the posterior marginal distribution estimates ......................... 192
5.14 IC Fault Model optimisation with PBILh - TEST 1: Final posterior marginal distribution estimates ........................................... 192
5.15 IC Fault Model optimisation with PBILh - TEST 1: Final cumulative posterior marginal distribution estimates ......................... 192
5.16 IC Fault Model optimisation with PBILh - Procedure for constructing a 2D projection of a 3D solution space for a three-variable optimisation problem ................................................. 194
5.17 IC Fault Model optimisation with PBILh - TEST 1: Image of the sampled solution space represented through its 2D projections ........ 194
5.18 IC Fault Model optimisation with PBILh - TEST 2: Influence of the learning rate parameter \( \alpha \) on the quality of sampling .................. 196
5.19 IC Fault Model optimisation with PBILh - TEST 2: Influence of the learning rate parameter \( \alpha \) on the convergence speed of the algorithm .......................... 197
5.20 IC Fault Model optimisation with PBILh - TEST 2: Influence of the learning rate parameter \( \alpha \) on the estimate of posterior marginal probability distribution for the fault throw parameter .......................... 199
5.21 IC Fault Model optimisation with PBILh - TEST 2: Influence of the learning rate parameter \( \alpha \) on the estimate of posterior marginal probability distribution for the good quality sand permeability parameter .......................... 199
5.22 IC Fault Model optimisation with PBILh - TEST 2: Influence of the learning rate parameter \( \alpha \) on the estimate of posterior marginal probability distribution for the bad quality sand permeability parameter .......................... 199
5.23 IC Fault Model optimisation with PBILh - TEST 2: Individual and generation-wise convergence of the algorithm (\( \alpha = 0.01 \)) .................. 201
5.24 IC Fault Model optimisation with PBILh - TEST 2: Individual and generation-wise convergence of the algorithm (\( \alpha = 0.1 \)) .................. 201
5.25 IC Fault Model optimisation with PBILh - TEST 2: Evolution of the posterior marginal distribution estimates ($\alpha = 0.01$) ......................... 202
5.26 IC Fault Model optimisation with PBILh - TEST 2: Evolution of the posterior marginal distribution estimates ($\alpha = 0.1$) ......................... 202
5.27 IC Fault Model optimisation with PBILh - TEST 2: Final posterior marginal distribution estimates ($\alpha = 0.01$) .............................. 205
5.28 IC Fault Model optimisation with PBILh - TEST 2: Final posterior marginal distribution estimates ($\alpha = 0.1$) .............................. 205
5.29 IC Fault Model optimisation with PBILh - TEST 2: Final cumulative posterior marginal distribution estimates ($\alpha = 0.01$) .............................. 205
5.30 IC Fault Model optimisation with PBILh - TEST 2: Final cumulative posterior marginal distribution estimates ($\alpha = 0.1$) .............................. 205
5.31 IC Fault Model optimisation with PBILh - TEST 2: Image of the sampled solution space .......................................................... 206
5.32 IC Fault Model optimisation with PBILh - TEST 2: Image of the sampled solution space .......................................................... 206
5.33 IC Fault Model optimisation with PBILh - TEST 2: 10% of the sampled models are plotted, with misfit value $MF < 4579$ ($\alpha = 0.01$) .............. 207
5.34 IC Fault Model optimisation with PBILh - TEST 2: 10% of the sampled models are plotted, with misfit value $MF < 621$ ($\alpha = 0.1$) .............. 207
5.35 IC Fault Model optimisation with PBILh - TEST 3: Influence of the size of initial generation on the quality of sampling ($\alpha = 0.01$) .............. 211
5.36 IC Fault Model optimisation with PBILh - TEST 3: Influence of the size of initial generation on the quality of sampling ($\alpha = 0.1$) .............. 211
5.37 IC Fault Model optimisation with PBILh - TEST 3: Influence of the size of initial generation on the estimate of posterior marginal probability distribution for the fault throw parameter ($\alpha = 0.01$) .............. 213
5.38 IC Fault Model optimisation with PBILh - TEST 3: Influence of the size of initial generation on the estimate of posterior marginal probability distribution for the fault throw parameter ($\alpha = 0.1$) .............. 213
5.39 IC Fault Model optimisation with PBILh - TEST 3: Influence of the size of initial generation on the estimate of posterior marginal probability distribution for the good quality sand permeability parameter ($\alpha = 0.01$) ................................................................. 214
5.40 IC Fault Model optimisation with PBILh - TEST 3: Influence of the size of initial generation on the estimate of posterior marginal probability distribution for the good quality sand permeability parameter ($\alpha = 0.1$) ................................................................. 214
5.41 IC Fault Model optimisation with PBILh - Influence of the size of initial generation on the estimate of posterior marginal probability distribution for the bad quality sand permeability parameter ($\alpha = 0.01$)  

5.42 IC Fault Model optimisation with PBILh - Influence of the size of initial generation on the estimate of posterior marginal probability distribution for the bad quality sand permeability parameter ($\alpha = 0.1$)  

6.1 Comparison of the cumulative distribution of model misfit obtained for different values of the learning rate parameter $\alpha$  
6.2 Evolution of the objective function ($\alpha = 0.2$)  
6.3 Evolution of the objective function ($\alpha = 0.5$)  
6.4 Mean objective function of a single generation ($\alpha = 0.2$)  
6.5 Mean objective function of a single generation ($\alpha = 0.5$)  
6.6 Final posterior marginal distribution estimate ($\alpha = 0.2$)  
6.7 Final posterior cumulative distribution estimate ($\alpha = 0.2$)  
6.8 Final posterior marginal distribution estimate ($\alpha = 0.5$)  
6.9 Final posterior cumulative distribution estimate ($\alpha = 0.5$)  
6.10 Matching the oil, water and gas production rates of the Well-1 ($\alpha = 0.2$)  
6.11 Matching the oil, water and gas production rates of the Well-1 ($\alpha = 0.5$)  
6.12 Matching the oil, water and gas production rates of the Well-7 ($\alpha = 0.2$)  
6.13 Matching the oil, water and gas production rates of the Well-7 ($\alpha = 0.5$)  
6.14 Comparison of the cumulative distribution of model misfit for physically realistic and unrealistic parameter ranges  
6.15 Evolution of the objective function (real. ranges)  
6.16 Evolution of the objective function (unreal. ranges)  
6.17 Mean objective function of a single generation (real. ranges)  
6.18 Mean objective function of a single generation (unreal. ranges)  
6.19 Final posterior marginal distribution estimate (real. ranges)  
6.20 Final posterior cumulative distribution estimate (real. ranges)  
6.21 Final posterior marginal distribution estimate (unreal. ranges)  
6.22 Final posterior cumulative distribution estimate (unreal. ranges)  
6.23 Matching the oil, water and gas production rates of the Well-1 (real. ranges)  
6.24 Matching the oil, water and gas production rates of the Well-1 (unreal. ranges)  
6.25 Matching the oil, water and gas production rates of the Well-7 (real. ranges)  
6.26 Matching the oil, water and gas production rates of the Well-7 (unreal. ranges)
6.27 Setup of the folded Plackett-Burman design in 48 runs 

6.28 Pareto chart of the Folded Plackett-Burman design 

6.29 Comparison of the cumulative distribution of model misfit for PBILh and GA estimated over the first half of the entire processed population 

6.30 Comparison of the cumulative distribution of model misfit for PBILh and GA estimated over the entire processed population 

6.31 Comparison of the frequency-based (GA) and fitness-based (PBILh) final posterior marginal distribution estimates 

6.32 Evolution of the posterior marginal distribution estimate for horizontal permeability multiplier \( kh\)-mult-1. Scale normalised 

6.33 Evolution of the posterior marginal cumulative distribution estimate for horizontal permeability multiplier \( kh\)-mult-1. Scale normalised 

6.34 Evolution of the posterior marginal distribution estimate for horizontal permeability multiplier \( kh\)-mult-2. Scale normalised 

6.35 Evolution of the posterior marginal cumulative distribution estimate for horizontal permeability multiplier \( kh\)-mult-2. Scale normalised 

6.36 Evolution of the posterior marginal distribution estimate for horizontal permeability multiplier \( kh\)-mult-3. Scale normalised 

6.37 Evolution of the posterior marginal cumulative distribution estimate for horizontal permeability multiplier \( kh\)-mult-3. Scale normalised 

6.38 Evolution of the posterior marginal distribution estimate for horizontal permeability multiplier \( kh\)-mult-4. Scale normalised 

6.39 Evolution of the posterior marginal cumulative distribution estimate for horizontal permeability multiplier \( kh\)-mult-4. Scale normalised 

6.40 Evolution of the posterior marginal distribution estimate for horizontal permeability multiplier \( kh\)-mult-5. Scale normalised 

6.41 Evolution of the posterior marginal cumulative distribution estimate for horizontal permeability multiplier \( kh\)-mult-5. Scale normalised 

6.42 Evolution of the posterior marginal distribution estimate for horizontal permeability multiplier \( kh\)-mult-6. Scale normalised 

6.43 Evolution of the posterior marginal cumulative distribution estimate for horizontal permeability multiplier \( kh\)-mult-6. Scale normalised 

6.44 Evolution of the posterior marginal distribution estimate for vertical transmissibility multiplier \( zt\)-1. Scale normalised 

6.45 Evolution of the posterior marginal cumulative distribution estimate for vertical transmissibility multiplier \( zt\)-1. Scale normalised 

6.46 Evolution of the posterior marginal distribution estimate for vertical transmissibility multiplier \( zt\)-2. Scale normalised
6.47 Evolution of the posterior marginal cumulative distribution estimate for vertical transmissibility multiplier $zt-2$. Scale normalised

6.48 Evolution of the posterior marginal distribution estimate for pore volume multiplier $pv-1$. Scale normalised

6.49 Evolution of the posterior marginal cumulative distribution estimate for pore volume multiplier $pv-1$. Scale normalised

6.50 Evolution of the posterior marginal distribution estimate for horizontal permeability multiplier $kh-mult-7$. Scale normalised

6.51 Evolution of the posterior marginal cumulative distribution estimate for horizontal permeability multiplier $kh-mult-7$. Scale normalised

6.52 Evolution of the posterior marginal distribution estimate for horizontal permeability multiplier $kz-mult-1$. Scale normalised

6.53 Evolution of the posterior marginal cumulative distribution estimate for horizontal permeability multiplier $kz-mult-1$. Scale normalised

6.54 SGS pressure matches of the 10 best history-matched models for selected wells

7.1 Pseudo-code of the IDEEA framework

7.2 Pseudo-code of the IDEEA framework with conditional sampling specifications

7.3 Image of the IC Fault Model solution space overlaid by the estimates of the marginal posterior distributions of model parameters

7.4 Sampling strategy of PBILh-chain for a predefined dependency chain structure

7.5 Sampling strategy of PBILh-chain for a predefined dependency chain structure

7.6 Sampling strategy of PBILh-chain for a predefined dependency chain structure

7.7 IC Fault Model optimisation with PBILh - 10% of the sampled models are plotted, with misfit value $MF < 621$

7.8 IC Fault Model optimisation with PBILh\_chain – Chain ordering $k_{good} \rightarrow h \rightarrow k_{bad}$. Evolution of the single posterior probability distribution estimate for good quality sand permeability parameter

7.9 IC Fault Model optimisation with PBILh\_chain – Chain ordering $k_{good} \rightarrow h \rightarrow k_{bad}$. Evolution of the conditional posterior probability distribution estimate for fault throw parameter

7.10 IC Fault Model optimisation with PBILh\_chain – Chain ordering $k_{good} \rightarrow h \rightarrow k_{bad}$. Evolution of the conditional posterior probability distribution estimate for poor quality sand permeability parameter
7.11 IC Fault Model optimisation with PBILh_chain – Chain ordering \( k_{\text{good}} \rightarrow h \rightarrow k_{\text{bad}} \). Final posterior conditional distribution estimates 282
7.12 IC Fault Model optimisation with PBILh_chain – Influence of the parameters ordering in a chain on the quality of sampling 284
7.13 IC Fault Model optimisation with PBILh_chain – Chain ordering \( h \rightarrow k_{\text{good}} \rightarrow k_{\text{bad}} \) 287
7.14 IC Fault Model optimisation with PBILh_chain – Chain ordering \( h \rightarrow k_{\text{bad}} \rightarrow k_{\text{good}} \) 287
7.15 IC Fault Model optimisation with PBILh_chain – Chain ordering \( k_{\text{good}} \rightarrow h \rightarrow k_{\text{bad}} \) 288
7.16 IC Fault Model optimisation with PBILh_chain – Chain ordering \( k_{\text{bad}} \rightarrow h \rightarrow k_{\text{good}} \) 288
7.17 IC Fault Model optimisation with PBILh_chain – Chain ordering \( k_{\text{bad}} \rightarrow h \rightarrow k_{\text{good}} \) 290
7.18 IC Fault Model optimisation with PBILh_chain – Chain ordering \( k_{\text{good}} \rightarrow h \rightarrow k_{\text{bad}} \) 290
7.19 Image of the solution space sampled by PBILh with \( \alpha = 0.1 \) 291
7.20 IC Fault Model optimisation with PBILh_chain – Chain ordering \( k_{\text{good}} \rightarrow h \rightarrow k_{\text{bad}} \). Image of the sampled solution space represented through its 2D projections 291

A.1 Sampling of discrete and continuous variables in PBILh 323
B.2 Matlab’s PEAKS function 330
B.3 Marginal pdfs of the Matlab’s PEAKS function estimated by numerical integration using the trapezium rule 330
C.4 IC Fault Model – reservoir model 333
C.5 IC Fault Model – Oil and water production to be history-matched 333
# List of Tables

3.1 Classification of existing EDAs according to the complexity of models they use and representation of candidate solutions .................................. 101

4.1 OneMax function optimisation - PBILb algorithm setup .................... 108
4.2 PEAKS function optimisation - PBILb algorithm setup .................... 109
4.3 OneMax function optimisation - PBILh algorithm setup .................... 116
4.4 PEAKS function optimisation - PBILh algorithm setup .................... 118
4.5 Influence of learning rate $\alpha$ on the performance of PBIL. Test setup 128
4.6 Influence of search rate $s$ on the performance of PBIL. Test setup ... 137
4.7 Influence of the size of generation $\text{SizeGen}$ on the performance of PBIL. Test setup ......................................................... 141
4.8 Influence of the size of binary representation on the performance of PBIL. Test setup ......................................................... 154
4.9 Influence of histogram discretisation on the performance of PBIL. Test setup ......................................................... 158
4.10 Influence of the scaling coefficient $C_{\text{scale}}$ on the misfit-to-fitness transformation sensitivity ......................................................... 158
4.11 Influence of fitness scaling coefficient $C_{\text{scale}}$ on the performance of PBIL. Test setup ......................................................... 169

5.1 IC Fault Model optimisation with PBILh - TEST 1: Forward model setup ......................................................... 187
5.2 IC Fault Model optimisation with PBILh - TEST 1: Algorithm setup 187
5.3 IC Fault Model optimisation with PBILh - TEST 2: Forward model setup ......................................................... 196
5.4 IC Fault Model optimisation with PBILh - TEST 2: Algorithm setup 196
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5 IC Fault Model optimisation with PBILh - TEST 3: Forward model setup</td>
<td>209</td>
</tr>
<tr>
<td>5.6 IC Fault Model optimisation with PBILh - TEST 3: Algorithm setup</td>
<td>210</td>
</tr>
<tr>
<td>6.1 PBILh algorithm setup</td>
<td>224</td>
</tr>
<tr>
<td>6.2 Forward model formulation</td>
<td>224</td>
</tr>
<tr>
<td>6.3 Fault transmissibility multipliers in previously history matched model</td>
<td>232</td>
</tr>
<tr>
<td>6.4 PBILh algorithm setup</td>
<td>234</td>
</tr>
<tr>
<td>6.5 Forward model formulation</td>
<td>234</td>
</tr>
<tr>
<td>7.1 IC Fault Model optimisation with PBILh-chain: Forward model setup</td>
<td>277</td>
</tr>
<tr>
<td>7.2 IC Fault Model optimisation with PBILh-chain: Algorithm setup</td>
<td>278</td>
</tr>
<tr>
<td>7.3 IC Fault Model optimisation with PBILh-chain: Run statistics</td>
<td>285</td>
</tr>
<tr>
<td>A.1 OneMax function optimisation with PBILh - Identification of the lowest misfit obtained by both 0 and 1 being generated in the given binary position</td>
<td>324</td>
</tr>
</tbody>
</table>
First and foremost I would like to express sincere gratitude to my research supervisor – Dr. Jonathan N Carter, for giving me this wonderful opportunity to carry out my research at Imperial College London. I appreciate the trust he had in me, the advice and encouragement he provided and the research freedom he gave me.

I am also grateful to my PhD examiners – Mike Christie from Heriot-Watt University in Edinburgh and Peter King from the Earth Science and Engineering department at Imperial College London for their interest and critical evaluation of my work. Their comments and suggestions are greatly appreciated.

I have spent a few amazing summers in the Abingdon Technology Center of Schlumberger Plc UK. I would like to say a very special thank you to Marie Ann Giddins for being the best mentor in the world and to Paul Naccache for all those very interesting and exciting things I learned from him. I would also like to acknowledge the financial support that Schlumberger AbTC provided for me in my second year of PhD, and specifically thank Rick Owen and Love Zubiller for making it possible. But most of all I would like to thank Cesar Zarpa, Vikram Sharma, Masoud Safaei Motlagh, Baris Guaguler and many others for being really great friends.

I would also like to express my gratitude to the Geoscience Research Center of Total E&P UK (Olivier Dubrule, Andre Fil and Danila Kuznetsov) and to the Chevron Energy Technology Company (Kaveh Dehghani, Hao Cheng, Wen Chen, Alex Castellini and Burak Yeten) for showing interest in my work. I spent a wonderful time working in both institutions realising that not everything that runs smoothly on a synthetic model can work as well on a real field. It was a challenging yet excellent experience not the least thanks to the wonderful people I met there.
My friends from Ukraine (Hrytsyuk Zoryana and Volodya Zamansky), Great Britain (Vera Pancaldi, Karl Charvin, Peter Deveugle, Martin Putz-Perrier, Romain Guises, Julian Mindel, Matthew Rhodes, Maximillian Kendall and many others) and USA (Ezekiel Alvarez and Tommy Tran) are credited for always being there for me through these last three years and for making it a great time to remember.

I am grateful to the Abbott family and specifically to Philip and Christine Abbott, for making me feel at home in Britain. If it were not for Philip, who knows, I might have never ended up in London in the first place.

I would also like to acknowledge the continuous support, both IT and personal, that Galina Lozhkin has given me.

A special thank you goes to Gracjan (Gracek) Lambert for his frequent (and at times objective) criticism, from which of course I as well as this thesis have benefited immensely.

But most of all I am grateful to my family for supporting me along the way, not just in this PhD but in life in general. My father Oleksandr has been a continuous source of inspiration throughout the years, and even though he is a geophysicist, I don’t hold that against him and am very proud to be his daughter. I am grateful to my mother Olenka, for if it were not for her love and care, I would have not managed to stay strong so far away from home. My little brother Dima is credited for being surprisingly much more mature than his older sister at times, and for keeping me entertained with his hilarious theories on life, love and girls. He is truly my pride and joy!

There is one person though, that I am sure would have been the happiest and the most proud – my grandmother. Her bedtime stories and her gentle smile will always hold a very special place in my memory and in my heart.

Iryna Petrovska
London
June 2008
To my grandmother
Chapter 1

Introduction

Physical theories allow us to make predictions: given a complete description of a physical system, we can predict the outcome of some measurements. This problem of predicting the result of measurements is called the *modellisation problem*, the *simulation problem*, or the *forward problem*. The inverse problem consists of using the actual result of some measurements to infer the values of the parameters that characterise the system.

Tarantola (2005)

Reservoir modelling is widely used in the oil and gas industry to quantify the risk associated with alternative production scenarios. However, reservoir models themselves still contain a high level of uncertainty because of the typically very limited, sparse and multi-scaled field knowledge available.

History-matching reduces this uncertainty by constraining the reservoir model to the available field data. History-matching is an example of a typical non-linear inverse problem which yields the existence of not one but multiple solutions, which all satisfy available data constraints.

In order to introduce the reader to the class of mathematical problems one deals with when carrying out a reservoir history-matching study, a brief background to the inverse problem theory is given in Section 1.1.

The main subject of this thesis’s research, the reservoir history-matching optimization problem, is then introduced in Section 1.2 as an example of a typical real-world highly non-linear inverse problem. We consider the position history-matching problem holds within the general framework of a reservoir simulation...
study, emphasising the importance of efficient uncertainty management at this particular stage within the framework.

The aims and goals of the presented research are given in the Section 1.3 followed by an outline of the thesis in Section 1.4.

1.1 Inverse problem theory

While a forward problem will have one unique solution, this does not necessarily apply to an inverse problem. The challenge is in the ability to identify and adequately represent the uncertainty of available prior data from which solution(s) will be derived.

The general theory has a simple probabilistic formulation and applies to any kind of inverse problem, including linear as well as strongly nonlinear problems. The probabilistic formulation of the inverse problem normally requires a resolution in terms of samples of a posterior probability distribution in the model space. This, in particular, means that the solution of an inverse problem is not a model but a collection of models that are consistent with both the data and the prior information.

To solve a forward problem means to predict the error-free values of the observable parameters $d$ that would correspond to a given model $m$. This theoretical prediction can be denoted as

$$d = f(m) \quad (1.1)$$

where $f()$ represents a group of mathematical equations describing the modelled (simulated) problem or simply some kind of a transfer function, $m$ is our subjective understanding of the modelled (simulated) problem and $d$ is its response.

Consequently, an inverse problem can be represented as

$$m = g(d) \quad (1.2)$$

There are two major sources of uncertainty that a solution of an inverse
problem can suffer from:

1. uncertainty caused by measurement-related errors;

2. uncertainty that results from a particular modelling (simulation) method being applied.

As discussed in Snieder (1998) and Tarantola (2005), in many inverse problems the model that one aims to determine is a continuous function of the space variables. However, in a realistic experiment the amount of data that can be used for the determination of the model is usually finite and in practice one often replaces the model with infinitely many degrees of freedom by a model that is characterised by a finite number of parameters.

The fact that in realistic experiments one has a finite amount of data to reconstruct a model with infinitely many degrees of freedom necessarily means that the inverse problem is not unique in the sense that there exists a family of models that explain the data equally well. The model obtained from the inversion of the data is therefore not necessarily equal to the true model that one seeks. This implies that the view of inverse problems shown in Figure 1.1 is too simplistic. For realistic problems, inversion really consists of two steps according to Snieder (1998).

Let the true model be denoted by $m$ and the data by $d$. From the data $d$ one
reconstructs an estimated model $m^*$, this is called the estimation problem, see Figure 1.1b. Apart from estimating a model $m^*$ that is consistent with the data, one also needs to investigate what is the relationship between the estimated model $m^*$ and the true model $m$. This is called the appraisal problem. In the appraisal problem one determines what properties of the true model are constrained by the estimated model.

In practice the estimation problem is usually solved by fitting the given model to available data. Given the relationship illustrated in Figure 1.1, data fitting can be achieved by minimising the difference between the real data $d$ and the estimated data $f(m^*)$ as a function of estimated model $m^*$. In the simplest case the un-weighted least-squares misfit is optimised

$$F(m^*) = \sum_i (d - f(m^*))^2$$

(1.3)

If the problem is linear, then the shape of the misfit function will be parabolically-shaped, which will ensure the existence of a single minimum (Figure 1.2 a). However when the forward problem is non-linear, the misfit function can be characterised by a presence of multiple local minima (Figure 1.2 b).
1.2 Reservoir history-matching as an example of a real world inverse problem

The problem with the local minima is that search methods for the global minimum may misidentify a local minimum as the global minimum; in that case the estimated model will not be the model of best data fit, so the identification and characterization of multiple local optima regions are major challenges in the solution of non-linear inverse problems.

The next section gives an introduction to a practical non-linear inverse problem example from the petroleum industry.

1.2 Reservoir history-matching as an example of a real world inverse problem

The purpose of petroleum reservoir history-matching (HM) is to provide support for reservoir management, which is becoming increasingly risk based.

However it is important first to highlight where exactly does the history-matching issue arise in the overall framework of reservoir simulation (RS) study. The RS process can be split into the following six steps according to Carlson (2003):

1. **Data gathering.** This stage can be quite extensive and consists of a geological and geophysical review, providing input for static reservoir model. The amount of input data and its diversity (scale-wise), as well as treatment of this data at this stage, have an outstanding influence on the accuracy of the sought after “final model”. Quality control (QC) activities, therefore, are essential at this stage. Such input data as porosity and net pay maps, as well as data coming from logs (i.e facies, permeability) have to be processed and translated/discretized onto the chosen grid.

2. **Initialisation.** A decision should be taken on the way the reservoir will be modelled, which assumptions will be made in order to simplify the simulation process while still preserving the overall behaviour of the system. The choice of fluids present in the system, choice of development scenario and the drive
present in the reservoir should also be amongst those things considered at this stage. SCAL data are then used to assign water saturation values to the grid-blocks of the static model. Initial conditions (i.e. depth of contacts, datum depth and pressure) should be determined. The final goal of the initialisation stage is to produce a STOIIP estimate, a figure which will heavily influence further RS processes, regardless whether it is green or a brown field reservoir that’s being studied.

3. History match. If the reservoir simulation study is conducted on a brown field (i.e. field that has been producing for a number of years, and its production history is available), history-matching is the next crucial step in the RS workflow. The aim of HM process is to validate the previously constructed reservoir model by means of matching simulated productions quantities (such as for example oil/gas/water production, gas-oil ratios, reservoir pressures) to the recorded quantities over the historical period of production.

As it is stressed by Carlson (2003), history-matching can take up to one third of the entire time cycle required for a full-blown reservoir simulation study, which makes it the single largest component in the RS workflow.

4. Tuning. Since the previous history-matching step mainly aims at a matching of field or group production volumes, a tuning stage is required as a way of QC (quality control) for individual well performance. Such parameters as $kh$ and skin are adjusted in order to match MDT (modular dynamic tests), RFT (repeat formation testing) or SGS (static gradient survey) data. However, if the reservoir in question has a limited number of producing wells and has not been producing for long, all these additional parameters can also be included in the history-matching stage.

5. Prediction. A study of a variety of alternative reservoir/field development scenarios and assessment of the uncertainty associated with implementation of
these scenarios takes place during the prediction stage. This assessment can either be based on a single history-matched reservoir model obtained from the previous step in RS process, or on a set of equiprobable model realisations. The aim of the prediction stage is to provide estimates of the reservoir recovery factor, cumulative production figures and ultimately the expected NPV (net-present value).

6. Reporting. Here all the assumptions made throughout the simulation process should be accordingly outlined and accounted for.

As noted above, history-matching process is the one that takes the longest period of time within the RS framework. Being situated in the middle of the reservoir simulation loop, HM process needs to be aware of both input and output uncertainties involved.

By input uncertainties we mean the inherently uncertain and multi-scale reservoir data, incomplete knowledge of subsurface geology and sparse field information used to construct a geological model. Discretisation issues arising from reservoir model gridding, as well as the actual choice of simulator (finite difference or streamline-based) will also contribute to the input uncertainties.

Output uncertainties, on the other hand, are those that HM process will itself contribute to. Some engineers avoid or ignore the impact of such uncertainties by providing a single history-matched model for future prediction studies to be carried out on. And while that somehow reduces the workload in the prediction stage, it increases the risks associated with inaccurate management, even if indirect, of the input uncertainties.

History-Matching can be viewed as a way of reducing the input uncertainties by constraining the reservoir model to available production data.

History-Matching is itself a non-linear inverse and typically ill-posed problem, which results in the existence of not one but a family of models which satisfy the observed data to some degree of accuracy (Figure 1.3). Existence of these multiple
problem solutions (alternative model realisations) is a proof of the discussed output uncertainties.

Here a typical method of uncertainty quantification is to generate a family of reservoir models, all being constrained by production history data, and use them to predict future production patterns. So the question arises: how to generate a family of solutions without exhaustive exploration of parameter space?

The best known and recommended practice here is the application of Monte Carlo techniques. However in the oil industry and particularly history-matching, a single run of a forward model, full-blown reservoir simulation, can be too time and cost consuming, which will considerably question the efficiency of these techniques.

The challenge here is therefore to produce algorithms that will automatically identify good quality history matches and estimate the uncertainty in the model parameters at that point in parameter space, whilst remembering that there are likely to be multiple independent good quality optima (history matches). These algorithms should also be robust, easy to apply (integrate with available commercial software packages) and CPU efficient.
1.3 Thesis aims and goals

Following is a list of the objectives for this research work, and subsequently the thesis:

- To investigate the application potential of the class of Estimation of Distribution Algorithms to the problem of reservoir history-matching optimisation. We aim to develop an application framework of the simplest representative of the class of Estimation of Distribution Algorithms and examine its efficiency by performing exhaustive testing on both synthetic and real-world reservoir models.

- In order to minimize CPU losses commonly associated with the implementation of stochastic optimization techniques as well as convergence problems which can be experienced during the forward modelling phase of the RS study, our goal is to develop a simple, robust and efficient parallelisation scheme for the implemented optimization algorithm. The parallelisation setup should be capable of running on a heterogeneous parallel cluster.

- The core of an efficient history-matching optimisation technique is its sampling quality. Any extra information capable of guiding the search within the solution space based on the assumptions of dependence or independence between the optimisation/model parameters should be welcomed into the optimisation framework. This is an especially valid point for the petroleum industry, and history-matching optimization, in particular. When modelling of the oil and gas reservoirs is concerned, one normally considers a wide range of reservoir model parameters, some of which may be dependent on others (such as is the case with porosity and permeability relationship for example). Although a reservoir engineer is always aware of the possibility of such interactions being present within the studied model, currently, none of the stochas-
tic optimisation techniques used in the industry explicitly take into account this extra information of possible parameter interactions.

Our aim, therefore, is to make a first step towards giving the engineer a stochastic algorithm which is able to sample conditional probability distributions of model parameters and use this information to improve the quality and efficiency of solution space exploration.

1.4 Thesis outline

The remainder of this thesis is organised as follows:

- Chapter 2 gives an insight into the work previously carried out in the area of automatic history-matching with the analysis of the most widely applied techniques and algorithms. Benefits and shortcomings of each of the methods will be discussed, and the history of their application within the industry analysed.

- Introduction to the class of Estimation of Distribution Algorithms is given in the Chapter 3. Algorithms are described and analyzed from a purely mathematical point of view, however, the justification is given for the possibilities of their application in the domain of reservoir history-matching optimisation.

- Population-Based Incremental Learning algorithm is described in Chapter 4. Both discrete and continuous versions of the algorithm are introduced and analyzed, with general algorithm performance and influence of the control parameters covered as well.

- In Chapter 5 we use the IC Fault Model to test the applicability of the researched algorithm to reservoir history-matching optimisation of a synthetic model. We also introduce a new improved definition of the objective function as well as outline the potential and benefits of the application of parallel cluster computing in conjunction with the evolutionary inspired optimisation
techniques. The parallelisation scheme of the optimisation algorithm used in current work is also presented.

- Chapter 6 presents and discusses the results of the algorithm application in two real-world reservoir history-matching optimisation studies.

- A theoretical background of the suggested extension to the original continuous version of the researched algorithm capable of sampling conditional probability distributions of model parameters and using this information to improve the quality of solution space exploration is given in Chapter 7. Preliminary results of the algorithm testing are presented.

- Finally, the thesis is concluded in Chapter 8 with general discussion of the achieved results, outline of the main research contributions and suggestions for future research directions.

- Bibliography, appendixes and nomenclature follow.
Chapter 2

Reservoir History-Matching: the story so far

When we discuss reservoir history-matching we normally talk about optimisation on highly multi-modal and multi-dimensional solution spaces.

Treatment of such type of problems requires a clear understanding of what is the ultimate aim of the optimisation process: a) to find a single optimum, hoping that it will, in deed, be a global optimum solution, or b) identify all the potential optima locations in order to get a clearer insight into the characteristics of the parameter space being optimised.

In the case of reservoir history-matching, this would constitute the question: whether one aims to history match a single model realisation and use it in further reservoir prediction and development studies; or, alternatively, produce a range of models which can all match production history within a given degree of accuracy. The latter approach will provide an opportunity for uncertainty assessment both in model parameters and prediction figures.

Depending on which of the above questions one aims to give an answer to, a wide variety of alternative optimisation methods exist.

In this chapter a general overview and classification of existing optimisation algorithms is given in Section 2.1. Algorithms are first classified with respect to the solution of an arbitrary optimisation problem; we then update this classification with classes which are commonly associated with reservoir history-matching optimisation.

A brief theory and comments on implementation practice (within the industry) of some of the most widely applied and prominent optimisation algorithms are
given in the following Sections 2.2-2.5.

The chapter is summarised by presenting and analyzing a time-line of the introduction and industrial application of the discussed optimisation techniques. Pros and cons of the covered techniques are then discussed with a view to the application of a new class of optimisation algorithms suggested in this thesis.

2.1 Classification of optimisation techniques

There does not exist a clear classification of optimisation techniques used for reservoir history-matching. Each engineer, depending on background (mathematics, physics or pure engineering) tends to classify methods differently.

Therefore, the aim is not only to provide an overview of the applied techniques, but to make a first step towards classifying them. Having a unified framework of optimisation methods’ classification will make it easier to identify and apply best-fitting techniques for the wide variety of existing optimisation problems.

As a starting point, a general classification of optimisation methods presented in Haupt and Haupt (1998) has been chosen. Here the authors identified six separate branches of optimisation techniques (Fig. 2.1, I-6). We have added four more branches that are understood to be important for the particular case of reservoir history-matching optimisation (Fig. 2.1, I-IV).

Let us consider a clockwise movement through these six branches in Fig. 2.1.

**Branch 1: Function-based and trial&error.** Function-based optimisation algorithms assume that the optimised system is described by a known mathematical function. Trial&error type techniques, on the other hand, can be viewed as a case of so called black box optimisation. Here input parameters are adjusted in order to influence output without knowing much about the system itself.

**Branch 2: Single and multiple variable.** When we talk about history-matching op-
2.1. Classification of optimisation techniques

Figure 2.1: Six categories of optimisation algorithms (1-6) according to Haupt and Haupt (1998) plus four additional reservoir history-matching specific categories (I-IV)

- Geologically consistent
- No account of geology
- Full physics methods
- Reduced physics methods
- Grid-based methods
- Non grid-based methods
- Single objective
- Multi objective
- Function
- Trial & error
- Single variable
- Multiple variable
- Static
- Dynamic
- Discrete
- Continuous
- Constrained
- Unconstrained
- Random
- Minimum seeking

Optimisation

- Branch 3: Static and dynamic. The difference between static and dynamic optimisation methods is in the way output is treated. If the output is time-dependent (i.e. the actual system setup changes in time), this leads to the production of time-varying results and then one deals with dynamic optimisation. Otherwise it is assumed static.
Dynamic optimisation is more complicated than static, which is not easy in its own right, mainly due to an introduction of an added dimension - time.

In the case of reservoir HM we deal with a static type optimisation. This is since we believe that reservoir (black box), whose behaviour we are trying to simulate and optimise, does not undergo any changes in the time-frame of the history-matching period.

Branch 4: Discrete and Continuous. *Discrete* variables, by definition, can take a limited number of values; therefore a fully-discrete optimisation yields an existence of a finite number of combinations of these variables (in our case reservoir model setups). Within the framework of reservoir history-matching well locations (grid block IJKs), facies region identifiers or preselected group of geostatistically generated properties (realisation1, realisation2,... realisation N) can be viewed as examples of discrete variables.

*Continuous* optimisation assumes to have an infinite number of continuous variable values, therefore producing an infinite number of their combinations. Most of the other variables involved in reservoir history-matching, like fault multipliers, porosity and permeability, vertical and horizontal permeability multipliers are good examples of continuous variables.

Branch 5: Constrained and Unconstrained. According to the general definition given by Haupt and Haupt (1998), unconstrained optimisation allows the optimisation variables to take any value, while constrained takes into account possible restrictions (i.e. equalities and inequalities) when calculating a misfit function.

In reservoir HM we optimise reservoir model parameters with clearly defined ranges (i.e. min and max allowed permeability, porosity, depth of GOC/OWC), in this case we talk about *unconstrained optimisation*. However, if we consider optimising multiple well locations within the reservoir of interest, selection of their I-J-K coordinates will constitute an excellent example of a *constrained optimisation*. 
2.1. Classification of optimisation techniques

Branch 6: Random and Minimum seeking. As noted by Sambridge and Mosegaard (2002) and Waziruddin et al. (2004), each of the optimisation techniques can be characterised according to their exploration and exploitation abilities (Fig. 2.2). By exploration ability of the optimisation algorithm one should understand an ability to sample the unknown solution space efficiently. Exploitation ability, on the other hand, ensures that the algorithm is able to use the available information for localisation of search and fast identification of optimal solutions efficiently. In this case random (or otherwise referred to as stochastic) optimisation techniques are those with a higher exploration ability. Exploitation ability is the main characteristic of a minimum seeking (deterministic) optimisation techniques.

In most, but not all cases, minimum seeking techniques can also be interpreted as

![Figure 2.2: A schematic representation of various search/optimisation algorithms in terms of the degrees to which they explore the parameter space and exploit information. Shaded borders indicate deterministic (nonMonte Carlo) methods. Uniform search includes the deterministic grid search. From Sambridge and Mosegaard (2002)](image-url)
local optimisation algorithms, and random methods - as global optimisation algorithms.

Branch I: Single and Multiple objective. Most of the optimisation techniques used in the industry are operated by means of evaluating the quality of the generated models by a single objective (misfit) function value. This single number is normally obtained mathematically as the sum of squared differences between the historical and the simulated production quantities, with each difference weighted individually in order to reflect different levels of errors or sensitivity expected for these quantities by engineers. While it is possible to produce and study the output of the contributions made by each of the production quantities’ mismatches, this data is not explicitly used within the optimisation routine.

Multi objective optimisation splits one single objective function into multiple objective functions. The added benefit of such representation is that it is possible to investigate the trade offs between each of them. As it is pointed out by De Jong (2006), the ultimate aim here is to produce/sample a model for which none of the multiple objectives is dominating, therefore achieving equal level of a match for all the reservoir quantities concerned.

Branch II: Grid-based and Non grid-based. The majority of reservoir flow simulations, which in essence constitute a forward problem evaluation, are carried out on a grid-based finite-difference type models. Regardless of the use of regular or irregular gridding techniques, it is believed that such methods can provide high enough accuracy to capture the physical processes taking place in the subsurface. Another characteristic of the grid-based approach is, unfortunately, its extensive computational cost.

If a single run of full blown reservoir simulation takes a number of hours or even days to finish, and for some optimisation techniques (mostly stochastic) the number of desired forward model evaluations can be in hundreds or even thousands, it becomes increasingly expensive, both money and time-wise, to utilize any of the
2.1. Classification of optimisation techniques

Three solutions to this problem exist, two of them are explicit and one implicit.

The first explicit solution suggests an application of a wide variety of upscaling techniques in order to reduce the size of the reservoir model, therefore reducing its simulation time. A drawback of such an approach is a potential decrease in the accuracy of produced results due to rough discretisation and coarsening of reservoir properties.

Application of the parallelisation schemes for reservoir simulations is the implicit solution considered here. Here a model is split into a number of sub-models (domains) and each of them is run on a separate parallel CPU. Information on the flux nature and quantities is exchanged between the domains throughout the course of simulation. This approach decreases elapsed computational time while preserving the initial precision of the model (size). However, the methodology is limited to relatively simple grids (for instance grids with non-neighbour cell connections can not be parallelised).

Second explicit solution is application of non grid-based simulation techniques. The aim being, that while the accuracy of underlying geological model is preserved, the time needed for a forward problem evaluation is significantly reduced. Amongst such techniques we highlight the use of streamline-based simulation.

Branch III: Full physics and Reduced physics methods. In case of the full physics methods, misfit of a given model is always calculated based on the results of forward modelling. Forward modelling (reservoir simulation) can be carried out on any type of available simulators.

Reduced physics methods attempt to replace the actual process of forward modelling by some kind of emulator or a proxy. A limited number of forward model runs and misfit function evaluations are performed at the starting stages of the process. Then an emulator/proxy is constructed and interpolated thereafter. It could then be verified and updated at random stages through the course of optimisation.
Branch IV: Geologically consistent and With no account of geology. Reservoir history-matching techniques, as well as engineers that implement them, can be divided into two groups: those that try to preserve geological reality and those that, “unintentionally” ignore it.

In respect to the techniques themselves, that can be explained by the inability of some algorithms to handle large numbers of optimisation variables (which is typically the case when geostatistical properties of the reservoir are being optimised). On the other hand, some of the optimisation techniques, currently applied in the industry, were explicitly designed to sample geologically consistent models.

As for the engineers themselves, choice of the algorithm and therefore treatment of geological constraints will largely depend on one’s background. It also depends on whether the HM study is conducted by a single reservoir engineer or by an integrated multidisciplinary team of both geoscientists and engineers. In the latter case the interaction between disciplines can play a crucial role in building an appropriate forward model and selecting the optimisation techniques to fully govern for all existing constraints (including geological).

Optimisation techniques, where geology is accounted for explicitly, are normally associated with an increased computational resources and set-up time, therefore no-geologically consistent techniques can be seen as much less time consuming methods. The final models produced by such methods may then be additionally perturbed to achieve a better representation of geological features.

Below, a description of the specific techniques currently used for reservoir history-matching optimisation is given. For the ease of presentation they have been divided into four groups. Deterministic (Section 2.2) and stochastic (Section 2.3) methods are presented in this thesis with an understanding of them not being able to explicitly account for geological constraints. Methods that are able to produce geologically meaningful history matched model realisations are presented in Section 2.4. And
finally, we concentrate on one of the main representatives of the class of reduced physics methods in Section 2.5.

### 2.2 Deterministic methods

Deterministic methods are based on the inverse problem theory (Tarantola (2005)), and fall predominantly within the group of function-based, unconstrained and minimum seeking algorithms, which can perform both continuous and discrete optimisation. A single objective value is used in order to evaluate the goodness of sampled models and to guide the search within the solution space. While for some application cases, discussed in due course, it is shown that these methods can be used to carry out geologically consistent optimisation, they are not explicitly designed for this purpose.

Among deterministic techniques we would like to discuss application of the following three algorithms: gradient-based methods, adjoint methods and tunnelling method.

#### Gradient-based algorithms

*Theory.* Gradient-based methods are some of the most widespread deterministic methods and have been used for many decades. They are recognised as the most efficient deterministic methods.

The objective function is optimised by means of calculating the gradients of the mathematical model with respect to its parameters.

Gradient methods belong to the group of iterative techniques, according to the description given in Box et al. (1969). These techniques require a single starting point to be defined $x_0$, they then proceed by generating a sequence of points $x_i, i = 1, 2, \ldots, n$ which represent improved approximations to the sought solution.

General idea behind the iterative methods is represented in the following rela-
2. Reservoir History-Matching: the story so far

**Figure 2.3:** Example of a solution space exploration by a gradient-based method of steepest descent

\[ x_{i+1} = x_i + h_i \cdot d_i \]  

(2.1)

\[ f(x_{i+1}) \leq f(x_i) \]

where \( d_i \) stands for an N-dimensional direction vector, with \( h_i \) is the distance measured along this vector. The direction is selected by performing a number of trial function evaluations, and once \( d_i \) is selected, \( f(x) \) is then evaluated in one or more locations along the vector. After this function evaluations (along \( d_i \)) take place, the decision is made regarding the optimal value of \( h_i \). In the case of gradient techniques, the direction \( d_i \) is selected depending on the values of the partial derivatives of the objective function \( f() \) in respect to independent variables. The derivatives being calculated are typically of the first order, however that is not always the case.

Amongst the most widely used gradient-based methods are a method of steepest descent (Fig. 2.3), Levenberg-Marquardt as well as Newton’s method and its variations.

Tarantola (2005) states that the theory and current application of gradient methods with large-scale inverse problems shows that they perform well and are naturally adapted to the cases where the relation between data and parameters is nonlinear. However the author does highlight, that despite their wide applicability there are two major drawbacks to the methods:

- Probability distributions tend to be bell shaped, and gradient methods tend to
work very inefficiently far from the maximum likelihood point;

- There is no guarantee that the maximum likelihood point is unique, or that a given point that is locally optimum is the global optimum. Only a full exploration of the space would give the proof, but this is generally too expensive to make when the number of dimensions, i.e. number of model parameters, is large (>10).

**Application.** One of the first examples of the application of gradient-based methods for reservoir history-matching optimisation was given in Anterion et al. (1989). It was argued that experience of the reservoir engineer cannot be replaced by an optimisation technique used in an automatic way, because the system cannot decide which parameters should be changed in the first place. However, authors noted that once appropriate optimisation parameters were chosen, it was possible to obtain a reasonable history match.

This work was followed by further publications by Lepine et al. (1999); Abacioglu et al. (2001); Zhang and Reynolds (2002); Brun et al. (2004) and Wang et al. (2005). Zhang and Reynolds (2002) compared a variety of gradient-based algorithms on the basis of computational efficiency and memory requirements. Among the methods applied were Gauss-Newton, Levenberg-Marquardt as well as Broyden-Fletcher-Goldfarb-Shano (BFGS) quasi-Newton method including its limited memory variation LBFGS and the Polak-Ribiére non-linear conjugate gradient algorithm (CG). Reservoir parameters being optimised included grid block porosities, horizontal and vertical permeabilities and well skin factors. Authors have indicated that even though the limited memory BFGS has demonstrated a much quicker convergence, while maintaining a high quality of sampled models, it was reported to be not robust enough.

Wang et al. (2005) among other tested methods also covered the application of steepest descent and BFGS quasi-Newton method. Reservoir parameters being op-
timised included average porosity, directional permeability and oil-water relative permeability curves. As in the previously discussed application the BFGS method was identified as the most rapid.

**Tunnelling method**

*Theory.* The major shortcoming associated with gradient-based techniques is their inability to escape a local minimum. Gomez et al. (1999) has integrated ideas of the gradient-based search with a *tunnelling method* (Fig. 2.4). The method assumes a multi-modal search space, where a number of *optima valleys* exist. The algorithm is performed in two phases.

The *minimisation phase* ensures identification of a local optimum solution within the bounded region of a parameter space. The *tunnelling phase* creates a tunnel to a nearby existing alternative valley. The starting point of the tunnel is not the exact location of the local optimum found in the previous phase, but instead a point in its neighbourhood.

![Figure 2.4: Example of a solution space exploration by a tunnelling method, from Gomez et al. (1999)](image-url)
2.2. Deterministic methods

Application. Gomez et al. (1999) tested the tunnelling approach on both synthetic and real field reservoirs. Pore volumes and transmissibility multipliers were adjusted in both cases, a total of 5 and 29 optimisation parameters for synthetic and real field cases respectively. Major conclusions from the study were that while the method was able to produce multiple history-matched models which differed very little from one another, forecasts based on the three identified model setups gave different predictions.

In some ways such a result further highlights the importance of sampling multiple solutions of the inverse problem as opposed to settling for just a single one.

Adjoint method

Theory. Adjoint methods take their origin in optimal control theory and are particularly useful for optimisation/inversion of large scale inverse problems. Cases considered are those where only a few independent experiments can be performed (due mainly to the high cost or long running time of a forward model), but the number of parameters to be optimised is very high.

Adjoint methods provide mathematical means for calculating the exact gradient information (sensitivity coefficients) of the objective function to be optimised. The adjoint system and its boundary conditions are derived from the system of partial differential equations governing the forward problem.

The efficiency and speed of convergence of adjoint optimisation techniques, as opposed to gradient-based methods, are not affected by the size of the parameter space to be optimised. These methods are able to estimate the gradients based on solely the observed data, independent of the number of optimisation variables. A single adjoint model run is used to backpropagate the mismatch (residuals) between the measured and modelled production data. The backpropagated error field is then converted into an estimate of the exact gradient of the objective function with respect to any of the optimisation parameters, regardless of the dimensionality of
the problem. It is then possible to use the estimated gradients in conjunction with any gradient-based technique to choose a new search direction within a parameter space.

*Application*. Two application examples of adjoint optimisation techniques are presented in publications by González-Rodríguez et al. (2005) and Sarma et al. (2006). González-Rodríguez et al. (2005) introduced a so called *propagation-backpropagation* adjoint method. Authors have noted that the introduction of accurate prior information at the beginning of the optimisation process can significantly improve the performance of the method regardless of the level of noise present in the available data. Application to real-time reservoir management was presented by Sarma et al. (2006).

Even at the starting stages of deterministic methods’ application, researchers and engineers realised the shortcomings and limitations of such techniques. As it was for example noted by Durrer and Slater (1977), future research directions were to be development and application of stochastic optimisation techniques.

### 2.3 Stochastic methods

Stochastic methods to some extend mimic the trial and error approach of the manual history-matching procedure, as Portella and Prais (1999) have rightly pointed out. Also referred to as *heuristic-based* optimisation techniques, these methods have enjoyed a substantial growth in their application over the last few decades. This growth was mainly triggered by the increasing availability of computing resources (parallel cluster computing) and the improved quality of the developed algorithms themselves.

According to the classification of optimisation techniques given earlier in this chapter (Section 2.1), stochastic methods fall within the category of trial&error, typi-
2.3. Stochastic methods

Stochastically multiple variable algorithms. Stochastic techniques, depending on the specific setup can handle discrete and continuous, constrained and unconstrained optimisation. They can be coupled with either grid-based or non grid-based simulation methods.

2.3.1 Simulated Annealing

Theory. Simulated Annealing (SA) presented in the works of Kirkpatrick et al. (1983); Geman and Geman (1984), is a numerical method, which uses an analogy between the process of physical annealing and the mathematical problem of obtaining the global minimum of a function that may have local minima.

It was introduced within the framework of inverse problem theory by Rothman (1985).

Simulated annealing algorithm is referred to as a single trial search approach. An energy (i.e. objective) function is to be minimised by the algorithm. A candidate move is generated from the current state, and the system must decide whether to accept that move based on a temperature $T$ and the resulting energy change $\Delta E$. In the present context of general engineering optimisation problem, a state refers to a particular configuration of the model defined by its parameters (i.e. size, type of material, physical properties etc.) and any associated constraints that restrict the values of these parameters. Objective function (energy) in this case is associated with a particular model setup and is a form of performance measure comparing historical and simulated system response. Variable values are perturbed to generate a new state of the system and to obtain an associated performance measure. A perturbation is referred to as a candidate move. The performance value is referred to as a candidate solution (Teegavarapu and Simonovic (2002)).

The temperature parameter $T$, is the most important control parameter of the algorithm. If $T$ is large (i.e. hot), many low-quality solutions are accepted, and a larger part of solution space is sampled. In this case one would expect SA to act like...
Reservoir History-Matching: the story so far

**Figure 2.5**: Pseudo-code of Simulated Annealing

```plaintext
while $T > T_{\text{min}}$
    repeat
        Perturb the system
        Compute the change in energy (misfit function) due to perturbation
        if ($\Delta E < 0$) then Accept perturbation
        else Maybe accept, with probability $\exp(-\Delta E/kT)$
    until The system is at thermal equilibrium at chosen $T$
    if $\Delta E$ was decreasing over the last few temperatures
        then $T = 0.9T$ (cool the temperature down in order to perturb more)
    else $T = T_{\text{min}}$
```

a global optimisation technique. If lower temperature values are chosen (i.e. cold) SA becomes a local optimisation method. A general workflow of the SA algorithm is presented in Fig. 2.5.

In analogy with the industrial process of annealing, the following is the common recommendation for the treatment of $T$ through the course of optimisation. It is suggested that, after a certain number of points within the parameter space have been evaluated and the misfit function was observed to decline very slowly, one lowers the temperature and thus increases the threshold for acceptable model quality. After lowering the temperature several times to a low value, one may then quench the process by accepting only the best fitting candidate solutions in order to find the local minimum of the misfit function.

The influence of the temperature $T$ on the algorithm performance is illustrated in the Fig. 2.6. Here an optimisation of some multi modal single variable function (black line) is presented. We consider the following settings for $T$: HOT, WARM, COLD and FROZEN. The black circle in the figure represents the starting point of the search.

In the first case of extremely high value of $T$ (red line), we expect the algorithm to be able to cover the entire parameter space, making sure that each of the existing optima gets a chance to be sampled. When temperature is lowered to WARM (yel-
2.3. Stochastic methods

Figure 2.6: Example of a solution space exploration by simulated annealing. Black line corresponds to the misfit function one tries to optimise with coloured lines representing different annealing schedules: red (HOT), yellow (WARM), blue (COLD) and violet (FROZEN). For example, given a starting point located in the bottom most valley, SA with a fast annealing strategy (FROZEN) will only explore a narrow neighbourhood around the starting point and will have almost no chances locating and exploring the nearby peaks and valleys. Low line, given the initial starting point, the algorithm is unlikely to sample global optima location (highest peak in the function). When $T$ is set to COLD (blue line), algorithm will just about make it to the neighbouring valley of the one where the starting point is located. And in the case of FROZEN setting (violet line), algorithm will be sampling a very narrow region around the starting point.

Application. An applicability of Simulated Annealing methods in reservoir inverse modelling has been shown in a number of publications over the years: Ouenes et al. (1993); Deutsch and Cockerham (1994); Portella and Prais (1999).

SA was applied to two real gas reservoir history-matching optimisation studies by Ouenes et al. (1993). Optimisation parameters included porosity and permeability distributions in reservoir and aquifer regions, reservoir dip angle (geometrical constraint), relative permeability curves related parameters and size of the aquifer. Authors reported good convergence performance of the algorithm and a good history match obtained for both models.

Portella and Prais (1999) applied SA to a synthetic reservoir model of 5000 cells. Such reservoir properties as end-points and exponents of the power law type rel-
ative permeability curves, grid block porosity, permeability and transmissibilities were among optimisation variables. Authors reported an "almost perfect" match with the observed data, however the produced permeability field did not correspond to the reference case. It was also highlighted, that though an acceptable history match was achieved, simulated prediction, based on the history match, differed from that produced by a reference case. Authors praised the ease with which SA can be implemented and its reliability, however they were left unimpressed by the slow rates of convergence of the method.

2.3.2 Evolutionary Algorithms

*Evolutionary Algorithms* (EAs) refer to a broad class of optimisation algorithms, which take some inspiration from evolutionary systems in the natural world. According to the general classification, EAs are trial&error, random (i.e stochastic) and global optimisation techniques, able to handle optimisation in highly multidimensional parameter spaces. EAs are equally applicable for optimisation of discrete and continuous type variables (or their mixtures). Although mostly applied in single objective optimisation, due to their population-based nature EAs are perfectly suited for multi-objective optimisation.

EAs can be coupled with grid and non-grid based reservoir simulators, as well as full-physics and reduced physics techniques. The methods are not designed to explicitly account for geological constraints in sampled models. Independent sampling of a considerably high number of optimization variables that may be strongly correlated between each other (as in the case with some geostatistical properties distributions) may not be very effective.

Usually EAs start with a population of randomly generated individuals. Then the fitness of each individual is calculated based on some performance measure, called *the misfit* or *objective function*. Next, a new population of individuals is generated through some direct manipulation of the current population.
EAs enjoy several advantages over other optimisation algorithms:

1. They do not have any special requirements of the fitness function such as continuity or differentiability;

2. Since EAs often maintain a population of individuals, while investigating many areas in parallel, they are not likely to get stuck in local optima and can easily be parallelised.

Evolutionary algorithms can be designed for optimisation of both discrete and continuous search spaces. In the analogy with the natural world evolution, these techniques operate through a hierarchy of individual representations such as:

\[
\text{Population of individuals} \rightarrow \text{Individual (chromosome)} \rightarrow \text{Pool of genes} \rightarrow \text{Alleles}
\]

which within the reservoir history-matching optimisation framework translates into the following relationship

\[
\text{Range of history – matched models} \rightarrow \text{Individual (single reservoir model)} \rightarrow \text{Parameter values} \rightarrow \text{Coded parameter values}
\]

For the case of optimisation in a continuous variable domain, the alleles definitions are absent from the structure. Perturbations are performed directly on the gene (parameter) values.

**Genetic algorithms**

**Theory.** Genetic Algorithms (GAs) are the most popular type of EAs and contrary to simulated annealing are known as multiple trial search approaches. Introduced by Holland (1975) and further extended by Goldberg (1983, 1989), these are the techniques that have received an increased interest in the area of multi discipline optimisation over the past two decades.
2. Reservoir History-Matching: the story so far

\[
t = 0
\]

Initialize and evaluate \( P(t) \)

while (Termination criteria are met)

\[
\begin{align*}
\text{Select and evaluate: } & S(t) \subset P(t) \\
\text{Generate offspring: } & O(t) \leftarrow \text{crossover, mutation}(S(t)) \\
\text{New population: } & P(t + 1) = \text{combine}(P(t), O(t)) \\
\text{t} & = t + 1
\end{align*}
\]

Figure 2.7: Pseudo-code of Genetic Algorithm

Genetic Algorithms belong to a class of optimisation methods which draw on ideas from natural evolution and genetics. The basic GA pseudocode is shown in Fig. 2.7.

Let us consider performance of a typical standard GA on a simple function optimisation problem illustrated in Fig. 2.8. Points in the figure should be understood to represent population of individuals sampled at each generation. Starting from the lower most point in the far left valley, the uphill search movement is driven by crossover operator, meaning that information from the best performing individuals in current population is used to produce a new and improved offspring. Let us assume that all the previously sampled populations for whatever reason never had a chance to sample even one individual in the area of parameter space around the global optimum location. In this case the crossover operator alone will not be able to guide the search towards that location, but instead, will contribute to the algorithm converging to the local optimum. The mutation operator is needed in order to escape from the local optimum region (as it is shown in Fig. 2.8) and explore other areas of parameter space. If no fitter individuals are found in the population sampled with mutation switched on, algorithm will simply stay within the same optimal region until mutation is applied again.

Nevertheless, GA performance is highly dependant on the choice of population size and probabilities of crossover and mutation, and it is relatively slow, being very computationally intensive compared to other methods.

Most of the theory of Genetic Algorithms deals with the so-called building blocks
2.3. Stochastic methods

Figure 2.8: Example of a solution space exploration by genetic algorithm

(BBs). By building blocks, partial solutions of a problem are meant (these are alleles equivalent coded variable values). The genetic algorithm implicitly manipulates a large number of building blocks by mechanisms of selection and recombination. It reproduces and mixes building blocks. GAs work very well only for problems where the building blocks are located tightly in strings representing the solutions. However, simple two-point recombination operators of classic GA proved to be insufficiently powerful for problems with the building blocks spread all over the solutions. In general, fixed, problem-independent recombination operators often break partial solutions what can sometimes lead to losing these and converging to a local optimum. Two crucial factors of GAs success - a proper growth and mixing of good building blocks - are often not achieved (Pelikan et al. (1999)).

Application. GAs have received a great recognition in the oil industry and results of their application can be found in a number of papers by Romero et al. (2000); Romero and Carter (2001); Schulze-Reigert et al. (2004). Velez-Langs (2005) gives an overview of a variety of GA application examples within the industry.

Castellini et al. (2006) investigate performance of GAs when optimising a group of geological realisations of the studied reservoir. Authors show that for some of the realisations GAs are able to produce acceptable history matches after only a few generation, while for others, fitness of the models never even comes close to
that of the accepted accuracy. This, obviously, again emphasizes the importance of geological information as a major constraint in the HM process.

Evolutionary Strategy

Theory. Evolutionary Strategies (ES) were also inspired by the ideas of Darwinian theory of evolution but developed and introduced independently of genetic algorithms in mid 1960s and early 1970s by Rechenberg (1973).

As opposed to evolution mechanisms in GAs, ES only evolve by means of selection and mutation. Crossover is absent. A good overview of evolution strategies is given in the paper by Back et al. (1994). Two main classes of ES can be identified: a so-called “plus” \((\mu+\lambda)-ES\) and “comma” \((\mu, \lambda)-ES\) strategies. Further to this general classification, authors differentiate two-membered ES \((1+1)-ES\), multi-membered ES \((\mu+1)-ES\) and refined ES \((\mu, \lambda)-ES\).

Two-membered ES of the \((1 + 1) – ES\) type are algorithms whose population consists of one parent individual (typically a real-valued vector), and one offspring individual. The offspring is produced by mutating the parent individual by adding a normally distributed random numbers \(\mathcal{N}(\sigma)\) to it, with \(\sigma\) being some standard deviation whose values could either be maintained constant or adopted through the course of evolution. The fittest of the two individuals is then carried on to become a parent in the new generation.

\[
t = 0
\]

\[
\text{Initialize and evaluate } P(t) \text{ of } \mu \text{ individuals}
\]

\[
\text{while} \quad \text{Termination criteria are met} \quad \text{do}
\]

\[
\begin{align*}
& \text{Generate } \lambda \text{ of offspring individuals} \quad : \quad O(t) \leftarrow \text{mutation}(P(t)) \\
& \text{if} \quad (\mu+\lambda) - ES \\
& \quad \text{then} \quad \text{New population} \quad : \quad P(t+1) \leftarrow \text{select}_{\mu \text{- fittest}}(O(t), P(t)) \\
& \quad \text{else if} \quad (\mu,\lambda) - ES \\
& \quad \quad \text{then} \quad \text{New population} \quad : \quad P(t+1) \leftarrow \text{select}_{\mu \text{- fittest}}(O(t)) \\
& \quad \quad t = t + 1
\end{align*}
\]

\[
\text{Figure 2.9: Pseudo-code of Evolutionary Strategy}
\]
Multi-membered ES of the form \((\mu + 1) - ES\), which can be generalised as \((\mu + \lambda) - ES\), were the first variant of ES to actually utilize the population principle. Here \(\mu > 1\) parents are used to produce single/multiple offspring individual(s). The top \(\mu\) best performing individuals out of the complete pool of \(\mu + \lambda\) individuals are adopted as parents for the next generation. This way parents were allowed to survive their children.

Refined ES \((\mu, \lambda) - ES\) employed a so called generational model of evolution. Only \(\mu\) fittest individuals out of the \(\lambda\) offspring can be selected to form subsequent parent population. Therefore no parents are allowed to survive.

A pseudo code of a typical ES and parameter space sampling example of the \((1, 3) - ES\) are given in Fig 2.9 and Fig 2.10 respectively.

**Application.** Although developed in early 70s, evolutionary strategies have only been adopted by petroleum industry as an efficient optimisation tool in early 00s.

The pioneering applications of ES for reservoir history-matching can be found in the publications by Schulze-Riegert et al. (2001, 2002, 2003). Encouraging results have also been reported by Al-Shamma and Teigland (2006); Griess et al.; Choudhary and Ludvigsen (2007), covering a variety of conducted real-field studies.
2.3.3 Neighbourhood Algorithm

Theory. The Neighbourhood Algorithm (NA) was first introduced by Sambridge (1999). This algorithm makes use of the geometrical constructs known as Voronoi cells to drive the search in a parameter space. These constructs are viewed as a nearest neighbour regions defined under a chosen distance norm.

Just like GA, Neighbourhood Algorithm makes use of individuals previously evaluated during the course of optimisation. However, compared to a much more complex procedure applied by GA, NA uses this prior information to approximate (i.e. interpolate) the misfit function throughout the parameter space. Here the approximate misfit surface is constructed by Voronoi diagram, which is a unique way of dividing the \( d \)-dimensional model space into \( n_p \) regions, which are called Voronoi cells (Fig. 2.12).

The general procedure of the Neighbourhood Algorithm is given in Fig. 2.11:

The trade off between exploration and exploitation in NA is controlled by number of iterations \( n_{\text{iter}} \), size of offspring \( n_s \) and size of the group of the best-performing models \( n_r \) (Erbaş (2007)). A ratio between the latter two parameters \( n_s/n_r \) is referred to as off-line refinement rate and is a measure of the exploration ability of the search.

---

\[ t = 0 \]

Initialize and evaluate \( P(t) \) of \( n_s \) individuals

while (Termination criteria are met)

\[
\begin{align*}
\text{Select } n_r \text{ best individuals} &: S(t) \subset P(t) \\
\text{and construct Voronoi cells around them} &
\end{align*}
\]

\[
\begin{align*}
\text{Generate offspring of } n_s \text{ new models} &: \text{Perform a uniform random walk} \\
\text{in the Voronoi cell of each of the } n_r \text{ chosen models and place } n_s \text{ new} \\
\text{models in each of them (i.e. } n_s/n_r) &
\end{align*}
\]

\[
\begin{align*}
\text{New population} &: P(t + 1) = \text{combine}(P(t), O(t)) \\
\end{align*}
\]

\[ t = t + 1 \]

---

Figure 2.11: Pseudo-code of Neighbourhood Algorithms
By increasing the size of parent population $n_r$, one ensures a wider exploration of the solution space, if the ratio between the two quantities is increased an exploitation ability of the algorithm is affected.

The NA computer package consists of two parts: NA-sampler and NA-Bayes. NA-Sampler represents a search method itself, while NA-Bayes is used to derive information from a set of evaluated individuals in form of Bayesian measures of resolution, covariance and marginal PDF’s. NA-Bayes is relatively independent of NA-Sampler and can be used as a post-processor for any other optimisation algorithm.

**Application.** This technique was originally applied to a problem of seismic inversion, but has recently been widely and successfully applied to the problem of history-matching optimisation and uncertainty quantification (Christie et al. (2001); Subbey et al. (2002, 2003)).

An automatic history-matching routine which amongst other techniques employs Neighbourhood Algorithm (as a solution spaces sampling tool) and aims to produce a geologically constrained reservoir realisations, is presented in the work of Suzuki and Caers (2006). Pickup and Christie (2006) performed stochastic history-

![Figure 2.12: Parameter space discretisation and sampling principles of Neighbourhood Algorithm, taken from Sambridge (1999)](image-url)
matching and uncertainty analysis study of a synthetic reservoir’s material balance problem, where noise was artificially added to the truth case production quantities. Authors then investigated impact of these data errors on the future production uncertainty estimates. NA was used as a sampling algorithm in order to differentiate the good from the bad performing models. Authors concluded that inaccurate accounting of the noise in the input data may result in misleading estimates of the posterior distributions (uncertainties).

Erbas (2007) has demonstrated application of the Neighbourhood Algorithm on a real field example. The author has highlighted the ability of the algorithm to efficiently sample the solution space (more efficiently than GA) and provide accurate uncertainty estimates for future performance predictions.

2.4 Geologically consistent methods

Four methods that are currently able to offer a geologically constrained history-matching optimisation are covered.

They are: Randomized Maximum Likelihood method (RML), Ensemble Kalman Filter (EnKF), Gradual Deformation method (GDM), Probability Perturbation Method (PPM) and Streamline-Based HM optimisation.

These methods differ in the way they choose to perturb the solution space, and in the way they direct the search and perform the alterations of the search direction itself and model properties (which are mostly geostatistical) in particular.

The later stream-line based technique can be distinguished amongst the remaining options in that geology is implicitly accounted for due to the stream-line based nature of the flow simulator used.

A brief theoretical background and history of industrial applications for each of the techniques is given below.
2.4. Geologically consistent methods

2.4.1 Randomized Maximum Likelihood

Theory. In the publications by Kitanidis (1995) and Oliver et al. (1996) Randomized Maximum Likelihood (RML) is described as a method of generating reservoir model realisations conditional to nonlinear data by using unconditional realisations of a Gaussian random field.

From Bayes’ theorem we have the following relationship

\[ p(m|d_{obs}) \propto p(d_{obs}|m)p(m) \]  \hspace{1cm} (2.2)

where \( m \) is the model or a \( N_m \)-dimensional vector of model parameters that need to be estimated, \( p(m) \) denotes the prior probability distribution for model state (parameters), \( p(d_{obs}|m) \) is the likelihood function, and \( p(m|d_{obs}) \) is the posterior probability distribution.

If one assumes that the prior model as well as data errors follow a Gaussian distribution, \( p(m|d_{obs}) \) will take the following form:

\[ p(m|d_{obs}) \propto \exp\left\{ -\frac{1}{2}(m - m_{prior})^T C_m^{-1}(m - m_{prior}) - \frac{1}{2}(g(m) - d_{obs})^T C_d^{-1}(g(m) - d_{obs}) \right\} \]  \hspace{1cm} (2.3)

In Eq. (2.3) \( m_{prior} \) is the prior mean and \( C_m \) is a \( N_m \times N_m \) covariance matrix of the prior model. \( d_{obs} \) and \( g(m) \) are \( N_d \)-dimensional vectors of the observed and predicted production data respectively. Matrix \( C_d \) is a covariance matrix of data measurement errors.

1. Generate unconditional realisation of the model parameters, \( m_{uncond} \leftarrow N[m_{prior}, C_m] \)
2. Generate a realisation of the data, \( d_{uncond} \leftarrow N[d_{obs}, C_d] \)
3. Compute the set of model parameters, \( m \), that minimizes the following function

\[ \frac{1}{2}(m - m_{uncond})^T C_m^{-1}(m - m_{uncond}) - \frac{1}{2}(g(m) - d_{uncond})^T C_d^{-1}(g(m) - d_{uncond}) \]

Figure 2.13: Pseudo-code of the Randomized Likelihood Algorithm from Liu et al. (2001)
A pseudo-code of a typical RML algorithms is given in Fig. 2.13.

**Application.** Liu and Oliver (2003) presented a comparative study of a range of uncertainty assessment algorithms applied to a 1D synthetic single-phase problem. Amongst the considered techniques, apart from RML, were Rejection Algorithm (REJ), Markov chain Monte Carlo (MCMC), linearisation about the MAP (LMAP) and the Pilot Point (PP) method. Conclusion of the conducted study was that RML in general performed better than other techniques.

Application of the RML technique to a PUNQ-3S study can be found in Gao et al. (2006). Authors show that the method is able to accurately sample the posteriori distributions for the studied problem where the relation between the data and the model is understood to be non-linear.

### 2.4.2 Ensemble Kalman Filter

**Theory.** The technique was introduced by Evensen (1994) as an extension to the earlier developed *Kalman Filter* by Kalman (1960).

EnKF is a recursive filter suitable for non-linear problems with large numbers of variables. It is used for data assimilation for ensemble forecasting. The objective of the method within the reservoir modelling framework is to use dynamic production data and the prior geologic models to provide the posterior geomodel parameters and forecast uncertainties. Wen and Chen (2005) summarize the main features of the method applied to reservoir optimisation as follows:

- EnKF incrementally updates reservoir models by assimilating the available production data in a sequential fashion. New production series are included into the process once they are available (at the next recorded time step). This is the largest difference EnKF has over almost all other optimisation techniques used for reservoir HM optimisation. Typically, production data for the entire period of field development are matched, which means that for each sampled
Figure 2.14: Basic framework of optimisation with Ensemble Kalman Filter. From http://cg.ensmp.fr/elefebvre/EnKF.ppt

point in the solution space a full cycle of reservoir simulation is carried out, which can make HM process very time consuming (Fig. 2.14);

- An ensemble of models is maintained and can be extracted at any time, if there is a need to perform an intermediate prediction study. This ensemble will be constrained/matched to the most recent production information (current time step);

- The ensemble-based nature of the method makes it also an excellent candidate
for grid computing (parallelisation), which makes it very CPU efficient;

- EnKF is simulator independent;

- Method does not require sensitivity coefficients calculation, as opposed to gradient-based optimisation techniques for example.

**Application.** The first application to reservoir modelling was reported by Nævdal et al. (2002). Over the years EnKF has been applied to a wide variety of models and optimisation cases, results of such applications can be found in Nævdal et al. (2003); Gu and Oliver (2004); Hauhen et al. (2006); Bianco et al. (2007) and Evensen et al. (2007). In the latter publication, authors note that the method does not appear to suffer from the curse of multiple local minima, which have been observed in many other methods. This must be attributed to the sequential processing of observations, but also the fact that the EnKF allows for model errors in addition to errors in the estimated parameters. Furthermore, the solution is searched for in the space spanned by the ensemble members rather than the high dimensional parameter space.

Nævdal et al. (2003) particularly highlight the importance of accurate sampling of the initial ensemble of models, to ensure that it accurately reflects the uncertainty ranges present in the system. In the work by Wen and Chen (2005) some of the questions regarding the choice of ensemble size and assimilation time interval size are raised.

### 2.4.3 Gradual Deformation Method

**Theory.** *Gradual deformation method* (GDM) is the technique specifically developed by and for petroleum industry.

The basic idea of the gradual deformation method, introduced by Roggero and Hu (1998), is to generate a process of realisations which evolve smoothly at each step. This is achieved by the introduction of a correlation between the new real-
**2.4. Geologically consistent methods**

![Image showing two realisations of a Gaussian random field with different values of \( \rho \).](image)

**Figure 2.15: Influence of the gradual deformation parameter \( \rho \)**

Let \( Z_1 \) and \( Z_2 \) be independent realisations of a Gaussian random field, then a new realisation \( Z \) can be obtained in the following way:

\[
Z = \cos(\rho \pi) Z_1 + \sin(\rho \pi) Z_2
\]  

(2.4)

where \( Z_1 \) and \( Z_2 \) should be normalised and unconstrained to the well data. Parameter \( \rho \) here is called a *gradual deformation parameter* and its value varies in the range \( \rho \in [0, 2] \).

A change in \( \rho \) leads to a continuous variation in the distribution \( Z \) at each grid node (Fig. 2.15).

A newly created random field \( Z \) can then be transformed into a log-normally distributed permeability field. This formulation ensures that geostatistical parameters of the new realisation \( Z \) (such as mean and standard deviation) will be preserved no matter which value of \( \rho \) is being used.

Based on the principle of linear combination of two independent realisations, there are two different types of gradual deformation strategies:
A so called *constantly evolving* structure (Fig. 2.16, a). Here one starts by randomly generating an initial realisation of a Gaussian random field $Z_1$, a new realisation $Z_2$ is then created at random and the optimal value of gradual deformation parameter, which minimizes the misfit between the actual data and result obtained from the resulting realisation $Z_3$, is chosen. At a next step a new randomly drawn realisation $Z_4$ is created and this process continues until a required precision has been reached.

A so called *limited* structure (Fig. 2.16, b), where one starts by generating $2^N$ realisations of a Gaussian random field, which are then paired up in a hierarchical fashion and each group is being optimised separately (i.e. $Z_1$ and $Z_2$, $Z_{11}$ and $Z_{12}$). Here a number of gradual deformation parameters is predetermined and equals to $(2^N - 1)$.

If we analyze each of the strategies, first one provides the ability to explore as many realisations as possible to reach a desired accuracy of the mismatch. On the other hand, the second strategy, despite being relatively limited to the parameter space area represented by initial set of realisations, can be easily implemented in parallel.

A good analysis of the method is given in Liu and Oliver (2004), where a comparison study is carried out. The study concluded that distributions of reservoir properties produced by the GDM method were comparable to those produced by Markov Chain Monte Carlo method.

Further extensions of the algorithm include GDM of conditional realisations (Hu
2.4. Geologically consistent methods

(2002)) and GDM with pilot points (Le Ravalec-Dupin and Hu (2007)).

Application. Introduced by Roggero and Hu (1998), GDM has successfully been applied for almost a decade. Publications by Le Ravalec-Dupin and Nœtinger (2002), where the authors comment on the method outperforming SA algorithm; by Hu and Jenni (2005), where GDM application to history-matching on an object-based reservoir model is presented; by Roggero et al. (2007), where a case of a real field history-matching optimisation is presented, are just a few of the application examples.

2.4.4 Probability Perturbation Method

Theory. The Probability Perturbation Method (PPM) was introduced by Caers (2004), and its detailed description can be found in the later publication by Caers and Hoffman (2006).

The method does not rely on the traditional Bayesian decomposition of posterior into likelihood and prior, instead it uses so-called “pre-posterior distributions”, i.e. the probability of the model parameters given some subset of the data. The method relies on fast non-iterative sequential simulation to generate model realisations, instead of an iterative sampling. The data is matched by perturbing an initial realisation using a perturbation mechanism termed probability perturbation.

The underlying idea of the PPM is that there exists a probability model which was used to generate a geostatistical realisation of some kind. This realisation could have come from some multi-point sequential simulation process for example. PPM perturbs the probability model not the actual property, the idea which is somewhat similar to that of GDM. The probability conditioned on the production data is calculated as follows:

\[ P(A|D) = (1 - r_D)^i(0)(u) + r_D P(A) \]  \hspace{1cm} (2.5)

where \( P(A|D) \) is a probability of some event \( A \) (property value) occurring given the
production data $D$, $P(A)$ is a marginal distribution of $A$, $\mathbf{u} = (x, y, z) \in \text{reservoir}$ is a spatial location within the reservoir, and $i^{(0)}$ is an initial realisation. $r_D$ is called a perturbation parameter.

Given the initial realisation of the model $i^{(0)}$, value of $r_D \in [0, 1]$ is then optimised in order to produce an optimal level of perturbation of the original model. When $r_D = 0$, production data do not contribute any information, and the initial realisation is maintained unchanged.

An optimal value of the perturbation parameter, once found, will produce a new realisation $i^{(1)}$ which will result in a better history match than the one obtained from $i^{(0)}$.

**Application.** First practical application of PPM was reported by Hoffman and Caers (2004). A North Sea reservoir was history matched by optimisation of the location and size of the low-permeability calcic bodies. The authors reported an ability of the method to produce a geologically consistent improved history match.

### 2.4.5 Streamline-based Optimisation

**Theory.** In case of a streamline-based methods, it is possible to calculate the trajectory of the streamlines in the subsurface along which movement of the fluids occurs. The saturation front is being tracked by means of the pressure and gravity field distribution. This method not only is much faster than the grid-based simulation techniques but is also able to identify specific zones within the reservoir, which may require special attention in order to obtain a geologically consistent history match.

Good overview of the theory of streamline-based modelling methods as well as specifically their application to the reservoir history-matching problem can be found in Datta-Gupta and King (2007). Authors highlight that apart from the advantage of fast forward model calculations, and flow-pattern identification, an added benefit of the method is its ability to quickly calculate the streamline-derived sensi-
Emanuel and Milliken (1998) introduced a term Assisted History-Matching (AHM). In the process of AHM, streamline identified interactions between the producer-injector groups were used to aid the history-matching process. The workflow of the method was as follows:

1. Forward model evaluation, by means of reservoir simulation (regardless of the type of simulator used);
2. Tracing of streamlines and computation of the time of flight;
3. Identification of the flow-based regions associated with each producing well (Fig. 2.17), given the streamline trajectories identified in the previous step;
4. Misfit function calculation individually for each well;
5. Manual alteration of the pre-identified zone properties in order to achieve a better history match for each of the producers.

Application. As well as providing the theoretical background of the AHM methodology, Emanuel and Milliken (1998) also give examples of two real field applications of the technique. Authors compared history matches obtained with both traditional and streamline-based assisted history-matching methods. It is reported that AHM was able to produce a much higher quality matches than those obtained by the traditional HM technique, and has done so only at a fraction of computational time (few iterations) traditional methods typically require.

Kulkarni and Datta-Gupta (2000) used streamline-based approach to estimate the relative permeability values from the available production data. The authors propose an analytical approach to calculation of the sensitivities in the production
response to variations in reservoir properties, particularly relative permeability values. Power functions and B-splines were used to approximate the shape of relative permeability curves.

2.5 Experimental Design and Response Surface Methodology

Theory. We consider Experimental Design (ED) or, as it also known, Design of Experiments (DoE) and Response Surface Methodology (RSM) as an excellent example of reduced physics methods (Section 2.1).

Response Surface Methodology (Box and Wilson (1951)) operates through the recognition of a relationship between the system’s input and response variables, assuming that this relationship can be described by a mathematical equation.

This relationship, once identified by means of some type of Experimental Design, is then used to construct a so called response surface, an emulator to replace the
2.5. Experimental Design and Response Surface Methodology

Define an initial design $D^N(0)$

Evaluate $D^N(0)$ and identify $M$ heavy hitters ($M \leq N$)

while (termination criteria are met)

1. Define design $D^M(t)$ and perform experiments
2. Generate response surface $RS(t)$
3. Optimize (here any of the optimization techniques can be used for parameter space exploration)
4. Define new refined design $D^M(t + 1)$

$t = t + 1$

Figure 2.18: Pseudo-code of Response Surface methodology coupled with Experimental Design

The actual simulated system. This response surface is also known as a surrogate model or a proxy. Therefore, after a sufficient amount of forward simulations, performed according to the selected design, the proxy model is fitted to the produced output data (misfit functions) and further forward model evaluations can be carried out purely by interpolating the proxy. The proxy can then be verified at various time intervals, by the performance of check runs.

As it was noted by Eide et al. (1994), HM is done by minimising the distance between the observed values of the response variables, and the response predicted by the proxy. The process is iterative and is presented in Fig. 2.18.

If one assumes a linear correlation between the inputs $(I_1, I_2, \ldots)$ and output $O$ of the system, a polynomial can be fitted to the solution space to represent/reflect the system’s performance.

\[ O = \alpha_0 + \alpha_1 I_1 + \alpha_2 I_2 + \alpha_3 I_1 I_2 + \alpha_4 I_1^2 + \alpha_5 I_2^2 \]  

(2.6)

In order to derive the coefficients of the polynomial, one should perform a range of experiments $E^j = (I_1^j, I_2^j, \ldots)$, $j \in 1 \ldots N$ predefined by a selected Experimental Design (ED), with $N$ being the dimension/resolution of the chosen design. Linear
2. Reservoir History-Matching: the story so far

Regression then enables us to calculate the coefficients of the polynomial.

In Eq. (2.6), $\alpha_0$ is an intercept term, $\alpha_1$ and $\alpha_2$ are known as main effects, which show the sensitivity of the proxy to the changes in $I_1$ and $I_2$ factors. $\alpha_3$ defines an effect of a two-factor interaction, and $\alpha_4$ and $\alpha_5$ are known as quadratic terms.

The general notation for a 2-level design is to use $+1$ and $-1$ in place of the high level (i.e. maximum value) and the low level (i.e. minimum value) respectively, for each factor (optimisation variable). $0$ is used as a notation for centre or middle point of the design.

In the case of reservoir history-matching, the output of the system $O$ is the misfit function value $MF$ which quantifies the difference between the historical and simulated production quantities, and the inputs $I$ are reservoir properties (parameters) which require alteration.

The reader is advised to refer to *e-Handbook of Statistical Methods* (2003) for a very good and basic description of RSM methodology and most widely applied experimental designs, some of which we chose to cover in this overview:

![Figure 2.19: Various types of experiment designs for response surface methodology. 2-level designs including full factorial design $2^3$ (a), fractional factorial design $2^{3-1}$ (b) and Box-Behnken design with middle point (c). 3-level full factorial design $3^3$ (d) ](image-url)
2.5. Experimental Design and Response Surface Methodology

- **Full factorial design**, where every setting of every factor appears with every setting of every other factor (Fig. 2.19, a). By factors one should understand optimisation variables. If there are \( k \) optimisation variables, with each allowed to take two values (minimum and maximum), a full factorial design will consist of \( 2^k \) runs. Such designs are only applicable to cases with less than 5 optimisation variables;

- **Fractional factorial design** is applied when one deals with high numbers of optimisation variables. In this case if one deals with a 2-level design, only \( 2^{k-1} \) experiments will be carried out (Fig. 2.19, b). Broadly speaking, with designs of resolution three, and sometimes four, we seek to screen out the few important main effects from the many less important others. For this reason, these designs are often termed *main effects designs, or screening designs*;

- **Plackett-Burman design** (PB), a very economical design with the number of experiments being a multiple of four (rather than a power of 2). PB designs are very efficient screening designs when only main effects are of interest, since these main effects are, in general, heavily confined within two-factor interactions. The PB design in 12 runs, for example, may be used for an experiment containing up to 11 factors. A folded PB (FPB) design is also amongst the most widely used screening designs. It is obtained from an original design with its appended copy in which signs in all the columns have been reversed.

- **Box-Behnken design** creates experiments which are formed by the variable values that represent the middle of the sampling intervals plus a mid (center) point (Fig. 2.19, c). Center points are typically added to check for the curvature in the response surface;

- **Higher level designs**, where variables are not only allowed to take minimum and maximum values, but also values in the middle of the defined interval.
2. Reservoir History-Matching: the story so far

Therefore a total number of performed experiments will be $N^k$. A case of 3-level 3 variable design is illustrated in Fig. 2.19, $d$.

Low-resolution designs are typically used for screening the main effects and higher-resolution designs are applied when interaction effects need to be estimated and response surfaces need to be constructed.

**Application.** Some of the earliest publications on the subject of RSM application in history-matching were by Egeland et al. (1992); Damsleth et al. (1992) and Aanonsen et al. (1995).

Results of a comparative study of different design schemes can be found in Yeten et al. (2005). Reis (2006) applied RSM for the optimisation of both synthetic and real field reservoirs.

RSM can be coupled with a wide variety of optimisation techniques. Examples include deterministic techniques such as simplex method, presented in the work by Hoffman and Caers (2000) and stochastic techniques such as genetic algorithms, presented in the works by Castellini et al. (2006); Yu et al. (2007, 2008).

### 2.6 Summary

In the current chapter, firstly an attempt was made to provide a unified framework for classification of optimisation algorithms applied to the problem of reservoir history-matching optimisation. A brief description of the theory and application practice of the most widely applied techniques was given.

Secondly, we have highlighted the importance of the proper use of terminology when referring to history-matching optimisation. Three main types should be correctly understood and differentiated within the HM optimisation domain: traditional (i.e. manual), assisted and automatic history-matching.

*Traditional or manual history-matching* approaches normally involve a reservoir engineer manually adjusting parameters of the reservoir model in order to obtain a better match between simulated and historical data.
2.6. Summary

Assisted history-matching, as introduced by Emanuel and Milliken (1998), is the use of algorithmic techniques to assist the process of traditional history-matching. Which means that there still remains a part in the history-matching loop that although guided by the optimisation algorithm still requires some amount of pre- or post-processing by the user.

And automatic history-matching is understood to represent a true black box type optimisation case. User interaction within the history-matching process is limited to input data preparation and postprocessing of the results once the optimisation has finished.

Apart from the streamline-based HM optimisation methods, we typically deal with either manual or automatic optimisation. With an increased availability of the computing resources, the later type of HM optimisation is currently receiving much more attention from the industry (Fig. 2.20). However some engineers still practice minor manual fine tuning to the results of automatic history-matching studies.

Amongst the automatic HM optimisation methods we have highlighted those that are and are not able to explicitly account for geological constraints while sam-
pling the solution space. And while a number of techniques such as EnKF, GDM and PPM were developed in order to allow for the explicit account of the geology, in this thesis we concentrate on the application potential of algorithms which can only implicitly handle these constraints.

Among such algorithms, stochastic techniques such as SA, GA and NA have been covered. Since Simulated Annealing is understood to be a single trial approach, we question its ability to efficiently sample the highly multi-modal parameter space in multiple potential optima locations simultaneously. Therefore GA and NA are considered to be superior to the latter approach.

Both algorithms have had a good track record of successful applications within the industry, and drawing from their main principles of population utilisation (implicit search parallelism) in this thesis we propose application of the technique, which is somewhat similar to the two stochastic methods.

Taking its origin from and being an alternative to genetic algorithms, it is a group of methods that learn the structure of the problem as they work their way through it and use this information to ensure a proper mixing and growth of building blocks (which was reported to be an issue with standard GAs). The approaches are based on probabilistic modelling of promising solutions to guide the exploration of the search space instead of using crossover and mutation like in the case of standard GA.

This idea lies at the basis of operation of the class of Estimation of Distribution Algorithms, and the following chapter will concentrate on the description of the theory of the methods and potential of their application for the problem of reservoir history-matching optimisation.
In the previous chapter the most widely applied algorithms for reservoir history-matching optimisation were described. Stochastic optimisation techniques, and specifically those of the class of Evolutionary Algorithms (Genetic Algorithms, Evolutionary Strategies) and Neighbourhood Algorithm, have been highlighted as some of the most rapidly developing with more and more production companies adopting them as a strong component of the reservoir history-matching process.

Genetic algorithms and evolutionary strategies operate by evolving a population of candidate solutions, in our case reservoir model realisations. This evolution is carried out through the application of such operators as selection, mutation and crossover (absent in ES). Both algorithms identify fit individuals and try to reproduce best features of these individuals by combining or modifying them in some way. The common output from a single run of GA or ES is a collection of sampled models. The engineer is then free to select a desirable threshold of the misfit function and filter out a group of models which could either be used to study the frequency-based statistics of the sampled model parameter ranges or as a basis for future prediction uncertainty studies.

The Neighbourhood Algorithm also operates by evolving a population of candidate solutions; however it uses their fitness information in order to estimate the likelihood values which are then used for the discretisation of the solution space. Voronoi cells with higher likelihood estimates are refined, therefore clearly indicating to the user the location of the potentially optimal regions within the parameter
space. While being able to sample a collection of history-matched models, NA can also provide estimates of the posterior marginal distributions of model parameters, which are inferred from the sampled Voronoi space using the NA-Bayes framework.

One thing that unites all three above mentioned optimisation techniques is that they are not able (were not designed) to account explicitly for possible interactions/correlations between model parameters and use this information to guide the exploration of the solution space.

This chapter will introduce a class of Estimation of Distribution Algorithms, which in essence extends the general principles of evolution in a standard genetic algorithm. EDAs replace the population of individuals with a probability model representative of that population; the model is then re-sampled in order to guide the search within the solution space. These probability models can then be interpreted as the estimates of uncertainty in model parameters, which, in a particular case of reservoir history-matching, given a limited and sparse nature of the available a priori data, can prove extremely useful.

In section 3.1 a brief introduction will be given to the class of discussed algorithms. Their general features and main properties will be presented. EDAs designed for optimisation of discrete parameter spaces will be discussed in greater detail in Section 3.2, followed by their continuous analogues in Section 3.3.

### 3.1 EDAs background

From an abstract point of view, the selected set of promising solutions can be viewed as a sample drawn from an unknown probability distribution. As pointed out by Pelikan et al. (2002), knowing that distribution would allow the optimisation algorithm to generate new solutions that are somehow similar to the ones contained in the original selected set of solutions.

*Estimation of Distribution Algorithms* (EDAs) or *Probabilistic Model-Building Genetic Algorithms* (PMBGAs), as they are also referred to in the literature, are algorithms
3.1. EDAs background

that estimate the unknown true probability distribution by using a selected set of solutions. This estimate is used to generate new solutions.

EDAs start, as simple GAs, by selecting better solutions from the previously randomly generated population of solutions. However, mutation and recombination operators of a typical GA are then replaced by the following two steps in EDA:

1. **Model building.** A process during which a probabilistic model of the promising solutions is constructed (Fig. 3.1, b and d);

2. **Model Sampling.** A process during which new candidate solutions are generated by sampling the constructed model (Fig. 3.1, a and c).

This transforms the standard GA framework presented in Chapter 2, Section 2.3.2 (Fig. 2.7) into that shown in Fig. 3.2. By *elitism*, in Fig. 3.1 and Fig. 3.2, one should understand a mechanism employed in some EAs which ensures that the chromosomes of the fittest individuals of the population are passed on to the next generation without being altered by evolution operators. Elitism can sometimes very rapidly increase performance of EA, because it prevents losing the best found solution to date.
$t = 0$

Initialize and evaluate $P(t)$

while (Termination criteria are met)

Select and evaluate: $S(t) \subset P(t)$

Generate offspring: $O(t)$ in the following two steps:

1. Construct a probabilistic model of promising solutions
2. Sample the constructed model to generate new candidate solutions

New population: $P(t + 1) = \begin{cases} 
\text{if EDAs with elitism} \\
\text{combine}(P(t), O(t)) \\
\text{if EDAs without elitism} \\
\text{then } O(t) 
\end{cases}$

$t = t + 1$

Figure 3.2: Pseudo-code of an Estimation of Distribution Algorithm

The major advantage of such model-based methods as EDAs is that a distribution estimate can capture the structure of a problem very accurately and ensure a very effective mixing and reproduction of individuals. However, estimation of the true distribution is not a trivial task. There is a trade off between the accuracy and the efficiency (convergence time) of the estimate.

There are different classes of EDAs currently being applied to the solution of optimisation problems. The algorithms are classified according to the complexity of models they use and the way in which candidate solutions are represented (Fig. 3.3).

The terms of the discrete and continuous optimisation were introduced in Chapter 2, Section 2.1 (Fig. 2.1, Branch 4). Discrete EDAs, just like a standard genetic algorithm, traditionally use a binary coded string to represent a candidate solution. In accordance with the evolution principles of the natural world, optimisation parameters are coded in the way presented in Fig. 3.4, with the HM optimisation specific case shown in Fig. 3.5. The probability model estimation and resampling are performed at the level of alleles, or coded optimisation parameter values in the case of reservoir HM optimisation.

If a $(M - 1)$-bit long binary string (string of alleles) is used to code a decimal
3.1. EDAs background

**Figure 3.3**: Classification of EDAs

- **No interactions**
- **Pairwise interactions**
- **Multivariate interactions**

**Figure 3.4**: Individual representation for discrete binary-based coded EA

**Figure 3.5**: Individual representation for discrete binary-based coded EA, history-matching specific example. Here $N$ parameters are being optimised, with each being coded into a $(M+1)$-bit binary string.
parameter value $V^{\text{dec}}$ (gene value), which is defined within the following range – \((V_{\text{min}}, V_{\text{max}}]\), following are the required transformations to convert the binary vector into a decimal number:

\[
V^{\text{bin}} = [b_M \ b_{M-1} \ldots \ b_0]
\]

\[
V^{\text{dec}\text{unscaled}} = b_M \times 2^M + b_{M-1} \times 2^{M-1} + \ldots + b_0 \times 2^0
\]

The maximum decimal number that can be encoded in a binary string is

\[
V^{\text{decmax}} = 1 \times 2^M + 1 \times 2^{M-1} + \ldots + 1 \times 2^0
\]

The value of $V^{\text{decmax}}$ essentially describes the level of discretisation for a specific parameter (i.e. a maximum number of values it may take). The scaled decimal value of the parameter, defined within the given range \((V_{\text{min}}, V_{\text{max}}]\) is then calculated in the following way:

\[
V^{\text{decscaled}} = \frac{V^{\text{decunscaled}} \times V^{\text{decmax}}}{V_{\text{max}} - V_{\text{min}}}
\]

Here, number of sampled models will be limited to the number of possible optimisation parameters’ combinations which in turn is dependent on the length of the chosen binary representation, as stated above.

Continuous optimisation assumes that a continuous variable can take any of an infinite number of values, therefore producing an infinite number of resulting models. Here EDAs, as do all continuous EAs, stop at the gene level and perform all required probability model evaluations and sampling directly on the continuous optimisation parameter values.

The following two sections will present the main representatives of both discrete and continuous Estimation of Distribution Algorithms, while categorising them according to the types of probability model they apply.
3.2 Discrete EDAs

As stated above, if candidate solutions (individuals) are represented by fixed-length strings over a finite alphabet (such as in case of a binary coding, for example) one deals with discrete EDAs.

In all the algorithms discussed it is assumed that model building and sampling is taking place at the binary solution string level. By the solution string one should understand a string comprised of the appended binary vector representations of the coded optimisation parameters. The solution string will therefore have a total of $D \times (M - 1)$ binary positions in the case when a problem with $D$ parameters is being optimised.

3.2.1 No interactions

EDAs of this class operate according to the assumption that all positions within the solution string are independent and evolve separately during the course of optimisation (Fig. 3.3, a).

Some of the representatives of this class of discrete EDAs include Population-Based Incremental Learning Algorithm (PBIL) and specifically its binary version introduced by Baluja (1994), Compact Genetic Algorithm (cGA) introduced by Lobo and Harik (1999) and Univariate Marginal Distribution Algorithm (UMDA) introduced by Mühlenbein and Paass (1996). Since interactions between optimisation parameters are not taken into account, a total probability distribution here can be represented as

$$p(x) = \prod_{i=1}^{D \times (M - 1)} p(x_i)$$

where $p(x_i)$ is a marginal distribution of the $i$th solution string position.

In PBIL a population of candidate solutions at some generation $t$ is replaced by a so-called probability or prototype vector $P(t) = (p_1, p_2, \ldots, p_{D \times (M - 1)}), \ i \in 1 \ldots D \times (M - 1)$, where $p_i$ denotes the probability of a 1 being generated in the $i$th position of a
solution string. Each position of the probability vector evolves independently of the remaining positions and thus no interactions are considered. The main difference between PBIL and a typical GA is that instead of using a population, PBIL replaces it by a probability vector/model. The probability vector is then updated according to the general updating rule

\[ P(t + 1) = (1 - \alpha)P(t) + \alpha X_{\text{best}} \]

where \( \alpha \in [0; 1] \) is referred to as learning rate, and is the most important control parameter of the algorithm, and \( X_{\text{best}} \) is the solution string which represents the fittest individual sampled in generation \( t \).

Just like in PBIL, cGA replaces the population by a probability vector. Initially all bits in the probability vector are initialised to 0.5, to represent a uniform distribution. cGA progresses by sampling the population of \( N \) candidate solutions and randomly selecting two of them to perform the probability model updating step. These solutions are then evaluated and identified as best and worst of the pair respectively.

Let us denote the bit in \( i \)th position of the best and worst of the two solution strings by \( x_{i}^{\text{best}} \) and \( x_{i}^{\text{worst}} \). The probability vector entries will be updated in the following way:

\[
\begin{align*}
p_i &= \begin{cases} 
p_i + \frac{1}{N} & \text{if } x_{i}^{\text{best}} = 1 \text{ and } x_{i}^{\text{worst}} = 0 \\
p_i - \frac{1}{N} & \text{if } x_{i}^{\text{best}} = 0 \text{ and } x_{i}^{\text{worst}} = 1 \\
p_i & \text{otherwise} \end{cases}
\end{align*}
\]

By applying this updating rule cGA emphasizes not only the importance of information carried by the individual (binary string) as a whole, but also information that can be individually extracted from each position in the binary string. The size of updating step in cGA, which is equivalent to the learning rate parameter \( \alpha \) in PBIL, is controlled by the size of population \( N \). The larger \( N \) is the slower will the probability vector evolve and the less drastic will be moves within the solution
3.2. Discrete EDAs

$t = 0$
Generate $N >> 0$ binary strings at random

while Termination criteria reached
  Select $M < N$ according to the chosen selection method
  $t = t + 1$
  Compute the marginal frequencies of the binary positions $p(x_i, t)$
  given the selected set of $M$ strings
  Generate new $N$ candidate solutions according to the marginal distributions
  $p(x, t) = \prod_{i=1}^{n} p(x_i, t)$

Figure 3.6: Pseudo-code of Univariate Marginal Distribution Algorithm (UMDA)

UMDA is slightly different from both PBIL and cGA, in that it maintains a population of solutions. At each iteration a set of promising solutions is chosen and probability vector is computed according to them. This probability vector is constructed as an estimate of marginal frequencies for each of the solution string positions. New solutions are generated by sampling the vector and are then used to replace the old ones. This process is repeated until a required stopping criterion is met. The pseudo-code of the algorithm is given in Fig. 3.6. A paper by Paul and Iba (2003) gives a good overview of the performance of both GA and UMDA for the optimisation of the OneMax problem.

PBIL, cGA and UMDA perform well for problems, which can easily be decomposed into subproblems of order at most one. Pelikan (2005), however, reported this group of algorithms to scale up poorly with problem size.

3.2.2 Pairwise interactions

Discrete EDAs of this type are also called bivariate algorithms (Fig. 3.3, b and c), as they can encode a pairwise-type dependencies between optimisation parameters (binary solution string positions) by representing them in the shape of chain, tree or
forest-like networks.

The Mutual-Information-Maximising Input Clustering algorithm (MIMIC) introduced in the work of de Bonet et al. (1997) utilizes a chain-type model (Fig. 3.3, b) to encode conditional probability distributions, such as

\[ p(x) = p(x_1)p(x_2|x_1)p(x_3|x_2) \ldots p(x_n|x_{n-1}) \]

where \( i \in 1 \ldots n \) denotes the ordering of variables in a dependency chain. The first variable in the chain is sampled independently and all the variables that follow are sampled using the conditional probabilities encoded by the dependency structures.

The Combining Optimizers with Mutual Information Trees (COMIT) algorithm was introduced by Baluja and Davies (1997). It takes into account pair-wise interactions between optimisation variables by building a tree-like dependency structures. As in PBIL, the population is replaced by a probability vector, but in the case of COMIT it contains all pairwise probabilities. A dependency tree-type structure operates by encoding a probability distribution in the way that every variable except for the root-variable is dependent on its parent in the tree (Fig. 3.3, c). The model of distribution used by COMIT is as follows:

\[ p(x) = \prod_{i=1}^{n} p(x_i|x_i^*) \]

where \( x_i^* \) represents a parent of \( x_i \), which makes it possible for a number of optimisation parameters (or solution string positions) to share the same parent.

The bivariate marginal distribution algorithm (BMDA) Pelikan and Mühlenbein (1999) is an extension of UMDA. It applies a forest-like dependency structure that can also be interpreted as a set of mutually independent dependency trees or chains. In this representation it is not necessary that every variable should have the common relative, so the algorithm can work with more than one root variable.
3.2.3 Multivariate interactions

This class of EDAs includes algorithms that take into account interactions between the variable of order more than two. Pelikan (2005) reports this class of EDAs to outperform their uni- and bivariate predecessors on a range of larger scale real-world problems which are characterised by a much more complex dependency structures. Extended Compact Genetic algorithm (EcGA) by Lobo and Harik (1999), Factorised Distribution algorithm (FDA) by Mühlenbein and Mahnig (1999) and Bayesian Optimisation algorithm (BOA) by Pelikan et al. (1999) are a few representatives of this class.

EcGA is an extension of cGA and operates by partitioning the entire set of variables into a smaller subsets, using a Marginal Product Model (MPM), each of them is then treated as a group of independent variables, and is processed by simple UMDA (Fig. 3.3, e). MPM used in this case can be defined as

\[ p(x) = \prod_{c \in m} p(x_c) \]

where \( m \) is a set of smaller subsets \( c \) and \( p(x_c) \) is the marginal probability of a set of variables \( x_c \) in the subset \( c \).

FDA applies a fixed factorised distribution throughout the entire optimisation period. While the model takes into account both multivariate marginal and conditional probabilities, it does not learn the structure and nature of dependencies. To be applied efficiently this algorithm requires our problem to be previously decomposed into sub problems. This may work well if there is a sufficient amount of prior information available, however this clearly does not fulfil the requirements posed by a typical black-box problem.

BOA (Fig. 3.3, d), first constructs a Bayesian network for a current population of promising solutions. By Bayesian network (BN) one should understand a directed acyclic graph with each node corresponding to the parameters being optimised (or positions of the solution string) and the edges which indicate conditional depen-
dencies between them. BN is typically accompanied by a set of conditional probability tables which specify a conditional probability for each node value given any instance of nodes this node depends on.

This network is then used to sample new candidate solutions. A network is created by the application of three main operations: edge addition, edge reversal and edge removal. By edge one understands a certain relationship between two parameters (solution string positions). BOA is very similar to FDA, however as well as learning probabilistic information about the problem, BOA is also capable of identifying its structure. And although BOA does not require any specific knowledge about the problem being solved, it can easily be incorporated using standard Bayesian statistics.

3.3 Continuous EDAs

In this section we will introduce and discuss only some (most relevant to our work) of the current extensions of EDAs into continuous optimisation domain.

Two groups of algorithms can be distinguished:

1. those that employ a single or multi-Gaussians as the probability model;

2. those that employ non-Gaussian type models.

3.3.1 Gaussian models

Continuous EDAs that employ Gaussian-type models to describe the true underlying distributions operate through either uni-modal also known as single-peak (Fig. 3.7, a) or multi-modal distributions, which essentially are a mixture of normal distributions (Fig. 3.7, b).

In the basic case of the uni-modal distribution the nature and character of the search performed by the algorithm depends on two parameters - mean value $\mu_i$ of the $i$th variable and a corresponding standard deviation $\sigma_i$. Here the value of $\mu_i$
largely determines the starting position of the search, while $\sigma_i$ is responsible for the level of diversity in population. Choosing large values of the standard deviation will result in an almost random sampling. The smaller the values of $\sigma_i$, the narrower would be the sampled neighbourhood region around the mean value. All the above however, makes uni-modal type algorithms extremely biased to the starting position, which of course can lead to premature convergence to a closest local optimum.

An algorithm introduced by Rudlof and Köppen (1996) and called the Stochastic Hill Climbing with Learning by Vectors of Normal Distributions (SHCLVND) is one of the first representatives of the single-peak Gaussian-based continuous EDAs. The algorithm was partially inspired by both PBIL and a Stochastic Hill Climbing with Learning (HCwL) algorithm by Kvasnica et al. (1995). Similar to HCwL and PBIL, SCHCLVND does not store information in the shape of population. For HCwL this information is stored as a vector of probabilities, for SCHCLVND - as a vector of
probability distributions.

SHCLVND operates through a vector of means \( \mu = (\mu_1, \ldots, \mu_n) \), where \( n \) is a total number of optimisation variables. This vector is then used with a single value of a standard deviation \( \sigma \), constant for all variables. Since one still considers independence between the optimisation variables, the overall distribution will be given by a product of individual Gaussian distributions for each variable.

Similar to the prototype vector in PBIL, SHCLVND uses a vector of means \( \mu \) and standard deviation \( \sigma \) to generate the first generation of individuals at random. In this case the vector of means would be centred in the middle of the variable ranges, and the choice of the standard deviation would be left to the user.

After all individuals have been evaluated, the vector of means is updated in a similar way to the prototype vector is in PBIL, towards the best solution in the given generation.

\[
\begin{align*}
\mu(t + 1) &= \mu(t) + \alpha (\mu_{\text{mid}}^{\text{best}} - \mu) \\
\mu_{\text{mid}}^{\text{best}} &= \frac{1}{N_{\text{best}}} \sum_{i=1}^{N_{\text{best}}} \mu_i^{\text{best}}
\end{align*}
\]

Relationship (3.1) represents a first step in the probability model updating process - mean vector update. Here \( N_{\text{best}} \) is the size of the set of best (fittest) vectors, and \( \mu_{\text{mid}}^{\text{best}} \) is the average of all vectors in this subset. The second step of the probability model updating process is the handling of the standard deviation \( \sigma \). It is modified according to the relationship (3.3), which yields a steady narrowing of the searched neighbourhood from generation to generation.:

\[
\sigma(t + 1) = \sigma(t) \cdot \sigma_{\text{reduce}}, \quad \sigma_{\text{reduce}} \in (0; 1]
\]

If left equal to 1, a standard deviation modification constant \( \sigma_{\text{reduce}} \) will result in an extensive Monte Carlo type sampling, which is something this technique was designed to avoid. With possible optima location identified by the \( \mu \) vector, smaller values of \( \sigma_{\text{reduce}} \) will provide a very concentrated sampling around the small neigh-
bourhood of $\mu$.

Another approach was proposed by Sebag and Ducoulombier (1998), and while being very similar to SHCLVND and operating through the evolution of a normal distribution $\mathcal{N}(\mu, \sigma)$, a much greater emphasis was put on the standard deviation $\sigma$ adjustment. The authors called this algorithm continuous PBIL (PBILc).

Sebag and Ducoulombier (1998) suggested updating the vector of means $\mu$ based on the information taken from two best and one worst individuals in the selected generation (3.4).

$$\mu(t+1) = (1-\alpha)\mu(t) + \alpha \cdot (\mu_{1}^{\text{best}} + \mu_{2}^{\text{best}} - \mu^{\text{wor}})$$ (3.4)

Since the diversity of the sampled population will largely depend on the value of $\sigma$, the authors consider it to be the main control parameter for the approach and suggest a number of alternative solutions to the problem. The first suggestion is to use a constant value of $\sigma$, similar to the case of SHCLVND when $\sigma_{\text{reduce}} = 1$. In this case the search will never be able to become too concentrated in particular areas of the solution space (identified by the vector of means $\mu$). The second option is to include $\sigma$ into the chromosome (individual) itself. In this way as well as perturbing the values of means, PBILc will also perturb or self-adapt the $\sigma$ values before they are used to generate new population of individuals.

Through the course of optimisation diversity of the sampled populations is expected to decrease, which is why the authors suggested to adjust $\sigma$ according to the diversity in the fittest sampled offspring. A set number of fittest individuals $N_{\text{best}}$ is selected from the current generation and the standard deviation is updated as follows:

$$\sigma_{i} = \sqrt{\frac{\sum_{i=1}^{N_{\text{best}}} (\mu_{i} - \mu_{\text{mid}}^{\text{best}})^{2}}{N_{\text{best}}}}$$ (3.5)

where $\mu_{\text{mid}}^{\text{best}}$ (as is in the case of SHCLVND) is the average of all vectors of means in the selected subset of fittest sampled offspring.

Another way of handling the standard deviation value is to treat it in the same
way as mean values $\mu$ are treated, and learn its value based on the diversity of $N_{\text{best}}$ offspring.

$$\sigma_i(t + 1) = (1 - \alpha)\sigma_i(t) + \alpha \cdot \sqrt{\frac{\sum_{i=1}^{N_{\text{best}}} (\mu_i - \mu_{\text{mid}})^2}{N_{\text{best}}}}$$

(3.6)

Sebag and Ducoulombier (1998) compare PBILc to the $(1, \lambda)$-ES type evolutionary strategy, where the single parent is PBILc’s $(\mu(t), \sigma(t))$. However major differences should be noted:

- In the case of $(1, \lambda)$-ES, a parent is replaced by the fittest individual in the offspring generation (in our case mean $\mu(t)$), while for PBILc $\mu(t)$ is updated instead;

- While ES takes quick decisions based on the instant information, due to the nature of means vector updating as well as various options for standard deviation treatment PBILc is able to maintain a much longer memory and move slowly towards the potential optima locations.

The above techniques (SHCLVND and PBILc) however suffer from one common shortcoming. If one deals with a highly multi-modal fitness landscape, a single dome Gaussian distribution will fail to approximate the true underlying distribution accurately. It will either end up close to uniform, with $\sigma$ not being able to decrease fast enough, or alternatively will concentrate on only one of all the possible optima locations within the parameter space.

In order to overcome this issue a number of techniques based on mixtures of normal distributions and normal kernel distributions were developed (Fig. 3.7, b).

Gallagher et al. (1999) considered using the Adaptive Mixture Model (AMix) of normal distributions for each of the optimisation variables. Here the final distribution consists of the number of single-peak normal component distributions which are combined in the following way:

$$p(x) = \sum_{i=1}^{M} C_{\text{mix}}^i p(x|i), \quad 0 \leq C_{\text{mix}}^i \leq 1$$

(3.7)
3.3. Continuous EDAs

\[ \sum_{i=1}^{M} C_{mix}^i = 1 \]

Here \( M \) is a number of mixture components and \( C_{mix}^i \) is the mixture coefficient, a weighting coefficient, of the \( i \)th mixture component. These coefficients can be varied, therefore allowing for modification of the contributions they make to the final model.

The number of components \( M \) can change throughout the course of optimisation. The algorithm starts with a single component, representing a simple single-peak normal distribution, a new component is then added based on the information obtained from the processed generation of individuals and is assigned a small \( C_{mix}^i \).

Use of the joint normal kernels was introduced in the work by Bosman and Thierens (1999) within the framework of the Iterated Density Estimation Algorithm (IDEA). In case of the joint normal kernels approach, each of the sampled solutions is assigned a Gaussian distribution which is centred at the solution’s location within the parameter space. It can be viewed as an extreme case of mixture distributions described previously, where the standard deviation for each of the kernels is very small.

3.3.2 Non-Gaussian models

The following section describes three different extensions of a standard univariate EDA into the continuous domain, all of which are based on the idea of using histograms as a probabilistic model for the continuous search domain.

First, an algorithm that employs marginal fixed-width (FWH) and fixed-height (FHH) histograms will be described, followed by the so-called estimation of distribution algorithm based on histogram model (HEDA). We will conclude by covering the specifications of the histogram-based PBIL (PBILh), implementation of which, for the reservoir history-matching optimisation problems, is the core subject of this thesis.
FWH/FHH marginal histogram models

Such terms as fixed-width (FWH) and fixed-height (FHH) histograms were introduced by Tsutsui et al. (2001), who proposed to use marginal histograms to model promising solutions in a continuous domain.

In case of a FWH type approach, the search space of each optimisation variable \( V_i \), defined by the given range of values \([\text{min}_i, \text{max}_i]\), is divided into \( B_i \) equal bins \((i = 1, \ldots, N, \text{ where } N \text{ is a total number of variables})\) forming histograms \( H_i \) (Fig. 3.7, c).

Then the estimated probability density \( P_{FWH}^i(b) \) of the bin \( b^i_j \) \((j = 1, \ldots, B_i)\) is calculated as a frequency count of the number of individuals that were drawn from that particular bin and selected for reproduction, normalised over the total size of the population involved in the process.

Fixed-height histogram approach creates a histogram for each parameter in which each bin has a constant height \( 1/B_i \). From the frequency point of view this means that throughout the course of optimisation number of sampled points in each bin stays constant. The thing that evolves is the width of the bin. Here bins in possible optima regions become narrower, which increases the accuracy of the underlying distribution estimate (Fig. 3.7, d).

Tsutsui et al. (2001) reported a slightly superior performance of FHH type representation over the FWH one. The authors explained this by suggesting that by narrowing the width of the bins corresponding to the possible optima locations, the algorithm was able to perform a much more precise sampling in those regions.

EDA based on histogram model (HEDA)

An introduction and some testing results of the algorithm can be found in Ding et al. (2006). As well as FWH, HEDA operates through the construction of a marginal histogram for each of the optimisation parameters (Fig. 3.7, c). However, the emphasis of the algorithm is not to explicitly evolve a population of the fittest individuals but
3.3. Continuous EDAs

to do it implicitly by solely evolving the probability distribution.

First, each marginal histogram $H_i$ is initialised to represent a uniform distribution. These are fixed-width histograms with bins of equal height ($1/B_i$). The initial population of individuals is sampled from this uniform distribution.

After all the individuals in the current generation have been evaluated, the algorithm ranks them and selects the top $50\%$ fittest individuals for further processing. The accumulation learning strategy is then proposed to update the histogram model. It is updated according to two kinds of information: historical ($H_i^H$) and current ($H_i^C$) information.

$$H_i(j) = \alpha H_i^H(j) + (1 - \alpha) H_i^C(j)$$

Where $H_i^C$ is a histogram model representative of the preselected subpopulation of individuals.

The height of each bin is dependent on the ranking and the position of each individual (i.e. different rankings of the individuals lead to different increments of the bins). The authors of the proposed approach consider this to be a linear relationship. If $N$ best individuals of the population are selected, the increment that the $k$th best individual ($k \leq N$) will make to the bin which it belongs to, will be:

$$\Delta h_k^i = \frac{(N - k + 1)}{\sum_{l=1}^{N} l} = \frac{2(N - k + 1)}{N(N + 1)}$$

which will make the current histogram $H_i^C$

$$H_i^C(j) = \sum_{k=1}^{N} \Delta h_k^i \cdot \delta_{jk}$$

$$\delta_{jk} = \begin{cases} 1 & \text{if } \min_j^i \leq V_k^i \leq \max_j^i \\ 0 & \text{otherwise} \end{cases}$$

here $V_k^i$ stands for the value of the $i$th variable of the $k$th best individual, $\min_j^i$ and $\max_j^i$ are the lower and upper bounds of bin $j$ of the $i$th variable.
**3. Introduction to Estimation of Distribution Algorithms**

**Histogram-Based PBIL**

PBILh was introduced by Yuan and Gallagher (2003) (Fig. 3.7, c). It is similar to HEDA in that it tries to extract more information (than just a simple frequency count, in case of FWH/FHH) from the processed generation of individuals and integrates it into the new updated probability model.

The main difference between the histogram-based PBIL and the two previously discussed techniques is that it skips the selection step. Information from the entire ensemble of individuals in each generation is used to construct a representative probability model (histogram). We regard this as an important characteristic of the algorithm, since it utilizes all the available information that is provided, while still implicitly being able to ignore the extremely under performing individuals. Probability model updating is performed according to the following rule:

\[ H(t + 1) = (1 - \alpha)H(t) + \alpha H_{\text{temp}}(t) \]

HEDA attempted to capture the topology of the solution space more accurately by ranking the individuals in preselected set and incrementing probabilities of the corresponding bins in histograms relative to these ranking positions. PBILh, in turn, explicitly uses the quantitative measure of the quality of generated individuals - their fitness - to construct a representative probability model, which it calls a *temporary histogram*. This temporary histogram \( H_{\text{temp}}(t) \) is then integrated with the histogram that was used to generate the given set of individuals \( H(t) \) in order to produce an updated model. The updated model \( H(t + 1) \) is then used to sample the new generation of candidate solutions.

**3.4 Summary**

A brief introduction to the class of Estimation of Distribution Algorithms has been given. We have introduced algorithms suitable for optimisation of discrete as well
as continuous search spaces. It was shown that higher order EDAs are able to encode and sample models with multivariate dependencies between their parameters, however even the univariate-type EDAs have been demonstrated to be a powerful optimisation tool. The categorisation of the discussed algorithms is summarised in Table 3.1.

Within the framework of reservoir history-matching optimisation, we are interested in the class of continuous univariate EDAs based on histogram models.

We consider PBILh to be more substantial in reflecting the true probability distribution over the space of all possible solutions due to the way it constructs the temporary histogram $H_{temp}$ (generation-specific underlying distribution estimate). Two major positive characteristics of such approach were highlighted. First being the absence of the selection step, which means that algorithm utilizes all the available information from the processed candidate solutions, even those with low fitness. Second, and the most important characteristics, is that an individual’s fitness is used to construct and update the probability model. In this way we believe PBILh has better chances of accurately approximating the true underlying probability distribution.

Table 3.1: Classification of existing EDAs according to the complexity of models they use and representation of candidate solutions

<table>
<thead>
<tr>
<th>Representation of candidate solutions →</th>
<th>Complexity of the probability model</th>
<th>Discrete</th>
<th>Continuous</th>
</tr>
</thead>
<tbody>
<tr>
<td>No interactions</td>
<td></td>
<td>PBILb</td>
<td>PBILh</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PBILc</td>
<td>HEDA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FWH/FHH</td>
<td>IDEA</td>
</tr>
<tr>
<td>Bivariate interactions</td>
<td></td>
<td>BMoDA</td>
<td>IDEA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MIMIC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMIT</td>
<td></td>
</tr>
<tr>
<td>Multivariate interactions</td>
<td></td>
<td>BOA</td>
<td>IDEA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FDA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>EcGA</td>
<td></td>
</tr>
</tbody>
</table>
PBILh as well as its binary parent PBIL, will be discussed in greater detail in the following Chapter 4. Its general workflow, specifications, basic application examples and control parameters will be presented and analyzed.
Population-Based Incremental Learning algorithm

Population-Based Incremental Learning algorithm (PBIL) is one of the earliest and simplest model-based evolutionary algorithms. It belongs to the class of univariate Estimation of Distribution Algorithms which means that all optimisation parameters are treated (sampled) independently of each other. PBIL was first introduced by Baluja (1994) and was originally designed for optimisation of binary search spaces. This implementation of the algorithm is referred to as *binary Population-Based Incremental Learning* algorithm or PBILb.

The algorithm has recently been extended to optimisation of continuous domains by Yuan and Gallagher (2003) who introduced a notation of the *histogram-based PBIL* or PBILh.

In this chapter both discrete and continuous versions of the researched algorithms will be described in detail and their implementations will be illustrated on two test problems (discrete and continuous). We will also study the impact of the control parameters on the performance of the algorithm.

The chapter will be concluded by outlining the application framework of the histogram-based PBIL for reservoir history-matching optimisation problem.

4.1 Binary PBIL

In a binary representation of the algorithm (PBILb) a Bernoulli random variable is employed as a probability model for each bit in the solution string, which is collected into a real-valued vector. This vector is referred to as a *probability* or a *prototype*.
vector, with each element independently representing the probability of generating a 1 in a corresponding bit. During the course of evolution, each element of the vector is updated towards the best individual(s) of the current generation. This is achieved using the following equation

\[ P(t + 1) = (1 - \alpha)P(t) + \alpha \frac{1}{n} \sum_{j=1}^{n} X_j \] (4.1)

where \( P(t) \) is the probability vector at current generation \( t \), \( X_j \) is the \( j \)th individual from this generation of \( n \) individuals, and \( \alpha \) is the learning rate (\( \alpha \in 0 \ldots 1 \)) (Höhfeld and Rudolph (1997)).

Equation (4.1) represents the general updating rule for the algorithm, which following the common practice has been reduced to the form

\[ P(t + 1) = (1 - \alpha)P(t) + \alpha X_{\text{best}}(t) \] (4.2)

where only the best individual \( X_{\text{best}}(t) \) in the population of \( n \) individuals is used to update the probability vector.

The general workflow of the algorithm is illustrated in Fig. 4.1, it starts with initialisation and is then split into three main steps: sampling step, evaluation step and updating step.

**Initialisation**

Given \( N_{\text{var}} \) optimisation variables, each represented by a \( N_b \)-bit binary vector, a probability vector of a size \( N_{\text{var}} \times N_b \) is created. Each position in the vector is then initialised to 0.5 to represent a uniform distribution.

**Sampling step**

1. At a given time \( t \) a corresponding probability vector \( P(t) \) will consist of \( N_{\text{var}} \times N_b \) entries, each of them will represent an estimate of the probability of 1 being
4.1. Binary PBIL

![Flowchart of Binary PBIL](image)

**Figure 4.1: General workflow of the binary PBIL**

generated at a current binary position $i \in 0 \ldots (N_{var} \times N_b - 1)$

$$P(t) = (p_0, p_1, \ldots, p_{N_{var} \times N_b - 1}), \ p_i \in [0 \ldots 1]$$

2. For each new individual $X_j$ in current generation $t$ a vector of uniform random numbers $R_j(t)$ of the same length as $P(t)$ is generated

3. Corresponding positions in vectors $P(t)$ and $R_j(t)$ are compared and a new individual $X_j(t) = [x_0^j, x_1^j, \ldots, x_{N_{var} \times N_b - 1}^j]$ is created according to the following rule

$$x_i^j(t) = \begin{cases} 1 & \text{if } p_i(t) > r_i^j(t) \\ 0 & \text{otherwise} \end{cases} \quad (4.3)$$
4. Population-Based Incremental Learning algorithm

4.1 Evaluation step

Once individual $X_i$ is generated as a binary vector, if the solution space being optimised is continuous - it is then transformed into a decimal value and used in the forward model calculation. This is the first stage of the evaluation step. The main aim of this step is to provide the entire population of individuals with a numerical evaluation of their fitness, in order to be able to differentiate well-performing from the under-performing ones. And in a particular case of binary PBIL - to be able to identify the best-performing individual(s).

4.1.1 Discrete problem optimisation with PBILb

A 10-bit long OneMax problem (see Appendix B) was chosen to demonstrate performance of the algorithm. An example of a manually performed single iteration of the binary version of PBIL is given in Appendix A. The objective function we seek
to optimise here is the difference between the maximum possible sum of OneMax vector elements ($\sum_i x_i = 10$) and the sampled sum.

OneMax is a binary-coded discrete optimisation problem; therefore, use of the binary-based implementation of PBIL to optimise it is completely justified.

Table 4.1 contains the control parameter settings for PBILb. We set up the OneMax problem within the PBILb framework as a 10-parameter optimisation problem, with each of the optimisation parameters corresponding to each of the ten bit positions within the OneMax vector. Each of the parameters is coded as a 1-bit vector, which, when appended, form the 10-bit solution string.

As one can see from the Fig. 4.2 it took under 140 individual evaluations and 7 generations (probability vector estimates) for PBILb to sample the optimum – a OneMax vector with 1s in each of its positions. Only in the 5th generation did the algorithm start sampling individuals fitter than those sampled randomly in the first generation. Here the algorithm was stopped as soon as the optimal solution was sampled.

Looking at the evolution of the probability vector, it can be seen that for all vector positions apart from that of the 10th position, probability of 1 being sampled has steadily increased throughout the 7 processed generations.

4.1.2 Continuous problem optimisation with PBILb

PEAKS function, described in greater detail in Appendix B, represents a continuous optimisation problem with two parameters $X$ and $Y$. This function is essentially a mixture of multiple gaussian distributions and is therefore characterised by the presence of multiple optima. Since the function is known a priori, we know that its maximum is located at

$$\max(f^*(X, Y)) = f^*(0.012, 1.524) = 8.0484$$
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parameters to optimise, N</td>
<td>10</td>
</tr>
<tr>
<td>Size of the binary string $\text{numbBit}_i$, $i \in 0 \ldots (N - 1)$</td>
<td>1</td>
</tr>
<tr>
<td>Learning rate $\alpha$</td>
<td>0.1</td>
</tr>
<tr>
<td>Size of initial generation $\text{SizeFirstGen}$</td>
<td>20</td>
</tr>
<tr>
<td>Size of generation $\text{SizeGen}$</td>
<td>20</td>
</tr>
<tr>
<td>Size of entire population $M_{\text{eval}}^{\text{max}}$</td>
<td>1000</td>
</tr>
<tr>
<td>Termination criteria</td>
<td>$MF = 0$ or $M_{\text{eval}} = M_{\text{eval}}^{\text{max}}$</td>
</tr>
</tbody>
</table>

**Table 4.1:** OneMax function optimisation - PBILb algorithm setup

**Figure 4.2:** OneMax function optimisation with PBILb - Individual-wise (a) and generation-wise (b) evolution of the model quality with time. The solid red line in figure (b) corresponds to the 50% percentile of the distribution of misfit values in each generations, lower and upper error bars are plotted for 10% and 90% percentiles respectively.

**Figure 4.3:** OneMax function optimisation with PBILb - Evolution of the prototype vector in time.
4.1. Binary PBIL

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parameters to optimise, $N$</td>
<td>2</td>
</tr>
<tr>
<td>Size of the binary string $\text{numBit}_i$, $i \in 0 \ldots (N - 1)$</td>
<td>10</td>
</tr>
<tr>
<td>Learning rate $\alpha$</td>
<td>0.1</td>
</tr>
<tr>
<td>Size of initial generation $\text{SizeFirstGen}$</td>
<td>20</td>
</tr>
<tr>
<td>Size of generation $\text{SizeGen}$</td>
<td>20</td>
</tr>
<tr>
<td>Size of entire population $M_{\text{max eval}}$</td>
<td>1000</td>
</tr>
<tr>
<td>Termination criteria</td>
<td>$M_{\text{eval}} = M_{\text{max eval}}$</td>
</tr>
</tbody>
</table>

Table 4.2: PEAKS function optimisation - PBIL\textsubscript{b} algorithm setup

![Figure 4.4: PEAKS function optimisation with PBIL\textsubscript{b} - Individual-wise (a) and generation-wise (b) evolution of the model quality with time. The solid red line in figure (b) corresponds to the 50\% percentile of the distribution of misfit values in each generations, lower and upper error bars are plotted for 10\% and 90\% percentiles respectively.](image)

![Figure 4.5: PEAKS function optimisation with PBIL\textsubscript{b} - Evolution of the prototype vector in time.](image)
therefore we deal with a minimisation problem with our misfit (objective) function defined as follows:

\[ MF = |8.0484 - f^*(X, Y)| \]

We choose to code the decimal values of the parameters in 10-bit long binary strings. The resulting solution string, which PBILb will be operating on, therefore becomes 20-bit long. The remaining settings of the experimental algorithm setup are given in Table 4.2.

Our aim in performing such a test is to show to the reader the performance of the discrete binary-based optimisation technique (PBILb) when optimising a continuous solution space.

Given the selected set of control parameters for the algorithm, one may observe a slow convergence of PBILb when optimising a continuous type problem. As it can be seen in the Fig. 4.4 (b), PBILb takes over 30 generations before demonstrating any significant improvement in the fitness values of the sampled models. Such tendency can also be identified in the evolution of the prototype vector positions. Most of them converge to a steady growth or start gradually decreasing soon after the 30th processed generation (for example positions #2 to #7, #17 and #18).

### 4.2 Histogram-Based PBIL

PBILh was introduced by Yuan and Gallagher (2003) as an extension of the original binary algorithm into continuous domain.

The key feature of PBILh is that for each variable \( V_i \) \( (i \in 0\ldots N_{\text{var}} - 1) \) a histogram with \( B_i \) probability bins between its upper and lower bounds is maintained. In the limit of many trials it is expected that each histogram will approximate the marginal pdfs that would be obtained from a full Bayesian analysis. The major advantages of such histogram-based model representation are:

- The histogram can be created quickly and easily as each point is processed
No assumption is made about the initial distribution of data points, which makes the approach very flexible. However, if any amount of “reliable” prior information is available, it can easily be integrated into optimisation process by means of specifying a non-uniform prior distribution;

Due to the initial assumption of complete independence of optimisation variables, the problem being solved decreases in size from $N_{var}^B$ to $N_{var} \times B$;

Another significant difference of PBILh over binary PBIL (Eq. (4.2)), as well as a number of previously researched histogram-based techniques (see Section 3.3.2), is that it uses the information from all data points from the current generation to update the probability model, not just the fittest individual(s). This is well illustrated in the general updating rule for PBILh:

$$H(t + 1) = (1 - \alpha)H(t) + \alpha H_{\text{temp}}(t)$$

where $H(t)$ and $H(t + 1)$ are equivalent to $P(t)$ and $P(t + 1)$ in Eq. (4.2) respectively, $H_{\text{temp}}(t)$ is a temporary histogram representing statistics drawn from the current set of evaluated individuals in given generation $t$ as opposed to information from only the best individual in this generation $X_{\text{best}}$.

Bin values in the histograms represent fitness values not frequency count. In this case fitness value is used to represent the goodness of a particular range that a bin stands for. And as it is very likely that a number of individuals will fall in the same bin, instead of incrementing its value the highest fitness among the individuals will be adopted as a bin value.

This is illustrated in the Fig. 4.6, where fitness values of four individuals are evaluated, with two of them ($Ind_2$ and $Ind_3$) corresponding to the same bin range, and two other individuals $Ind_1$ and $Ind_4$ belonging to two different
ranges of variable values (different bins). In case of a standard frequency-based approach (Fig. 4.6, a), one would expect to find bins corresponding to Ind$_1$ and Ind$_4$ to be of equivalent height and the bin containing both Ind$_2$ and Ind$_3$ to be twice higher.

In the Fig. 4.6 (b) however, an estimate of individual’s fitness is plotted against the range along which a given variable (which together with other variables forms this individual) is defined. Here one can see that due to the difference in fitness values of Ind$_1$ and Ind$_4$ their corresponding bins have different heights. At the same time, the bin containing both Ind$_2$ and Ind$_3$ is not incremented in its value, but instead the highest fitness value of the two individuals (Ind$_3$) is adopted as the bin height.

The basic framework of the algorithm is given in Fig. 4.7. Similar to the binary version of PBIL, we start with initialisation process and continue through sampling, evaluation and updating steps within the PBILh workflow.

**Initialisation**

The algorithm starts by initialising all bins in the initial histograms $H_i$ to a fraction of the selected number of bins $B_i$ to represent a uniform distribution.
4.2. Histogram-Based PBIL

Evaluate individuals (misfit)

Sample population of individuals from $H_i(t)$

Transform evaluate misfit into fitness estimate

Create temporary histograms $H_{temp}^i(t)$

Initialize each bin to 0

Pick up an individual and locate the corresponding bin in each $H_{temp}^i(t)$. If individual’s fitness value is greater than the current bin value, update the bin value by the fitness value of the individual.

Until all individuals in the generation are processed

Initialize each bin in histogram $H_i(t)$ to $1/B_i$

Normalize $H_{temp}^i(t)$

Update $H_i(t)$ towards the normalized $H_{temp}^i(t)$

$H_i(t+1) = (1-\alpha)H_i(t) + \alpha H_{temp}^i(t)$

Stopping criteria met

Figure 4.7: General workflow of the histogram-based PBIL
4. Population-Based Incremental Learning algorithm

Figure 4.8: Sampling of discrete (a and b) and continuous (c and d) variables in PBILh

Sampling step

First generation of individuals is generated using these uniform distributions according to the principle illustrated in Fig. 4.8.

Evaluation step

Once all individuals in current generation have been evaluated a new empty temporary histogram $H_{\text{temp}}^i$ is created for each variable. One by one individuals are processed and corresponding bins are localised in $H_{\text{temp}}^i$, their values are updated when the new fitness value is higher than current bin value or left the same otherwise. Once all the individuals from the current generation have been processed, the $H_{\text{temp}}^i$ are normalised to represent a probability distribution.

Updating step

Histograms $H_i$ are updated towards the newly constructed $H_{\text{temp}}^i$ using the general updating rule (Eq. (4.4)), and updated probability models (histograms) are used to sample next generation of individuals.
4.2.1 Discrete problem optimisation with PBILh

An example of a manually performed single iteration of the histogram-based version of PBIL, when optimising the OneMax problem, is given in Appendix A.

Since we have considered optimising a 10-bit OneMax problem, PBILh will evolve ten separate histograms for each of the OneMax vector positions. Since each of this positions can take values of either 0 or 1, each of the 10 histograms will contain only two bins. The height of the first and the second bins will correspond respectively to the probability of 0 and 1 being generated in the given position of the OneMax vector. The values of the second bin in all ten histograms evolved by PBILh can be compared to the probability vector values evolved by PBILb. The remaining settings of the algorithm are given in Table 4.3.

Judging by the evolution of the misfit function values of the sampled models, and specifically the generation-wise lack of improvement in mean fitness of the sampled models (Fig. 4.9, b), one can conclude that PBILh works very inefficiently for the given type of discrete problem.

Such behaviour can also be observed when looking at the evolution of the marginal posterior distribution estimates for each of the OneMax string positions in Fig. 4.10. For most of the vector positions, with the exception of bits #2, #3, #6 and #7, bins corresponding to 1 being generated in the corresponding OneMax vector position had a higher and consistently increasing probabilities. However, the speed with which the probability of the second bin being resampled increased was too slow too quickly and efficiently guide the search within the parameter space. Fig. 4.11 shows marginal probability distribution estimate for all ten OneMax vector positions.

4.2.2 Continuous problem optimisation with PBILh

The histogram-based version of Population-Based Incremental Learning algorithm was designed specifically for optimising continuous problem domains. Here, we investigate the efficiency of its application when optimising a simple problem with
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parameters to optimise, N</td>
<td>10</td>
</tr>
<tr>
<td>Number of bins in histograms $nmbBins_i, i \in 0 \ldots (N - 1)$</td>
<td>2</td>
</tr>
<tr>
<td>Learning rate $\alpha$</td>
<td>0.1</td>
</tr>
<tr>
<td>Size of initial generation $SizeFirstGen$</td>
<td>20</td>
</tr>
<tr>
<td>Size of generation $SizeGen$</td>
<td>20</td>
</tr>
<tr>
<td>Size of entire population $M_{eval}^{max}$</td>
<td>1000</td>
</tr>
<tr>
<td>Termination criteria</td>
<td>$MF = 0$ or $M_{eval} = M_{eval}^{max}$</td>
</tr>
</tbody>
</table>

Table 4.3: OneMax function optimisation - PBILh algorithm setup

Figure 4.9: OneMax function optimisation with PBILh - Individual-wise (a) and generation-wise (b) evolution of the model quality with time. The solid red line in figure (b) corresponds to the 50% percentile of the distribution of misfit values in each generations, lower and upper error bars are plotted for 10% and 90% percentiles respectively.
Figure 4.10: OneMax function optimisation with PBILh - Evolution of the marginal probability distribution estimate (histogram) throughout the course of optimisation. Here black corresponds to lower probability, and white - to the higher one.

Figure 4.11: OneMax function optimisation with PBILh - Estimate of the marginal probability distribution (histogram) for binary vector positions at the end of optimisation cycle
two continuous parameters – the MATLAB’s PEAKS function (see Appendix B).

PBILh evolves a separate histograms for each of the two optimisation parameters $X$ and $Y$. Histograms are initialised to represent a uniform distributions and these distributions are used to sample first generation of individuals. It was decided to discretize the histograms into 20 bins each, providing that in each sampled generation 20 new individuals will be sampled.

As can be seen in Fig. 4.12 (b) the algorithm is able to achieve a noticeable improvement in the mean generation-wise fitness after only 10 processed generations. The trend is echoed in Fig. 4.13 where evolution of the histograms (estimates of the posterior marginal distributions) of optimisation parameters is shown. Starting at the 5th generation a clear shape of the histograms is established. From that point onwards parameter ranges characterised by the higher likelihood of resampling slowly become dominant (see parameter $Y$ in Fig. 4.13).

It takes PBILh a further 10 generations to establish a stable shape of the marginal posterior distribution estimates, where likelihood of resampling the parameter ranges corresponding to the underperforming regions of parameter space is reduced to zero. From this point (20th generation onwards), the algorithm drastically improves the average fitness of the sampled models (Fig. 4.12, b).

Fig. 4.14 and 4.15 demonstrate the final posterior probability distribution estimate at the end of the 50th generation of PBILh.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parameters to optimise, $N$</td>
<td>2</td>
</tr>
<tr>
<td>Number of bins in histograms $nmbBins_{i}$, $i \in 0 \ldots (N-1)$</td>
<td>20</td>
</tr>
<tr>
<td>Learning rate $\alpha$</td>
<td>0.1</td>
</tr>
<tr>
<td>Size of initial generation $SizeFirstGen$</td>
<td>20</td>
</tr>
<tr>
<td>Size of generation $SizeGen$</td>
<td>20</td>
</tr>
<tr>
<td>Size of entire population $M_{eval}^{\text{max}}$</td>
<td>1000</td>
</tr>
<tr>
<td>Termination criteria $M_{eval} = M_{eval}^{\text{max}}$</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4: PEAKS function optimisation - PBILh algorithm setup
4.3. Comparison of PBILb and PBILh

Figure 4.12: PEAKS function optimisation with PBILh - Individual-wise (a) and generation-wise (b) evolution of the model quality with time. The solid red line in figure (b) corresponds to the 50% percentile of the distribution of misfit values in each generations, lower and upper error bars are plotted for 10% and 90% percentiles respectively.

Figure 4.13: PEAKS function optimisation with PBILh - Evolution of the marginal probability distribution estimate (histogram) throughout the course of optimisation.
4. Population-Based Incremental Learning algorithm

Figure 4.14: PEAKS function optimisation with PBILh - Estimate of the marginal probability distribution (histogram) for model parameters at the end of optimisation cycle

Figure 4.15: PEAKS function optimisation with PBILh - Estimate of the marginal cumulative distribution for model parameters at the end of optimisation cycle. Red line represents cdf of a uniform distribution
4.3 Comparison of PBILb and PBILh

Analyzing the results for OneMax problem optimisation with both versions of the algorithm, the following observations can be made:

- A much quicker convergence to the single global optimum is observed for the binary version of PBIL. In this case the algorithm reached the sought after location in the parameter space in 7 generations (125 sampled individuals), while PBILh was able to reach it only after 12 generations (225 sampled individuals) (Fig. 4.2 and Fig. 4.9);

- Despite being able to reach the optimum location at a later stage, PBILh failed to demonstrate even slow but steady improvement in mean generation misfit which was evident in the case of PBILb (solid line in Fig. 4.2, b and Fig. 4.9, b). Such a trend can also be observed in the Fig. 4.16, where a comparison of the cumulative distributions of sampled model misfit for PBILb and PBILh runs is plotted. It can even be argued that the fact that PBILh sampled the optimum solution was more the effect of random sampling rather than the probability model guided sampling;

- An evolution of the prototype vector probabilities can be interpreted identically with the marginal pdf of the second bin in parameter histograms. We observe that the range of probability encoded in the prototype vector is 0.5 . . . 0.7, while that in the marginal histogram - 0.3 . . . 0.7. Which means that the proportion of 1s being sampled in binary positions by PBILb would still be higher than that sampled by PBILh.

Such superior performance of the binary version of PBIL over its histogram-based extension is mainly due to the type of parameter space being optimised - a discrete binary space. In this case, PBILb’s tactics of learning from the fittest individuals in processed generations and moving swiftly in their direction paid off. As
for PBILh, it attempted to sample parameter space in order to approximate the true underlying distribution, not just identify a single and possibly local solution. This is why its performance was much slower.

Based on the results of both PBILb and PBILh runs for the optimisation of the considered continuous problem (PEAKS function) the following observations can be made:

- As opposed to the case with OneMax problem, for PEAKS function optimisation PBILh has demonstrated a much quicker convergence speed than PBILb. This can be seen in Fig. 4.4, where by the 20th generation of PBILh, algorithm has demonstrated a steep drop in its median generation-wise misfit, while in the case of PBILb similar objective function behaviour was only apparent after more than 40 processed generations;

- The above noted observations are also evident if one analyzes the evolution of the prototype vector positions (Fig. 4.5) for PBILb and histogram evolution (Fig. 4.13) for PBILh. Only after around 40 processed generations do most of prototype vector positions converge very close to either 0 or 1, which means
that the diversity of sampled individuals will decrease and their fitness will increase steadily or stay at the level of the fittest individual found so far.

As for PBILh, histogram evolution demonstrates that the algorithm is able to identify under- and well-performing regions of parameter space relatively quickly (after about 10 generations). However, only after about 20 generations does the probability of under-performing regions decrease low enough to prevent the algorithm from re-sampling those areas again, or sampling them but with a much lower frequency.

While being able to cut down on the exploration of the lower performing areas of parameter space for the case of PEAKS problem, PBILh was still able to identify and re-sample multiple optima regions continuously, without converging to a single point solution. In case of the PBILb, while it took a while for the algorithm to find its way around the parameter space, once it did, it mainly concentrated its search on a very small neighbourhood around the global optimum (Fig. 4.17).

Fig. 4.18 gives a good illustration to the differences in parameter space sampling strategies the two algorithms have. General quality and amount of fit models sampled by PBILh is higher at the starting stages of optimisation than that of PBILb (Fig. 4.18, a). At the final stages, when PBILb have identified the location of the “sought” optimum region, it re-samples more of the models in that area and therefore raises the average fitness of population compared to PBILh, which continues to sample simultaneously in all identified “potential” optima regions (Fig. 4.18, b).

It is important to note however, that results of only single runs of both versions of PBIL were considered in this particular example for both types of test problems. This was done mostly to illustrate the general performance trends and specification of both binary and histogram-based implementation of the researched algorithm are characterised by. A more substantial testing is performed in the following section, where influence of the algorithm’s control parameters will be studied and its convergence stability properly assessed.
4. Population-Based Incremental Learning algorithm

Figure 4.17: PEAKS function optimisation - Comparison of the solution space sampling by PBILb (a) and PBILh (b). Scale shows model misfit values

Figure 4.18: PEAKS function optimisation - Comparison of the cumulative distribution of model misfit for PBILb and PBILh. Figure (a) corresponds the first half of the processed population, while figure (b) corresponds to the entire population of sampled individuals
4.4 PBIL control parameters

Both binary and histogram-based PBIL algorithms share two common control parameters which can significantly influence their performance. These are learning rate parameter and population size (initial and current). Apart from them each of the algorithms is further dependant either on the size of binary representation of optimisation variables (in case of PBILb) or histogram discretisation (in case of PBILh).

As it was noted before (see Section 4.2), in PBILh we update the probability model of given set of individuals relative to their fitness values. These fitness values are obtained from the misfit (also called objective) functions through some form of transformation. Which is why last, but not at all the least important parameter for effective performance of PBILh is this exact transformation of misfit into fitness, or in other words - fitness scaling.

This section is dedicated to the analysis of the discussed control parameters.

4.4.1 Learning rate

Learning rate $\alpha$ (Eq. (4.2) and Eq. (4.4)) is the most important tuning parameter for the algorithm. There is always a trade-off between accuracy and convergence speed, which one should consider when choosing the value of $\alpha$.

As pointed out by Baluja (1994) if the learning rate is high, the initial population generated will largely determine the focus of the search, without enabling the algorithm to effectively explore the parameter space. In the case of the absence of local optima, a high learning rate may work well, but if multiple local optima are present, a smaller learning rate will ensure a wider exploration. Parameter values lie in the range 0 to 1, but in practice, a range of 0…0.4 was found satisfactory for most problems, as reported by Hughes (1998).
From the general updating rule in Eq. (4.2) we can derive the following

\[
\begin{align*}
t = 0, & \quad P_0 = P_{\text{init}} \\
t = 1, & \quad P_1 = (1 - \alpha)P_{\text{init}} + \alpha X_{1}^{\text{best}} \\
t = 2, & \quad P_2 = (1 - \alpha)P_1 + \alpha X_{2}^{\text{best}} \\
& \quad P_2 = (1 - \alpha) \left( (1 - \alpha)P_{\text{init}} + \alpha X_{1}^{\text{best}} \right) + \alpha X_{2}^{\text{best}} = \\
& \quad = (1 - \alpha)^2 P_{\text{init}} + \alpha(1 - \alpha) X_{1}^{\text{best}} + \alpha X_{2}^{\text{best}} \\
t = 3, & \quad P_3 = (1 - \alpha)P_2 + \alpha X_{3}^{\text{best}} \\
& \quad P_3 = (1 - \alpha) \left( (1 - \alpha)^2 P_{\text{init}} + \alpha(1 - \alpha) X_{1}^{\text{best}} + \alpha X_{2}^{\text{best}} \right) + \alpha X_{3}^{\text{best}} = \\
& \quad = (1 - \alpha)^3 P_{\text{init}} + \alpha(1 - \alpha)^2 X_{1}^{\text{best}} + \alpha(1 - \alpha) X_{2}^{\text{best}} + \alpha X_{3}^{\text{best}} \\
& \quad \vdots \\
t = n, & \quad P_n = (1 - \alpha)P_{n-1} + \alpha X_{n}^{\text{best}}
\end{align*}
\]

so the general expression for deriving the resulting probability vector information is:

\[
P_n = (1 - \alpha)^n P_{\text{init}} + \alpha \sum_{t=1}^{n} X_{t}^{\text{best}} (1 - \alpha)^{n-t} \tag{4.5}
\]

and for histogram-based representation

\[
H_n = (1 - \alpha)^n H_{\text{init}} + \alpha \sum_{t=1}^{n} H_t (1 - \alpha)^{n-t} \tag{4.6}
\]

where \(P_t, H_t\), \((t \in 1 \ldots n)\) represent a probability vector (histogram) at the end of \(t\) processed generations.

Fig. 4.19 displays the fraction of information from the initial probability model preserved in its final version after a given number of generation. Each line represents a particular learning rate value, ranging from 0.01 to 0.4. One may observe that for a smaller learning rate of 0.01, after 100 processed generations, around 40% of a resulting probability model will still depend on its initial state. While for a much higher \(\alpha = 0.4\), this percentage becomes very insignificant after 10-12 generations,
Figure 4.19: Fraction of information preserved from initial generation

Figure 4.20: Fraction of information preserved from each generation
and completely loses its influence in due course.

The influence of an intermediate probability model, representative of each generation is pictured in Fig. 4.20. Here a slightly different dynamics may be observed. For smaller learning rates, as it was described above, most of the information is preserved from the initial probability model. The remaining part of the information comes from the updated probability models at each generation. Smaller learning rate ensures that a relatively equal degree of information from each of the recorded probability models is taken into account (in Fig. 4.20, bold blue line corresponding to $\alpha = 0.01$), and with the increase of $\alpha$ a sharp decrease in significance of a much earlier constructed probability models can be observed (in Fig. 4.20, bold red line corresponding to $\alpha = 0.4$).

To understand the influence of learning rate parameter on both algorithm’s ability of identifying optimal solutions as well as the speed with which they converges to this solutions, we have performed tests on both OneMax and PEAKS (Appendix B) test problems using the following values of $\alpha$: 0.01, 0.05, 0.1, 0.2 and 0.3.

For each of the learning rate values PBILb and PBILh were both run 50 times,

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parameters to optimise, N</td>
<td>2</td>
</tr>
<tr>
<td>Number of bins in histograms $nmbBins_i, i \in 0\ldots(N-1)$</td>
<td>20</td>
</tr>
<tr>
<td>Size of the binary string $nmbBit_i, i \in 0\ldots(N-1)$</td>
<td>10</td>
</tr>
<tr>
<td>Learning rate $\alpha$</td>
<td>0.01, 0.05, 0.1, 0.2 and 0.3</td>
</tr>
<tr>
<td>Search rate $s$</td>
<td>0.0</td>
</tr>
<tr>
<td>Size of initial generation $SizeFirstGen$</td>
<td>20</td>
</tr>
<tr>
<td>Size of generation $SizeGen$</td>
<td>20</td>
</tr>
<tr>
<td>Size of entire population $M_{max}^{eval}$</td>
<td>1000</td>
</tr>
<tr>
<td>Fitness scaling coefficient $C_{scale}$</td>
<td>1</td>
</tr>
<tr>
<td>Termination criteria for OneMax</td>
<td>$MF = 0$ or $M_{eval} = M_{max}^{eval}$</td>
</tr>
<tr>
<td>Termination criteria for PEAKS</td>
<td></td>
</tr>
</tbody>
</table>

*Table 4.5: Influence of learning rate $\alpha$ on the performance of PBIL. Test setup*
these runs were performed in order to assess the robustness and stability of the algorithm. The uniform random number generator was reset for each of the runs. In this way we simulated a case where the algorithms had different starting positions in parameter space in each of the runs.

Remaining test setup parameters are given in Table 4.5. Statistics obtained from the performed tests is presented in Fig. 4.21-4.26 and can be summarised as follows:

- The value of $\alpha$, as it was stated earlier in this section, largely determines the size of steps which PBIL takes in order to explore parameter space. When the algorithm’s termination criteria is a maximum number of function evaluations, then the larger the learning rate the higher is the portion of the sampled high-fitness individuals in the total population. In case of the OneMax problem, for some of the tests runs of PBILb with $\alpha = 0.01$, the algorithm was only able to find the optimal solution towards the end of the optimisation cycle (in about 5% of the conducted 50 test runs), while for $\alpha = 0.3$, the worst performing run was able to reach optimum after just over 200 function evaluations. The convergence speed of PBILb improved significantly with the increase in learning rate (Fig. 4.23), and the spread of misfit function cumulative distributions became much narrower (Fig. 4.21).

- For the PEAKS problem, on the other hand, a slightly different trend is visible (Fig. 4.22 and 4.24). Here, a steep decrease in median of the misfit function (from under $MF = 8$ to just over $MF = 3$) is triggered by only a small increase in learning rate from 0.01 to 0.05. The trend continues with median $MF$ reaching the value of around 1 for $\alpha = 0.3$.

For the lower values of the learning rate, regardless of the starting point, PBILh was able to converge to the same group of solutions (areas within the parameter space with equal or similar fitness). While for the higher learning rates, the spread in cumulative distribution of the sampled individual fitness was much
4. Population-Based Incremental Learning algorithm

Figure 4.21: OneMax function optimisation with PBILb - Influence of the learning rate parameter $\alpha$ on the quality of sampling. Cumulative distributions of model misfit over 50 separate runs of the algorithm are plotted for each of the considered values of $\alpha$ (LR).

Figure 4.22: PEAKS function optimisation with PBILh - Influence of the learning rate parameter $\alpha$ on the quality of sampling. Cumulative distributions of model misfit over 50 separate runs of the algorithm are plotted for each of the considered values of $\alpha$ (LR).
Figure 4.23: OneMax function optimisation with PBILb - Influence of the learning rate parameter $\alpha$ on the convergence speed of the algorithm. Here $Q_{10}$, $Q_{50}$ and $Q_{90}$ over 50 separate runs of the algorithm are plotted for each of the considered values of $\alpha$ (LR).
4. Population-Based Incremental Learning algorithm

![Graphs showing 90%, 50%, and 10% quantiles over 50 runs for different learning rates](image)

**Figure 4.24**: PEAKS function optimisation with PBILh - Influence of the learning rate parameter $\alpha$ on the convergence speed of the algorithm. Here $Q_{10}$, $Q_{50}$ and $Q_{90}$ over 50 separate runs of the algorithm are plotted for each of the considered values of $\alpha$ (LR).
4.4. PBIL control parameters

Figure 4.25: PEAKS function optimisation with PBILh - Influence of the learning rate parameter $\alpha$ on the stochastic variability in the estimate of posterior marginal probability distribution for variable $X$. Bin height corresponds to the 50% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10% and 90% percentiles respectively.
Figure 4.26: PEAKS function optimisation with PBILh - Influence of the learning rate parameter $\alpha$ on the stochastic variability in the estimate of posterior marginal probability distribution for variable $Y$. Bin height corresponds to the 50% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10% and 90% percentiles respectively.
higher, with some of the runs failing to reach optima or even the near-optima region. In these extreme cases, the algorithm’s steps within the parameter space are too large and there is a chance that PBIL can simply skip optima that are too narrow;

- The above point can also negatively influence the accuracy of the estimate of the underlying probability distribution in the case of PBILh. The histogram-based version of the algorithm was designed in a specific way in order to provide an implicit parallel search within the parameter space, so that it can identify and re-sample a number of existing optima regions simultaneously. If one considers the experimental results presented in Fig. 4.26, it is clear that while for smaller learning rates algorithm detects the existing local, as well as global optima with a very low degree of uncertainty, for the higher values of $\alpha$ this local optima region is barely sampled, since the algorithm rapidly focuses its search around the best performing region. If compared to the reference analytically derived marginal pdf estimates shown in Appendix B (Fig. B.3), it can be seen that PBILh is however not very efficient in detecting the local optima located in the middle of the given range of the $Y$ variable.

### 4.4.2 Search rate

One of the drawbacks of the current updating rule in both short (Eq. (4.2)) and extended (Eq. (4.1)) forms is that while each of the positions in the prototype vector evolve to either 1 or 0 during the course of optimisation, once one of the terminal values has been reached, if the element has converged in a wrong direction, there is no way for it to be corrected.

A few attempts have been made to compensate or eliminate this disadvantage. For instance Hughes (1998) used an additional search rate control parameter, which represented the distance by which the final convergence level was offset by. In other words a search rate parameter represented a probability of selecting one instead of
zero after an infinite number of generations. So by increasing the value of the search rate one prevents elements of the prototype vector converging exactly to 0 or 1. An introduction of the new search rate parameter transforms the updating rule in equation (4.2) in the following way:

\[ P(t + 1) = ((1 - \alpha)P(t) + \alpha X_{t+1}^{\text{best}})(1 - f) + \frac{f}{2} \]  

(4.7)

\[ f = \frac{2s\alpha}{1 - 2s(1 - \alpha)} \]

where \( s \) is a search rate.

The evolution of a single position of a probability vector over the course of optimisation is demonstrated in the Fig. 4.27. The solid line corresponds to a classical updating rule (Eq. (4.2)) with learning rate \( \alpha \) as the only parameter, whereas a dashed line represents the influence of an additional search rate parameter (Eq. (4.7)). As one may observe, while the original algorithm setup provides final convergence of the current bit probability to 0 at around 90 – 100 generations mark, the setup which uses the search rate limitation shows similar behaviour converging to 0.1, but after about 40 additional generations a sudden increase in bit’s probability is observed with a continuous tendency to rise more.

This particular example illustrates evolution of only one of the five bit positions of the chosen binary representation of the variable, hence the unsteady nature of the modified algorithm performance. This can be explained by the occurrence of a particular combination of bit probabilities (values) at any given moment in time. So while the original updating rule takes us to one of the existing optima in the parameter space, and once found - stays there, a modified updating rule provides further exploration of the parameter space even if current algorithm position is in a local optimum.

The PEAKS test problem (Appendix B) was chosen and the following values of \( s \) were considered: 0.0, 0.05, 0.1 and 0.2. For each of the parameter values 50 runs of PBILb were performed. Remaining algorithm parameters are given in Table 4.6.
4.4. PBIL control parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parameters to optimise, N</td>
<td>2</td>
</tr>
<tr>
<td>Number of bins in histograms ( nmbBins_i ), ( i \in 0 \ldots (N - 1) )</td>
<td>20</td>
</tr>
<tr>
<td>Size of the binary string ( nmbBit_i ), ( i \in 0 \ldots (N - 1) )</td>
<td>10</td>
</tr>
<tr>
<td>Learning rate ( \alpha )</td>
<td>0.1</td>
</tr>
<tr>
<td>Search rate ( s )</td>
<td>0.0, 0.05, 0.1 and 0.2</td>
</tr>
<tr>
<td>Size of initial generation ( SizeFirstGen )</td>
<td>20</td>
</tr>
<tr>
<td>Size of generation ( SizeGen )</td>
<td>20</td>
</tr>
<tr>
<td>Size of entire population ( M_{eval}^{\text{max}} )</td>
<td>1000</td>
</tr>
<tr>
<td>Fitness scaling coefficient ( C_{scal} )</td>
<td>1</td>
</tr>
<tr>
<td>Termination criteria for OneMax</td>
<td>( MF = 0 ) or ( M_{eval} = M_{eval}^{\text{max}} )</td>
</tr>
</tbody>
</table>

Table 4.6: Influence of search rate \( s \) on the performance of PBIL. Test setup

![Influence of search rate s on the performance of PBIL. Test setup](image)

Figure 4.27: Influence of the search rate parameter on the evolution of a single position of a probability vector
Figure 4.28: PEAKS function optimisation with PBILb - Influence of the search rate parameter $s$ on the prototype vector evolution

Figure 4.29: PEAKS function optimisation with PBILb - Influence of the search rate parameter $s$ on the quality of sampling. Cumulative distributions of model misfit over 50 separate runs of the algorithm are plotted for each of the considered values of $s$
4.4. PBIL control parameters

Figure 4.30: PEAKS function optimisation with PBILb - Influence of the search rate parameter $s$ on the convergence speed of the algorithm. Here $Q_{10}$, $Q_{50}$ and $Q_{90}$ over 50 separate runs of the algorithm are plotted for each of the considered values of $s$. 
Statistics obtained from the performed tests is presented in Fig. 4.28-4.30. As expected, increase in the search rate has resulted in a much slower convergence of the algorithm. While some of the prototype vector positions have evolved to the level of \((0 + s)\) or \((1 - s)\) and their probability has not changed throughout the course of optimisation (i.e. positions #12 – #14, #10 and #11), other positions have changed completely, such as position 9 for example, where probability of generation 1 has gone from around 0 (in the case where no search rate was used) to up to 0.7 in the case of \(s = 0.2\). Another characteristic, which can be clearly visible from Fig. 4.29, is that with the increase in search rate, spread of the cumulative distribution over the quality of sampled models has narrowed down.

In this test example, introduction of the search rate control parameter does not seem to have a positive effect on the performance of binary PBIL.

### 4.4.3 Population size

The size of the initial population may have a significant effect on the resulting convergence speed and accuracy. Given a particular learning rate value (as mentioned above), the initial generation can be largely responsible for chosen search directions and, if not informative enough (did not represent a wide and diverse enough group of samples), may lead to a poor exploration of parameter space.

To test this hypothesis, PBILh was run with the following sizes of initial population: 25, 50, 75 and 100 individuals. The size of the total ensemble in each of the runs was 1000 individuals.

Each of the four setups was run 50 times to be able to assess the algorithms stability and robustness. In these particular test runs a uniform prior distribution was used for all the optimisation variables, and a learning rate of 0.1 was chosen.

Test run setup for the assessment of the influence of the size of initial generation as well as size of a generation in general was as follows:

Both OneMax and PEAKS test problems were chosen and the following values
of $SizeFirstGen$ and $SizeGen$ were considered: 20, 40, 80, 160 and 320. When the influence of the size of initial generation $SizeFirstGen$ was tested all the subsequent generations had a size of 20 individuals, for $SizeGen$ testing all generations throughout the course of optimisation were of an equal size. For each of the generation size values 50 runs of both PBILb and PBILh were performed. The remaining algorithm parameters are given in Table 4.7.

The statistics obtained from the performed tests is presented in Fig. 4.31-4.42. In order to better understand and explain the role that population size plays in PBIL we will summarize influence of, firstly, size of only the initial generation $SizeFirstGen$, and secondly - size of a single generation (in this case we have considered all generations to have equal size, including the initial one) $SizeGen$. It should be noted that the summary given below assumes that regardless of the selected size of single generation, the total size of population (maximum number of evaluated individuals was set to 1000).

$SizeFirstGen$ has shown to have the following effects on the performance of both binary and histogram-based versions of PBIL:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parameters to optimise, $N$</td>
<td>2</td>
</tr>
<tr>
<td>Number of bins in histograms $nmbBins_i, i \in 0\ldots(N - 1)$</td>
<td>20</td>
</tr>
<tr>
<td>Size of the binary string $nmbBit_i, i \in 0\ldots(N - 1)$</td>
<td>10</td>
</tr>
<tr>
<td>Learning rate $\alpha$</td>
<td>0.1</td>
</tr>
<tr>
<td>Search rate $s$</td>
<td>0.0</td>
</tr>
<tr>
<td>Size of initial generation $SizeFirstGen$</td>
<td>20, 40, 80, 160 and 320</td>
</tr>
<tr>
<td>Size of generation $SizeGen$</td>
<td>20, 40, 80, 160 and 320</td>
</tr>
<tr>
<td>Size of entire population $M_{\text{eval}}^{\text{max}}$</td>
<td>1000</td>
</tr>
<tr>
<td>Fitness scaling coefficient $C_{\text{scale}}$</td>
<td>1</td>
</tr>
<tr>
<td>Termination criteria for OneMax</td>
<td>$MF = 0$ or $M_{\text{eval}} = M_{\text{eval}}^{\text{max}}$</td>
</tr>
</tbody>
</table>

Table 4.7: Influence of the size of generation $SizeGen$ on the performance of PBIL. Test setup
4. Population-Based Incremental Learning algorithm

Figure 4.31: OneMax function optimisation with PBILb - Influence of the size of initial generation $SizeFirstGen$ on the quality of sampling. Cumulative distributions of model misfit over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter.

Figure 4.32: PEAKS function optimisation with PBILb - Influence of the size of initial generation $SizeFirstGen$ on the quality of sampling. Cumulative distributions of model misfit over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter.
4.4. PBIL control parameters

Figure 4.33: OneMax function optimisation with PBILb - Influence of the size of initial generation $Size_{FirstGen}$ on the convergence speed of the algorithm. Here $Q_{10}$, $Q_{50}$ and $Q_{90}$ over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter.
Figure 4.34: PEAKS function optimisation with PBILh - Influence of the size of initial generation \( \text{SizeFirstGen} \) on the convergence speed of the algorithm. Here \( Q_{10}, Q_{50} \) and \( Q_{90} \) over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter.
4.4. PBIL control parameters

Figure 4.35: PEAKS function optimisation with PBILh - Influence of the size of initial generation on the stochastic variability in the estimate of posterior marginal probability distribution for variable $X$. Bin height corresponds to the 50% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10% and 90% percentiles respectively.
Figure 4.36: PEAKS function optimisation with PBILh - Influence of the size of initial generation on the stochastic variability in the estimate of posterior marginal probability distribution for variable $Y$. Bin height corresponds to the 50% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10% and 90% percentiles respectively.
4.4. PBIL control parameters

- From the general updating rule of PBIL, especially its histogram-based version, it is clear that with a high value of $\alpha$ first generation (which is typically drawn at random) will largely determine the focus of the search within the parameter space. However, in case of the imposed limitation on the maximum number of forward model evaluations, if the learning rate is not high enough, the algorithm will carry on performing an exhaustive sampling of a parameter space in the subsequent generations. The probability model will not evolve fast enough (at the starting stages of the search) to enable the re-sampling of the areas of high fitness identified in the first generation;

- For the case of OneMax problem optimisation, in the best case scenario PBILb was able to sample the optimum in the first generation (for runs with values of $SizeFirstGen = 160; 320$) sampled at random. However in some of the cases the algorithm had to perform around 500 function evaluations before it was able to reach the optimal solution (Fig. 4.31);

- PBILh optimisation of the PEAKS function demonstrated similar behaviour to that of the OneMax problem, with the median of the spread of misfit function cumulative distributions decreasing from just over 1 ($SizeFirstGen = 20$) to just under 5 ($SizeFirstGen = 320$). As is shown in Fig. 4.34, in case of the larger $SizeFirstGen$, the portion of the under-performing individuals within the total population is much higher (mostly due to the sole contribution of the initial generation);

- However, an important observation can be made regarding the estimates of the marginal probability distributions (histograms) of the optimisation variables. After performing and processing an equal number (1000) of function evaluations, for all the considered sizes of initial generation, the actual distribution estimates don’t look that different (Fig. 4.35-4.36). The conclusion here is that, if no prior information about the parameter space or the problem being
4. Population-Based Incremental Learning algorithm

Figure 4.37: OneMax function optimisation with PBIL - Influence of the size of generation SizeGen on the quality of sampling. Cumulative distributions of model misfit over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter.

Figure 4.38: PEAKS function optimisation with PBIL - Influence of the size of generation SizeGen on the quality of sampling. Cumulative distributions of model misfit over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter.
Figure 4.39: OneMax function optimisation with PBILb - Influence of the size of generation SizeGen on the convergence speed of the algorithm. Here Q10, Q50 and Q90 over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter.
Figure 4.40: PEAKS function optimisation with PBILh - Influence of the size of generation SizeGen on the convergence speed of the algorithm. Here Q10, Q50 and Q90 over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter.
4.4. PBIL control parameters

Figure 4.41: PEAKS function optimisation with PBILh - Influence of the size of generation SizeGen on the stochastic variability in the estimate of posterior marginal probability distribution for variable $X$. Bin height corresponds to the 50% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10% and 90% percentiles respectively.
Figure 4.42: PEAKS function optimisation with PBILh - Influence of the size of generation \( \text{SizeGen} \) on the stochastic variability in the estimate of posterior marginal probability distribution for variable \( Y \). Bin height corresponds to the 50\% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10\% and 90\% percentiles respectively.
solved is available and the first generation is to be sampled at random, PBILh does not require a large initial random sample within the parameter space in order to effectively drive future search within the parameter space.

*SizeGen* has been shown to have the following effects on the performance of both binary and histogram-based versions of PBIL:

- Increasing the size of generation in both PBILb (OneMax problem) and PBILh (PEAKS problem), while maintaining the total maximum size of the population at the same level has resulted in the sharp decrease in the convergence speed of the algorithms (Fig. 4.37-4.38). In some cases (*SizeGen* = 320), PBILb was not even able to locate the optima. With the median of the misfit function cumulative distribution increasing from just under 2, to over 7 for the case of PEAKS function optimisation with PBILh;

- Due to the fixed size of the entire population, with an increasing size of generations their number decreased, so did the number of probability model updates that took place during the optimisation. Which is why marginal pdf estimates for smaller sizes of generation are much more evolved from the initial (uniform) distribution than those of the larger sized generation cases (Fig. 4.41 and Fig. 4.42).

### 4.4.4 Size of the binary representation of discrete variables: *PBILb*

Binary representation of optimisation variables is a very important point that needs to be considered carefully before optimisation takes place. The size of the chosen binary representation can largely determine the algorithm’s efficiency as it basically represents a degree of discretisation adopted for the solution space being optimised.

Given binary number $B$ of a particular size $n$, its decimal representation can be derived in the following way:

$$D_n = B_1 \cdot 2^{(B-1)} + B_{i+1} \cdot 2^{(B-2)} + B_{i+2} \cdot 2^{(B-3)} + \ldots + B_n \cdot 2^0$$
Table 4.8: Influence of the size of binary representation on the performance of PBIL. Test setup

Fig. 4.43 represents the number of possible combinations or, alternatively, a maximum integer value that can be encoded by a binary string of a given length. In our particular case this specific dynamics is important as it influences the discretisation of parameter space for each optimisation variable.

For example, let us choose a particular binary representation for a variable \( v \)—\( n \)-bit string. Variable \( v \) is defined within the range \( (v_{\min}, v_{\max}) \). Then this range will be discretised with the following intervals:

\[
\Delta v = \frac{v_{\max} - v_{\min}}{D_{n_{\text{max}}}}
\] (4.8)

If the intervals \( \Delta v \) are too large, the algorithm will not be able to sample particular areas of parameter space, which may actually represent one of the existing optima, efficiently. On the other hand, too fine a discretisation may result in an unnecessary refinement in the parameter space in areas of lower significance.

PEAKS function (Appendix B) was chosen for testing and the following values of \( \text{nmbBit} \) were considered: 5, 10, 15 and 20. For each of the parameter values 50 runs of PBILb were performed. Test run setup is given in Table 4.8.

Effects of a continuous solution space discretisation can be seen in the case where
4.4. PBIL control parameters

Figure 4.43: Maximal decimal number that can be encoded in a binary string of a given length

Figure 4.44: PEAKS function optimisation with PBILb - Influence of the size of binary representation of variables nmbBit on the quality of sampling. Cumulative distributions of model misfit over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter.
4. Population-Based Incremental Learning algorithm

Figure 4.45: PEAKS function optimisation with PBILb - Influence of the size of binary representation of variables \(nmbBit\) on the convergence speed of the algorithm. Here \(Q_{10}, Q_{50}\) and \(Q_{90}\) over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter.
4.4. PBIL control parameters

length of the binary representation of optimisation variables was set to 5 (Fig. 4.44). One may also observe a considerable degree of instability in convergence speed for PBILb with such binary representation (spread in cumulative misfit function distributions over 50 test runs of the algorithm), compared to the performance of PBILb for \( nmbBit = 20 \). Due to the limitation on the values that function variables can take, quality of the sampled candidate solutions also deteriorates for too small binary representations (Fig. 4.45).

4.4.5 Histogram discretisation: PBILh

Histogram discretisation in PBILh is similar to the binary representation length in PBILb in the way it influences algorithm’s performance.

When initialising the histogram representing the probability distribution for a particular variable, it is important to choose an appropriate width of the bins (\( \varepsilon \)) to provide the necessary precision during optimisation. In the presence of multiple optima too coarse a histogram will fail to recognize a very narrow optimum, while a very fine histogram can potentially produce a blurred probability model.

A number of papers have been published on the subject of effective histogram discretisation, one of the earliest was a work by Scott (1979), who believed that an optimal choice of \( \varepsilon \) requires knowledge of the true underlying density, which is rare. Gaussian density was then suggested to be used as an underlying standard and the following expression was introduced:

\[
\varepsilon = 3.49 \cdot s n^{(-1/3)}
\]  

(4.9)

where \( s \) is an estimate of standard deviation and \( n \) is a size of available samples.

For the case of a two-optima problem with a much narrower global optimum than the local one, Tsutsui et al. (2001) propose to use the following relationship

\[
\varepsilon = \frac{w_1 \times h_1}{2 \times h_2}
\]  

(4.10)
where $w_1$ is the width of the global optimum, $h_1$ and $h_2$ are global and local optima values respectively.

However if one does not know the topology of the search space beforehand it is impossible to justify any of the input parameter values in the above relationship. A recommendation here might be to carry out a number of test runs defining the optimal bin width.

PEAKS test problem (Appendix B) was chosen in order to illustrate the importance of the accurate choice of the histogram discretisation scheme. Fig. 4.46 shows the level of accuracy in PEAKS function representation, which can be achieved by the discretisation of its solution space into given number of bins. One can see that for cases with number of bins less than 20, certain features of the function surface may be lost (due to interpolation).

Histograms representations with the following number of bins ($nmbBins$) were considered for testing: 5, 10, 20 and 75. For each of the parameter values 50 runs of PBILh were performed. Test run setup for the assessment of the influence of histogram discretisation is given in Table 4.9.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parameters to optimise, $N$</td>
<td>2</td>
</tr>
<tr>
<td>Number of bins in histograms $nmbBins_i$, $i \in 0 \ldots (N - 1)$</td>
<td>5, 10, 20 and 75</td>
</tr>
<tr>
<td>Learning rate $\alpha$</td>
<td>0.1</td>
</tr>
<tr>
<td>Search rate $s$</td>
<td>0.0</td>
</tr>
<tr>
<td>Size of initial generation $SizeFirstGen$</td>
<td>20</td>
</tr>
<tr>
<td>(75 for $nmbBins = 75$)</td>
<td></td>
</tr>
<tr>
<td>Size of generation $SizeGen$</td>
<td>20</td>
</tr>
<tr>
<td>(75 for $nmbBins = 75$)</td>
<td></td>
</tr>
<tr>
<td>Size of entire population $M_{\text{eval}}^\text{max}$</td>
<td>1000</td>
</tr>
<tr>
<td>Fitness scaling coefficient $C_{\text{scale}}$</td>
<td>1</td>
</tr>
<tr>
<td>Termination criteria for OneMax $MF = 0$ or</td>
<td></td>
</tr>
<tr>
<td>$M_{\text{eval}} = M_{\text{eval}}^\text{max}$</td>
<td></td>
</tr>
</tbody>
</table>

*Table 4.9: Influence of histogram discretisation on the performance of PBIL. Test setup*
4.4. PBIL control parameters

**Figure 4.46**: Influence of the optimal histogram discretisation on the accuracy of the underlying distribution (or solution space) approximation. Plotted data are that of the Matlab’s PEAKS function (Appendix B). Data legend: black, blue, green and red lines correspond to 5, 10, 20 and 75-bin discretisation respectively.

**Figure 4.47**: PEAKS function optimisation with PBILh - Influence of the histogram discretisation nmbBins on the quality of sampling. Cumulative distributions of model misfit over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter.
Figure 4.48: PEAKS function optimisation with PBILh - Influence of the histogram discretisation nmbBins on the convergence speed of the algorithm. Here Q_{10}, Q_{50} and Q_{90} over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter.
4.4. PBIL control parameters

Figure 4.49: PEAKS function optimisation with PBILh - Influence of the histogram discretisation on the stochastic variability in the estimate of posterior marginal probability distribution for variable $X$. Bin height corresponds to the 50% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10% and 90% percentiles respectively.
4. Population-Based Incremental Learning algorithm

Figure 4.50: PEAKS function optimisation with PBILh - Influence of the histogram discretisation on the stochastic variability in the estimate of posterior marginal probability distribution for variable $Y$. Bin height corresponds to the 50% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10% and 90% percentiles respectively.
The statistics obtained from the performed tests is presented in Fig. 4.47-4.50 and can be summarised as follows:

- Although PBILh was able to correctly identify the locations of optima regions for all investigated histogram discretisation options, in the case of smaller values of $nmbBins$ it was not able to shrink the radius of the area being searched and re-sampled, therefore resulting in extensive and unnecessarily sampling of mediocre regions of parameter space. Here PBILh failed its main task - to accurately approximate the true underlying marginal probability distribution;

- The optimal performance of PBILh (within the considered setup scenarios) was achieved for $nmbBins = 20$, in this case median value of the cumulative misfit function distributions (over the 50 conducted test runs) was around 1, which is considerable improvement over 5 in the case of $nmbBins = 5$;

- It is important to note however, and this is backed up by the results obtained for $nmbBins = 75$, that if the bin width is too small, the resulting histogram can result in much slower algorithm convergence, and depending on the relationship between $nmbBins$ and $SizeGen$ - a sparse and an inaccurate probability distribution estimate.

### 4.4.6 Fitness function: PBILh

In standard GAs one seeks a way to qualitatively characterize and rank the available set of individuals in a given generation. This ranking can then be used in combination with a variety of selection schemes to define a subset of individuals from this generation, which will be carried through to the next evolution step.

The evaluation of the quality of a single individual (model) is usually done by means of a so-called fitness function ($FF$). The fitness function essentially characterizes the reproduction capability of an individual. For a numerical or optimisation problem, the fitness function is usually identical to the objective function, or is oth-
otherwise obtained from the objective function through some simple transformation step.

In order to progress, evolutionary algorithms need to operate with a population of individuals with enough fitness difference among them to be able to drive further search within the parameter space. The conclusion here is that the way one defines \( FF \) may be critical to the overall success of the optimisation process (Dumitrescu (2000)).

In case of a ranking metric, even with minor differences in fitness values of the current population, typical GA might be able to perform a selection step. For the continuous histogram-based implementation of PBIL, fitness is used to construct an estimate of the likelihood distribution. Here it is critical that the \( FF \) is sensitive enough to provide us with an adequate estimate of the underlying distribution.

Further to the general calculation of fitness function, one also has to consider whether this quantity will be calculated independent of the other individuals in the population or relative to them. The later case results from the fact that in the later stages of the evolutionary search, spread of the quality of sampled models narrows down considerably and we no longer obtain sensitive enough likelihood estimates of the processed generations.

To evaluate the quality of each processed individual, the first step is the evaluation of the user-defined objective or misfit function (\( OF \) or \( MF \)), which indicates the goodness of a processed individual (Baeck et al. (1997)). This function is commonly calculated as a sum of squared differences between the simulated and real responses of the physical system being optimised. And in this case, an objective function minimisation problem is considered. For a maximisation problem, one expects the objective function to be an indicator of some reward to be increased.

The objective function should provide enough information to effectively drive the search in parameter space, i.e. it should not be favourable to a very narrow region in a parameter space, while completely neglecting other locations.

In general, the fitness function is introduced to convert the \( MF \) values into a
non-negative quantities, typically through the application of a scaling operator.

\[ FF(x_i) = g(MF(x_i)) \] (4.11)

where \( x_i \in X \) is a single model (individual), and \( X \) is a space of all possible solutions to the problem, bounded only by the limits chosen for the optimisation parameters of the model.

Generally, a scaling function is applied when one seeks to minimize the objective function. In this case if the global maximum value of the objective function is known, following relationship can be applied:

\[ FF(x_i) = MF_{max} - MF(x_i) \] (4.12)

where \( MF_{max} \) is a known maximum value of the objective function.

However in most of the real world optimisation problems one is not aware of the location and value of the global minimum or maximum. Which is why relationship 4.12 is transformed as follows:

\[ FF(x_i(t)) = MF_{max}(t) - MF(x_i(t)) \] (4.13)

where \( MF_{max}(t) \) is a maximum observed value of the misfit function up to time \( t \). Alternatively, one may consider the following relationship for minimisation case

\[ FF(x_i(t)) = \frac{1}{1 + MF(x_i(t)) - MF_{min}(t)} \] (4.14)

where \( MF_{min}(t) \) is a minimum observed value of the misfit function up to time \( t \), or

\[ FF(x_i(t)) = \frac{1}{1 + MF_{max}(t) - MF(x_i(t))} \] (4.15)

for maximisation case.

Since one hopes that eventually the performed search will lead to the regions of possible optima locations within the solution space, it is expected that the range
of the misfit function values will reduce as well. These values, if used directly (as is the case with Eq. (4.12)-(4.15)), will result in an understandably narrower range of fitness values. This will therefore influence the selection pressure (i.e. ability of the algorithm to sense and differentiate the potentially fitter solutions within the generation of individuals) at the later stages of the search.

In order to overcome this issue of misfit-to-fitness conversion an additional process called fitness scaling (FS) is introduced. There are a number of FS approaches described in the literature, they include:

FF as a time-varying linear transformation of the MF is defined as follows:

$$FF(x_i(t)) = \alpha \cdot MF(x_i(t) - \beta(t))$$

(4.16)

where $\alpha$ is a multiplier which takes a value of “+1” for maximisation and “−1” for minimisation problems, $\beta(t)$ - is the worst value sampled in the last few generations. Due to the fact that $\beta(t)$ is expected to decrease with time, we would observe a larger selection pressure towards the later stages of optimisation.

Sigma scaling, where the general idea is to learn the statistical properties of the distribution of misfit values and use this information (mean and standard deviation) to define the scaled fitness functions.

$$FF(x_i(t)) =
\begin{cases}
MF(x_i(t)) - (\mu^{MF(t)} - c \cdot \sigma^{MF(t)}) & \text{if } MF(x_i(t)) > (\mu^{MF(t)} - c \cdot \sigma^{MF(t)}) \\
0 & \text{otherwise}
\end{cases}$$

(4.17)

here $\mu^{MF(t)}$ and $\sigma^{MF(t)}$ are mean and standard deviation of the MF values in current generation respectively, $c$ is some constant. It is considered that $\mu^{MF(t)} - c \cdot \sigma^{MF(t)}$ represents the least acceptable objective function value for any individual to be reproduced. As the population evolves and improves, this statistics changes as well, this results in the self-adaptive selective pressure which is sensitive to the current spread in misfit values.

Power-law-based fitness scaling can be performed in the following two ways: the
simple fixed transformation according to the Eq. (4.18) and time-varying transformation (Boltzmann selection) according to the Eq. (4.19)

\[
FF(x_i(t)) = MF(x_i(t))^k
\]

(4.18)

where \(k\) is a problem- and user-dependant parameter.

\[
FF(x_i(t)) = \exp(MF(x_i(t))/T)
\]

(4.19)

where \(T\), is similar to the temperature parameter in simulated annealing, and is used here to control the magnitude of the selective pressure during the course of optimisation.

In the framework of PBILh one requires to evaluate the fitness values of the processed individuals in order to construct the temporary histogram to be used in the general updating rule of PBILh (Eq. (4.4)).

We have used the relationship in the Eq. (4.20) to convert model misfit values into fitness (likelihood) characteristics. The chosen relationship is similar to that of Eq. (4.19), but should be viewed more from the point of Bayesian statistics. Here fitness function is interpreted as Bayesian likelihood. Within a reservoir history-matching framework, this likelihood estimate relies on the model specification for the assessment of uncertainty associated with the observed production.

\[
FF(x_i(t)) = \exp(-MF(x_i(t))/C_{scale})
\]

(4.20)

where \(MF(x_i(t))\) is a misfit function of the given model \(x_i\) in current generation \(t\), and \(FF(x_i(t))\) is its fitness equivalent. The exponential function here ensures the fitness values range between 0 and 1.

But due to the nature of the exponential functions, if one operates with a wide range of misfit function values, most of the extremely high values of \(MF(x_i(t))\) (after being transformed), will result in 0 fitness. To overcome this problem a scaling coefficient \(C_{scale}\) is introduced in the Eq. (4.20) similar to the scaling parameter \(T\)
4. Population-Based Incremental Learning algorithm

<table>
<thead>
<tr>
<th>Scaling coefficient</th>
<th>Fitness ≃ 0 for</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{\text{scale}} = 1$ (No scaling)</td>
<td>$f(x_i) &gt; 50$</td>
</tr>
<tr>
<td>$C_{\text{scale}} = 10$</td>
<td>$FF(x_i) &gt; 10^2$</td>
</tr>
<tr>
<td>$C_{\text{scale}} = 10^2$</td>
<td>$FF(x_i) &gt; 10^3$</td>
</tr>
<tr>
<td>$C_{\text{scale}} = 10^3$</td>
<td>$FF(x_i) &gt; 10^4$</td>
</tr>
<tr>
<td>$C_{\text{scale}} = 10^4$</td>
<td>$FF(x_i) &gt; 10^5$</td>
</tr>
</tbody>
</table>

Table 4.10: Influence of the scaling coefficient $C_{\text{scale}}$ on the misfit-to-fitness transformation sensitivity

Figure 4.51: Influence of the scaling coefficient $C_{\text{scale}}$ on the fitness estimate sensitivity
### 4.4. PBIL control parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parameters to optimise, ( N )</td>
<td>2</td>
</tr>
<tr>
<td>Number of bins in histograms ( nmbBins_i, i \in 0 \ldots (N - 1) )</td>
<td>20</td>
</tr>
<tr>
<td>Learning rate ( \alpha )</td>
<td>0.1</td>
</tr>
<tr>
<td>Size of initial generation ( SizeFirstGen )</td>
<td>20</td>
</tr>
<tr>
<td>Size of generation ( SizeGen )</td>
<td>20</td>
</tr>
<tr>
<td>Size of entire population ( M_{\text{max}}^{\text{eval}} )</td>
<td>1000</td>
</tr>
<tr>
<td>Fitness scaling coefficient ( C_{\text{scale}} )</td>
<td>0.5, 1.0, 2.0, and 5.0</td>
</tr>
<tr>
<td>Termination criteria for OneMax</td>
<td>( \sum_i x_i = 10 ) or ( M_{\text{eval}} = M_{\text{eval}}^{\text{max}} )</td>
</tr>
</tbody>
</table>

Table 4.11: Influence of fitness scaling coefficient \( C_{\text{scale}} \) on the performance of PBIL. Test setup

![Graphs](image1.png)

**Figure 4.52:** PEAKS function optimisation with PBILh - Influence of the fitness function scaling coefficient \( C_{\text{scale}} \) on the quality of sampling. Cumulative distributions of model misfit over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter
Figure 4.53: PEAKS function optimisation with PBILh - Influence of the fitness function scaling coefficient $C_{scale}$ on the stochastic variability in the estimate of posterior marginal probability distribution for variable $X$. Bin height corresponds to the 50% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10% and 90% percentiles respectively.
4.4. PBIL control parameters

Figure 4.54: PEAKS function optimisation with PBILh - Influence of the fitness function scaling coefficient $C_{\text{scale}}$ on the stochastic variability in the estimate of posterior marginal probability distribution for variable $Y$. Bin height corresponds to the 50% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10% and 90% percentiles respectively
in Eq. (4.19). However, in our case $C_{\text{scale}}$ stays constant throughout the course of evolution.

Fig. 4.51 and Table 4.10 demonstrate the influence of the scaling coefficient on the fitness estimation sensitivity for the ensemble of models with misfit values ranging from 0 to $10^5$.

PEAKS test problem (Appendix B) was chosen for testing, and the following values of $C_{\text{scale}}$ were considered: 0.5, 1.0, 2.0 and 5.0. For each of the parameter values 50 runs of PBILh were performed. Test run setup for the assessment of the influence of the fitness scaling coefficient $C_{\text{scale}}$ is given in Table 4.11.

Statistics obtained from the performed tests is presented in Fig. 4.52-4.54. As we can see in Fig. 4.52, convergence speed and quality of sampling in PBILh increases significantly when $C_{\text{scale}}$ is decreased from 5 to 0.5. Lower values of the fitness scaling coefficient yield a much sharper differentiation between the identified multiple optima (Fig. 4.54). In cases where only one single optimum location exists, lower $C_{\text{scale}}$ values ensure that only a very narrow neighbourhood of the solution space around this location is re-sampled (Fig. 4.53). It is important to note, that, having tested four different fitness function scaling coefficients, only runs with a higher $C_{\text{scale}}$ (Fig. 4.54) were just about able locate an additional local optimum for variable $Y$ which is present in the analytically derived pdf shown in Appendix B (Fig. B.3).

4.5 Summary

In this chapter Population Based Incremental Learning algorithm was introduced. The background and principles of operation were given for two of the algorithm’s implementations: binary PBIL and histogram-based PBIL.

Two synthetic optimisation problems (OneMax and PEAKS) were used to illustrate a step-by-step workflow of the studied algorithms. These test problems were chosen in order to investigate application potential of both versions of PBIL when optimising both discrete and continuous parameter spaces. In the case where a opti-
4.5. Summary

Figure 4.55: Reservoir history-matching optimisation workflow with PBILh

Optimisation was performed within the continuous problem domain, results of the conducted tests demonstrated that PBILh considerably outperformed PBILb in terms of convergence speed, while still providing good quality of sampled solutions. PBILh was also able to sample two out of three existing optima within the solution space (for PEAKS function example), while PBILb has eventually concentrated its search in one optima region. Only PBILh runs with higher fitness scaling coefficient values were able to provide a more accurate estimates to the marginal posterior distributions of model parameters, therefore highlighting the importance of an appropriately selected $C_{scale}$.

Since most of the reservoir history-matching optimisation studies deal with optimisation of continuous reservoir model parameters such as for example porosity and permeability values, sizes and strength of the aquifers, a histogram-based ver-
sion of the algorithm was identified as the most suitable for this kind of optimisation. In Fig. 4.55 the suggested reservoir history-matching optimisation workflow with the application of histogram-based Population-Based Incremental Learning algorithm is presented.

The chapter was concluded by extensive study of the algorithm’s control parameters such as learning and search rate, size of initial and current population, problem domain discretisation (length of the binary coded solution string for PBILb and histogram discretisation in PBILh), as well as fitness function scaling strategy.

Since histogram-based implementation of PBIL is the main focus of the thesis, we note that learning rate parameter $\alpha$ and fitness function scaling coefficient $C_{\text{scale}}$, together with histogram discretisation parameter $nmbBins$ have proved to be the most influential tuning parameters of the algorithm. An important observation was also made in regards to the size of initially sampled generation of individuals. Test results have shown that PBILh is less sensitive to the size of the sample drawn in the first generation, typically uniformly, than it was expected to be. It did not seem to influence the algorithm’s ability to accurately approximate the underlying probability distribution.
A synthetic dataset, known in the industry as the IC Fault Model (Appendix C) is used here to demonstrate the application of the researched algorithm to the problem of reservoir history-matching optimisation.

It is a layer-cake type model, which is characterised by three continuous variables: fault throw and two permeabilities (in low and high permeable layers). The model contains one production and one injection well. Oil and water production volumes as well as water injection volumes are matched over 36 time steps. A complete dataset description is provided in Appendix C.

We start by introducing a new parallelisation scheme for PBILh in Section 5.1 - an asynchronous Master/Slave parallel setup. The proposed scheme is designed to be run on a heterogeneous parallel cluster. By applying the proposed scheme we aim to minimize CPU losses that may result from a variety of failures in the entire optimisation process loop, such as model sampling, evaluation and updating, software and hardware related failures.

A new definition of the misfit function is presented in Section 5.2. It is based on the general sum-of-squares form which is commonly adopted in industry. The suggested modification aims at achieving a greater sensitivity of the history-matching process to both early and late water breakthrough.

From the general optimisation point of view we perform a variety of test runs investigating the following characteristics of PBILh:

- An ability of the algorithm to perform an effective global search within the solution space (i.e. able to simultaneously sample multiple optima regions);
• As noted in Chapter 4, the learning rate is the most influential control parameter of the researched algorithm. We aim to investigate the influence $\alpha$ will have on the algorithm’s convergence speed as well as on its ability to produce good quality history matched models and accurately estimate the underlying uncertainty in model parameters;

• Due to the implemented asynchronous parallelisation scheme, the general dynamics (inter-population information communication) of PBILh will be affected. We will investigate the effects this may have on the algorithm’s performance, and specifically its sensitivity to the choice of the size of initial and current generations within such an asynchronous setting.

Results of the conducted tests will be provided in Section 5.3 and we will conclude with a short summary at the end of the chapter.

5.1 Parallelisation of EDAs

Due to the population-based nature of EDAs, they are very easy to parallelize. In cases when individual evaluation (forward model solution) is too costly in CPU terms for a standard serial EA to handle, algorithm parallelisation can provide near-linear speed-ups. Parallelisation benefits become even more promising now, when there is a much wider availability and diversity of parallel clusters on the market.

5.1.1 Overview

As noted by Cantú-Paz (1998) and Nowostawski and Poli (1999), Master/Slave, Finely and Coarsely grained models are known to be a canonical setups for parallelisation of evolutionary algorithms.
5.1. Parallelisation of EDAs

Master/Slave parallelisation scheme

The main idea behind the Master/Slave parallel model is that a single population of individuals is managed by a master process, while evaluation of individuals is transferred to slave locations.

This setup particularly suits parallel clusters of a heterogeneous type and scales up very well up to a certain amount of slave processes. However with a significantly increased amount of incorporated slave locations, communication between them and master location can become overpowering and slow down the entire process. In cases where heterogeneous systems are used one may struggle to ensure an efficient workload balancing. To overcome this particular disadvantage an asynchronous Master/Slave setup can be applied. If the master process stops and waits until all individuals from the current generation have been evaluated at slave locations and returned to the master location before the next generation can be processed, the algorithm is assumed to be synchronous (which is equivalent to a serial EA with speed being the only difference). The algorithm is asynchronous if the master location does not wait for any slow-performing processes on slave locations, but post-processes a certain (predefined) portion of a population of individuals as they come back evaluated from the slave locations. But it is very important to point out that the asynchronous nature of parallel setup results in a modified population dynamics, one that is very hard to quantify and account for. A modified population dynamics results from the variable size of each generation of processed models as well as from the fact that evaluated individuals used to construct a new generation of candidate solutions may be offsprings of a range of different generations.

Finely grained parallelisation scheme

Finely grained model for parallel EAs maintains a single population which is spatially split into sub-populations called dames. Selection and mating are restricted to a small neighbourhood which evolves separately. These neighbourhoods over-
lap to allow a certain degree of interaction among the entire pool of individuals. This setup is suited for massively parallel computers; however possible communication overheads and the need to identify and treat chosen neighbourhoods’ sizes and boundaries can result in a much more complicated setup.

Coarsely grained parallelisation scheme

Coarsely grained parallel EAs, also known as Island models operate by splitting the population into sub-populations – islands. Each island evolves separately for a certain period of time before a so-called migration takes place. During the migration period a number of individuals are swapped between the separate islands. This parallel setup performs well on a message-passing type systems (MPI) providing a significant hardware as well as island-model-based related speed-ups. Understandably the general setup of such a parallel model becomes increasingly complicated.

A good analysis of all three types of parallel EAs described in this section can be found in work by Setzkorn and Paton (2002).

5.1.2 Experimental parallel setup

In our case an asynchronous Master/Slave model was partially chosen due to its simplicity. However the aim was mainly to choose a robust and reliable parallel setup that can be stable enough to survive any possible software or hardware-related failures while providing a desired speed-up to the optimisation process.

The following are the characteristics of the chosen setup presented in Fig. 5.1:

- Population generation takes place at the Master location. The initial population is drawn at random and consists of $N_{\text{init}}$ individuals;

- Individuals are then sent to and evaluated at Slave locations;

- The probability model updating process takes place as soon as $N$ individuals have been evaluated and returned to the Master location;
5.1. Parallelisation of EDAs

- \( M \) new individuals (representing offspring) are created using the current probability model as soon as there is no individuals left in the queue, waiting to be submitted for evaluation at slave locations.

These new individuals are then sent to the Slave locations to be evaluated.

Here \( N \) can be interpreted as a minimal size of population necessary to update the probability model - a parameter required by the asynchronous nature of chosen setup.

**Figure 5.1:** An asynchronous Master/Slave parallel model setup
5.2 Misfit function definition

A weighted sum-of-squares is commonly used as an objective function in history-matching to measure the goodness of fit of a proposed solution, i.e.

\[ F(x_i) = \sum_j \left( \frac{S_j(x_i) - \mu_j}{\sigma_j} \right)^2 \]  

(5.1)

where \( x_i \) is a vector of parameter values that characterizes a particular solution, \( \mu_j \) is the \( j \)th observed measurement, \( S_j(x_i) \) is the model (simulation) prediction of the \( j \)th measurement based on the vector \( x_i \), and \( \sigma_j \) is either a subjective measure of the perceived importance of the measurement or a measure of the accuracy of the measurement. It is often convenient to make \( \sigma_j \) proportional to \( \mu_j \),

\[ \sigma_j = \alpha \mu_j \]  

(5.2)

where typically \( \alpha \in (0, 0.5) \). This means that for small \( \mu_j \) we expect a good proposed solution to match quite closely, whilst for large \( \mu_j \) a larger absolute error is permitted.

This, however, can potentially have undesirable consequences, for instance consider a series of water production measurements given by

\[ \mu_j = \max \left( 1, 10 \ln(t_j - \tau_0) \right) \]

where \( t_j \) are positive integers and \( \tau_0 \) is a positive number. An example with \( \tau_0 = 15.5 \) is given in Figure 5.2.

If we pose this as a history-matching problem where

\[ S_j(x) = \max \left( 1, 10 \ln(t_j - x) \right) \]

and

\[ F(x_i) = \sum_j \left( \frac{S_j(x) - \mu_j}{\alpha \times \mu_j} \right)^2 \]
we can calculate $F(x)$, for any given $x$, as shown in Figure 5.3. From this figure it is obvious that solutions that produce early water breakthrough are much more heavily penalised compared to those that produce late water breakthrough. This will impact on the ability of any optimisation method to find the best solution quickly.

We suggest using a modified factor to weight the individual measurements.

$$\sigma_j = \alpha \left( \frac{\mu_j + S_j(x_i)}{2} \right)$$  (5.3)

The argument used to justify this is that the uncertainty in the measurement should be related to the true (unobserved) measurement, and not the observed measurement. $\mu_j$ and $S_j(x_i)$ are two different estimates of the true measurement and we are simply using the average of these. When this principle is applied to our simple example, Figure 5.4 is obtained. Here we can see that early and late water breakthrough are treated and penalised in a very similar fashion.

Eq. (5.1) represents misfit function definition in the case where a Gaussian (normal) distribution of model errors is assumed. Similarly, in the case of log-normally
Figure 5.3: *Plot of the standard objective function against model parameter*

Figure 5.4: *Plot of the proposed objective function against model parameter*
distributed errors, the misfit function can be defined as

\[ F(x_i) = \sum_j \left( \frac{\ln(S_j(x)) - \ln(\mu_j)}{\alpha \times \ln(\mu_j)} \right)^2 \]  

(5.4)

such misfit function definition will produce the profile shown as a blue line in Fig. 5.4.

Misfit function definition given in Eq. (5.4), while excluding the simulated data errors from the standard deviation expression in Eq. (5.3) and providing a better sensitivity to the early water breakthrough timing, still produces a much lower degree of model differentiation than that based on Eq. (5.3).

Therefore, in this thesis, unless otherwise stated, we use misfit function definition given in Eq. (5.1) and Eq. (5.3).

Since, as it was stated in Section 4.4.6 (Eq. (4.20)) Bayesian likelihood is based on the misfit function value, the analytically derived marginal distribution estimates will be different for the considered old and new misfit function definitions.

The database of a total of 159,661 uniformly sampled models (IC Fault Model web page (n.d.)) was post-processed in order to generate analytically derived “true” marginal distributions of model parameters. Fig. 5.5 represents comparison between the analytically derived marginal distribution estimates based on an entire database for different misfit function definitions considered in the thesis. Fig. 5.6 on the other hand represents statistics over a subset of the entire sampled parameter space which corresponds to a modified parameter ranges introduced in 5.3.2.

The analytically derived distributions in Fig. 5.6 corresponding to the misfit function definition \( MF\#2 \) (that according to Eq. (5.3)) are to be used as a reference for algorithm performance quality control.

\section*{5.3 Results}

Previously, a number of varying setups of the IC Fault Model were used for reservoir history-matching optimisation studies.
Figure 5.5: IC Fault Model optimisation - Comparison between the analytically derived marginal distribution estimates based on an entire IC Fault Model database for different misfit function definitions. MF #1 refers to a misfit function definition used in Erbas (2007), MF #2 refers to that defined by Eq. (5.3). Red lines correspond to the location of the global optimum.

Figure 5.6: IC Fault Model optimisation - Comparison between the analytically derived marginal distribution estimates based on a subset of the IC Fault Model database for different misfit function definitions. MF #1 refers to a misfit function definition used in Erbas (2007), MF #2 refers to that defined by Eq. (5.3). Red lines correspond to the location of the global optimum.
5.3. Results

Carter et al. (2004) and Ballester and Carter (2006) have considered matching oil and water production as well as water injection quantities. Authors have used the common misfit function definition in Eq. (5.1). Parameter boundaries were selected as follows: fault throw, $ft \in (0, 60)$, good and poor quality sand permeabilities, mD respectively $k_{good} \in (100, 200)$ and $k_{bad} \in (0, 50)$.

Erbas (2007) only used oil and water production quantities in order to estimate the mismatch between the historical and simulated volumes. The author has also introduced an offset value of $10^{-6}$ in standard deviation definition in Eq. (5.2) in order to avoid the division-by-zero for the time steps before the historical water breakthrough time in Eq. (5.1).

$$\sigma = \alpha (\mu + 10^{-6})$$

Parameter ranges were maintained same as in the studies by Carter et al. (2004); Ballester and Carter (2006).

In this thesis it was decided to change the size of the optimised parameter space for the performed tests. The major argument that led to this decision was that most of the model’s HM studies reported in the literature (for instance by Carter et al. (2004); Ballester and Carter (2006) and Erbas (2007)) have indicated an existence of a strong/sharp but narrow global optima in the model and a number of wider but weaker (fitness-wise) local optima. Therefore it can be assumed that by reducing the parameter space dimensions (concentrating the search area around optimum’s location), one should be able to observe a much quicker convergence.

Parameter boundaries were modified in the following way: fault throw, $ft \in (0, 60)$, good and poor quality sand permeabilities, mD respectively $k_{good} \in (0, 200)$ and $k_{bad} \in (0, 5)$. Although the ranges of $k_{good}$ were extended by 100%, we still considered the parameter space complexity to be reduced since a much larger portion (90%) of the poor quality sand $k_{bad}$ ranges has been excluded.

In order to demonstrate the general dynamics of PBILh when optimising large
parameter spaces we performed a single test run of the HM optimisation problem setup used in Erbas (2007). Results of this test are presented in Section 5.3.1.

The following Sections 5.3.2-5.3.3 present results of the optimisation studies where modified ranges of the optimisation parameters were used. The model misfit was calculated based on both production (oil and water) and injection (water) quantities. The new suggested definition of a standard deviation $\sigma$ according to Eq. (5.3) was applied in both tests.

### 5.3.1 TEST 1 – Optimisation of a large parameter space based only on oil and water production match

As it was previously stated, the aim is to illustrate optimisation capabilities of the researched algorithm when operating in a large size solution space. The selected forward model setup details and algorithm settings are given in Table 5.1 and Table 5.2.

Fig. 5.7 demonstrates the variability in the size of processed generations throughout the optimisation process which appeared due to the asynchronous nature of the chosen parallelisation scheme. Most generations were of the preselected minimal size of 10 individuals. However, for a small portion of generations, probability model estimates were constructed based on more than 13 individuals. This indicated the presence of under-performing (with slow forward model evaluation/convergence rates) individuals, which had the parallel setup been synchronous, could and would have slowed down the optimisation process, therefore increasing its CPU cost. A relatively low learning rate ($\alpha = 0.01$) was chosen for this test case in order to ensure an extensive global search within the solution space. This choice in $\alpha$ has resulted in an overall slow speed of convergence of the algorithm (Fig. 5.8). If we consider the generation-wise evolution of the misfit function of sampled models in Fig. 5.9, we can see that the average misfit of a single generation has decreased from over 1000 to just over 200, with the overall fittest model having fitness of less than 10.

The cumulative distribution of the pool of sampled models’ misfit is plotted in
### 5.3. Results

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>True case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of variables to optimise, N</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Fault throw ( h, ) ft</td>
<td>0 . . . 60</td>
<td>10.4</td>
</tr>
<tr>
<td>Good quality sand permeability ( k_{\text{good}}, ) mD</td>
<td>100 . . . 200</td>
<td>137.6</td>
</tr>
<tr>
<td>Poor quality sand permeability ( k_{\text{bad}}, ) mD</td>
<td>0 . . . 50</td>
<td>1.31</td>
</tr>
<tr>
<td>Production series to match</td>
<td>Oil production (WOPR)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Water production (WWPR)</td>
<td></td>
</tr>
</tbody>
</table>

**Table 5.1: IC Fault Model optimisation with PBILh - TEST 1: Forward model setup**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of variables to optimise, N</td>
<td>3</td>
</tr>
<tr>
<td>Number of bins in histograms ( H_i, i \in 0 . . . (N - 1) )</td>
<td>20</td>
</tr>
<tr>
<td>Learning rate ( \alpha )</td>
<td>0.01</td>
</tr>
<tr>
<td>Size of initial generation ( \text{SizeFirstGen} )</td>
<td>20</td>
</tr>
<tr>
<td>Minimal Size of generation ( \text{SizeGen} )</td>
<td>10</td>
</tr>
<tr>
<td>Size of offspring</td>
<td>13</td>
</tr>
<tr>
<td>Size of entire population ( M_{\text{eval}}^{\text{max}} )</td>
<td>5866</td>
</tr>
<tr>
<td>Fitness scaling coefficient ( C_{\text{sclae}} )</td>
<td>100</td>
</tr>
</tbody>
</table>

**Table 5.2: IC Fault Model optimisation with PBILh - TEST 1: Algorithm setup**

**Figure 5.7: IC Fault Model optimisation with PBILh - TEST 1: Variability in the sizes of processed generations due to asynchronous nature of parallelisation scheme**
Figure 5.8: IC Fault Model optimisation with PBILh - TEST 1: Individual-wise convergence of the algorithm

Figure 5.9: IC Fault Model optimisation with PBILh - TEST 1: Generation-wise convergence of the algorithm. The solid red line corresponds to the 50% percentile of the distribution of misfit values in each generations, lower and upper error bars are plotted for 10% and 90% percentiles respectively.
5.3. Results

Figure 5.10: IC Fault Model optimisation with PBILh - TEST 1: Cumulative distribution of sampled model misfit

Figure 5.11: IC Fault Model optimisation with PBILh - TEST 1: Simulated production of the 586 fittest models, which represent $Q_{10}$ quantile of the total number of sampled models with misfit function threshold of $MF < 127$. A solid red line represents historical data.
Fig. 5.10. We use it to estimate the top 10% of the fittest models sampled by identifying the $Q_{10}$ quantile value, which in the current case is $MF = 127$. During the course of optimisation we have sampled 586 models under the $Q_{10}$ threshold, their simulated production is plotted and overlaid with the historical data in Fig. 5.11. As it was stated at the beginning of this section, we have only used oil (WOPR) and water (WWPR) production quantities in order to constrain the model. All of the displayed models are able to match the water breakthrough time in the production well, however the majority of them demonstrate varying degrees of both oil and water over production.

This is especially easy to see on the cumulative oil production (WOPT) plot, with some of the models having produced around $2 \times 10^5$ m$^3$ more oil at the end of the prediction period. Such a large fluid displacement within the model can be explained by the increased volumes of water injection (WWIR), which can also be seen in Fig. 5.11.

Another important thing to note about the produced models is the spread in simulated production quantities within the selected set of models. If we plot the frequency count of the parameter values that composed these models we will obtain statistics presented in Fig. 5.12.

Fig. 5.13 illustrates evolution of the marginal probability model estimates through-

Figure 5.12: IC Fault Model optimisation with PBILh - TEST 1: Frequency count of the sampled parameter values based on the best 586 history-matched models
out the course of optimisation. Fig. 5.14 and Fig. 5.15 correspond to the final estimates of the posterior marginal and posterior cumulative marginal distributions of the model parameters respectively.

For all three model parameters PBILh is able to pick up the general shape of the underlying distribution relatively quickly, after around 100 first generations (see Fig. 5.13), this point in the optimisation process is marked by sampling a number of considerably fitter individuals between the 70 and 80 generations (see Fig. 5.9). After that point PBILh slowly begins to concentrate its search around the parameter regions of identified highest fitness/likelihood. This can be best observed through the steady grows in likelihood of bins #2, #4 (global optimum location) and #6 for the fault throw $h$, bins #5 and #6 (global optimum location) for $k_{good}$ and bins #4 and #5 for $k_{bad}$. Bin #1 of the histogram representing the poor quality sand permeability, which corresponds to the location of the global optimum, while not receiving the highest likelihood for re-sampling, maintained its value almost stable through the entire course of optimisation. This in our opinion is a good indication of the algorithms ability to still be able to sample under-performing areas despite the already identified local optima locations (such as bins #4 and #5 of the $k_{bad}$ model parameter).

When comparing the estimated marginal pdfs shown in Fig. 5.14 and those analytically inferred from the IC Fault Model database (MF#1 in Fig. 5.5), for the given misfit function setup, two main observations can be made:

- In case of the fault throw variable, algorithm is able to correctly identify and re-sample the neighbouring region around the global optimum. PBILh however did not identify a local optima region for $30 < h < 45$ ft;

- While correctly identifying the low likelihood region within the parameter space in relation to the poor quality sand permeability ($k_{bad} > 10$ mD), the lowest considered parameter values are not given as high likelihood as that indicated in the analytically derived marginals.
Figure 5.13: IC Fault Model optimisation with PBILh - TEST 1: Evolution of the posterior marginal distribution estimates

Figure 5.14: IC Fault Model optimisation with PBILh - TEST 1: Final posterior marginal distribution estimates

Figure 5.15: IC Fault Model optimisation with PBILh - TEST 1: Final cumulative posterior marginal distribution estimates
It is important to note however, that only a single run of PBILh was conducted in a given experiment, and further tests are required in order to accurately account for the algorithm's sampling efficiency.

In order to visually study the spatial sampling capabilities of the algorithm we suggest constructing 3D parameter space cubes (for the case of three parameter forward problem) which are comprised of the 2D projections of the sampled parameter space onto each of the 2D planes (formed by all possible paired combinations of model parameters).

The main idea behind the construction of these 2D projections is presented in Fig. 5.16. Here parameter space is discretised according to the number of bins chosen for each of the parameter histograms, and the resulting 2D images are just overlapping slice-wise projections of the parameter space onto each of the possible planes. In the example presented in Fig. 5.16 we aim to construct a 2D projection of the sampled parameter space onto the $Var_1-Var_2$ plane.

The parameter space cube (or the 2D projections for that matter) is at first initialised to be empty, i.e. each of its “cells” is assigned a zero fitness. Each time a model is generated and evaluated, its position is localised within the discretised parameter space cube (corresponding 2D projections) and if the model's fitness value is higher than the current “cell” value, the “cell” is assigned this model's fitness. “Cell” value is maintained unchanged otherwise.

For the performed test, after evaluating a total of 5866 models, the resulting 3D parameter space cube is displayed in Fig. 5.17.

An added benefit of such a 2D projection-type representation of the solution space is its ability to implicitly visually assess a possible presence of any type of correlations between model parameters.

As a result, after the performed optimisation we have decided to perform the following modification in the model setup, to be used in the further testing:

- Introduce water injection quantity (WWIR) into the misfit function definition
Figure 5.16: IC Fault Model optimisation with PBILh - Procedure for constructing a 2D projection of a 3D solution space for a three-variable optimisation problem. Here $\delta$ is the width of the bin in a histogram which represents marginal probability distribution estimate of variable $\text{Var}_3$, $N$ is a total number of bins in the histogram and $\text{min}$ and $\text{max}$ define range of the variable.

Figure 5.17: IC Fault Model optimisation with PBILh - TEST 1: Image of the sampled solution space represented through its 2D projections.
as an additional constrain imposed on the sampled models, to control the fluid flow dynamics within the reservoir;

- Given the nature/shapes of the estimates of posterior marginal distribution for all three model parameters (Fig. 5.14) as well as parameter space cube (Fig. 5.17) we can see that the sampled parameter space is not characterised by a presence of multiple clearly defined and separate optima. This is why parameter ranges will be altered in order to try and produce a much more multi-modal solution space.

5.3.2 TEST 2 – Influence of different learning rate values on the performance of PBILh

Similar to the tests performed in Chapter 4 (Section 4.4.1) for the PEAKS function optimisation example, we have considered optimising IC Fault Model with PBILh that uses three different learning rate values – 0.01, 0.1 and 0.2. The forward model definition as well as algorithm setup parameters are given in Table 5.3 and Table 5.4 respectively.

For each of the learning rate values we have performed 50 identical runs in order to assess the robustness and stability of PBILh.

Similar to the dynamics observed for the PEAKS test case, the instability in the algorithm’s performance appeared to be increasing with the increase of $\alpha$ (Fig. 5.18). The higher the value of the learning rate the more weight is put on the information sampled at the later stages of optimisation. Exploratory steps within the parameter space become larger and the algorithm begins to demonstrate characteristics typical of a local optimisation. Therefore, if the first randomly generated population of PBILh fails to sample close to the location of the true global optimum, when optimising with high values of $\alpha$, algorithm might prematurely converge, or to be more correct, concentrate its search in the earlier identified local optima regions.

If we compare the quality of the fittest-ever sampled models for all three learning
Parameter & Value \\
Number of variables to optimise, $N$ & 3 \\
Fault throw $h$, ft & $0 \ldots 60$ \\
Good quality sand permeability $k_{good}$, mD & $0 \ldots 200$ \\
Poor quality sand permeability $k_{bad}$, mD & $0 \ldots 5$ \\
Production series to match & Oil production (WOPR) \\
& Water production (WWPR) \\
& Water injection (WWIR) \\

**Table 5.3: IC Fault Model optimisation with PBILh - TEST 2: Forward model setup**

Parameter & Value \\
Number of variables to optimise, $N$ & 3 \\
Number of bins in histograms $H_i$, $i \in 0 \ldots (N-1)$ & 20 \\
Learning rate $\alpha$ & 0.01, 0.1 and 0.2 \\
Size of initial generation $\text{SizeFirstGen}$ & 20 \\
Minimal Size of generation $\text{SizeGen}$ & 10 \\
Size of offspring & 13 \\
Size of entire population $M_{\text{max}}^{\text{eval}}$ & 1000 \\
Fitness scaling coefficient $C_{\text{sclae}}$ & 1000 \\

**Table 5.4: IC Fault Model optimisation with PBILh - TEST 2: Algorithm setup**

**Figure 5.18: IC Fault Model - TEST 2: Influence of the learning rate parameter $\alpha$ (LR) on the quality of sampling. Cumulative distributions of model misfit over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter.**
Figure 5.19: IC fault Model - TEST 2: Influence of the learning rate parameter $\alpha$ (LR) on the convergence speed of the algorithm. Here $Q_{10}$, $Q_{50}$ and $Q_{90}$ over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter.
rate values, we can see that for a small $\alpha = 0.01$ in most of the 50 identical test runs, the overall-fittest models are of the same or very similar quality (from just over 100 to under 300). On the other hand, in the case of a much larger learning rate ($\alpha = 0.2$) the overall-fittest model misfit ranges from just under 100 to over 500. And while, as noted, some of the optimisation runs of PBILh with higher values of the learning rate were able to sample fitter models, performance of PBILh with such a setup was seen to be unstable and highly dependant on the sampling quality (i.e. random number generator).

Fig. 5.19 illustrates the influence learning rate parameter $\alpha$ has on the algorithm’s convergence speed. As expected, decrease in the misfit function of the sampled models is much slower for a smaller values of $\alpha$.

One of the main arguments for the application of PBILh in reservoir history-matching optimisation was the algorithm’s ability to provide estimates to the marginal probability distributions of model parameters. These estimates in turn can be used to quantify the uncertainty in model parameters. Therefore it was crucial for us to investigate just how much the accuracy of sampled underlying probability distributions is influenced by the choice of the learning rate parameter.

In Fig. 5.20, Fig. 5.21 and Fig. 5.22 statistics are plotted for the final posterior marginal probability estimates of model parameters for different values of $\alpha$. Bin height in the histograms corresponds to the 50% percentile of the distribution of the final posterior distribution estimates over the 50 separate identical runs of the algorithm. Lower and upper error bars are plotted for $10\%$ and $90\%$ percentiles respectively.

In the case of the fault throw variable $h$ (Fig. 5.20) for the lowest learning rate value $\alpha = 0.01$ the shape of the sampled distribution is largely maintained the same throughout most of the 50 test runs, with the uncertainty in the bin height (parameter range likelihood) being within acceptable ranges (error bars displaying $10\%$ and $90\%$ percentiles respectively). When $\alpha$ is increased to 0.1 and further to 0.2, height of most or all (in the case of the largest $\alpha$) of the bins in the histogram approaches zero.
5.3. Results

Figure 5.20: IC Fault Model optimisation with PBILh - TEST 2: Influence of the learning rate parameter $\alpha$ (LR) on the estimate of posterior marginal probability distribution for the fault throw parameter. Bin height corresponds to the 50\% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10\% and 90\% percentiles respectively.

Figure 5.21: IC Fault Model optimisation with PBILh - TEST 2: Influence of the learning rate parameter $\alpha$ (LR) on the estimate of posterior marginal probability distribution for the good quality sand permeability parameter. Bin height corresponds to the 50\% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10\% and 90\% percentiles respectively.

Figure 5.22: IC Fault Model optimisation with PBILh - TEST 2: Influence of the learning rate parameter $\alpha$ on (LR) the estimate of posterior marginal probability distribution for the bad quality sand permeability parameter. Bin height corresponds to the 50\% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10\% and 90\% percentiles respectively.
While for the case of $\alpha = 0.01$ algorithm was able to identify three high-likelihood regions (bins #4, #8, #12 and its surrounding bins) in most of the 50 performed runs, in the case of $\alpha = 0.1$ only bin #4 seemed to have been continuously re-sampled. Runs performed with $\alpha = 0.2$ failed to consistently re-sample even bin #4.

Estimates of the posterior marginal probability distribution for the good quality sand permeability parameter $k_{\text{good}}$ have demonstrated existence of a single clear optimum in all the performed runs of PBILh, regardless of the learning rate value used (Fig. 5.21). However for the case of higher learning rates, $\alpha = 0.1$ and $\alpha = 0.2$, we can see that for some of the runs the location of the highest likelihood parameter range has shifted to bin #13 which stands for the range $k_{\text{good}} = 120 \ldots 130 \text{ mD}$, while the global optima for this parameter corresponds to the value of 137.6 mD (bin #14).

In the case of the poor quality sand permeability parameter $k_{\text{bad}}$ (Fig. 5.22), estimates of the marginal posterior distribution produced by PBILh with $\alpha = 0.01$ have not been able to capture a clear shape/topology of the underlying probability distribution. A number of high likelihood areas can just about be differentiated, and mostly by looking at the error bar’s magnitudes of the histogram bins (parameter ranges) concerned. With the increasing values of $\alpha$, exploitation abilities of the algorithm are overcoming its exploration properties and, as a result, algorithm tends to perform more localised optimisation. This, as in the case with parameter $h$ for $\alpha = 0.2$, results in a very large variations between the final posterior estimates produced by PBILh in each of the 50 test runs.

In order to illustrate what effect choice of the learning rate has on the evolution of the posterior marginal distribution estimates, we select one of the 50 performed runs for each of the two values of $\alpha = 0.01$ and 0.1. Fig. 5.23 and Fig. 5.24 show individual and generation-wise evolution of the model misfit for the given values of $\alpha$. In Fig. 5.25 and Fig. 5.26 a generation-wise evolution of the posterior marginal distribution estimates is plotted. When analyzing the four mentioned images we
5.3. Results

Figure 5.23: IC Fault Model optimisation with PBILh - TEST 2: Individual and generation-wise convergence of the algorithm ($\alpha = 0.01$). The solid red line corresponds to the 50\% percentile of the distribution of misfit values in each generations, lower and upper error bars are plotted for 10\% and 90\% percentiles respectively.

Figure 5.24: IC Fault Model optimisation with PBILh - TEST 2: Individual and generation-wise convergence of the algorithm ($\alpha = 0.1$). The solid red line corresponds to the 50\% percentile of the distribution of misfit values in each generations, lower and upper error bars are plotted for 10\% and 90\% percentiles respectively.
Figure 5.25: IC Fault Model optimisation with PBILh - TEST 2: Evolution of the posterior marginal distribution estimates ($\alpha = 0.01$)

Figure 5.26: IC Fault Model optimisation with PBILh - TEST 2: Evolution of the posterior marginal distribution estimates ($\alpha = 0.1$)
can see that PBILh with $\alpha = 0.01$ takes much longer to establish the stable enough shape of the posterior marginal estimates, therefore continuing sampling the under-performing areas of the solution space, areas which, in the case of PBILh with $\alpha = 0.1$, are quickly excluded from sampling by the algorithm.

A sharp decrease in the processed generation’s mean misfit is occurring during the first 20 generations for PBILh with $\alpha = 0.1$, the algorithm then continues sampling fitter models but overall, generation-wise quality of sampling is stabilised (Fig. 5.24, b). Such a trend is also visible in Fig. 5.26. Here, for the fault throw parameter $h$ algorithm quickly identifies two main regions (ranges of interest), and at first assigns higher likelihood to the range representing one of the local optima solutions. Through the course of optimisation the algorithm still continues sampling in the slightly more under-performing region that corresponds to the global optimum solution and eventually, after about 50 generations shifts its focus into that region, therefore demonstrating its ability to escape local minima. In the case of the good quality sand permeability $k_{good}$, PBILh is able to accurately sample and approximate its marginal posterior distribution after around 20 generations and maintain the shape of the sampled distribution stable throughout the remainder of the optimisation cycle. The true posterior marginal distribution of the poor quality sand permeability $k_{bad}$ turned out to be the hardest to approximate for PBILh. During the first 20 iterations the algorithm was able to successfully identify and sample a number of existing optima regions, including the global optimum one, but failed to maintain sampling in this region of interest during the further exploration of the solution space.

On the other hand, for PBILh with $\alpha = 0.01$, only a very slow improvement in sampled model quality is taking place during the first 60 generations of PBILh (Fig. 5.23, b). The relative breakthrough in sampled model quality at this point can be explained by an emergence of a much clearer shape of the marginal distribution estimates of the model parameters shown in Fig. 5.25. Only around the 60th generation of the algorithm do both fault throw and good quality sand permeability distri-
bution estimates begin to evolve the highest likelihood regions around the location of the global optimum. The algorithm thereafter proceeds by slowly concentrating its search in the identified parameter ranges and model misfit starts to steadily improve. As opposed to the case with higher learning rate, PBILh with $\alpha = 0.01$ was able to identify and maintain resampling the $k_{bad}$ parameter ranges corresponding to the optimum location (bin #5).

Fig. 5.27-5.28 and Fig. 5.29-5.30 present the final posterior marginal and cumulative marginal distribution of the model parameters after 1000 processed individuals respectively.

According to the approach illustrated previously in Fig. 5.16, we have constructed a 3D solution space cubes for both runs of PBILh. These solution space cubes are presented in Fig. 5.31 and Fig. 5.32. For both considered values of the learning rate parameter $\alpha$ 2D projections plotted for the good quality sand permeability parameter $k_{good}$ (i.e. $h - k_{good}$ and $k_{good} - k_{bad}$ planes) indicate existence of a single optimum within the ranges represented by bin #14 in the $k_{good}$ histogram, which correctly corresponds to the global optimum value of 137.1 mD. However, the $h - k_{bad}$ plane represents the biggest interest for us, since it does not clearly indicate a single optimal solution.

For the case of lower learning rate $\alpha = 0.01$ parameter space sampling is much wider and it is clear that algorithm has not had enough time to narrow down its search to specific areas. PBILh is, however, able to sample models within a close proximity to the global optimum location (bin #4 for $h$, bin #5 for $k_{bad}$ and bin #14 for $k_{good}$). A much wider region of the parameter space was sampled for $h = 20\ldots25$ ft and $k_{bad} = 2.2\ldots4$ mD.

PBILh with higher learning rate $\alpha = 0.1$ sampled a somewhat narrower region of the parameter space, being quick to move away from the identified underperforming regions. By looking at the produced solution space cube (Fig. 5.32), and specifically at the $h - k_{bad}$ 2D plane we can differentiate four separate potential optima regions for the combination of these variables: $h = 9\ldots18$ ft and
5.3. Results

Figure 5.27: IC Fault Model optimisation with PBILh - TEST 2: Final posterior marginal distribution estimates ($\alpha = 0.01$)

Figure 5.28: IC Fault Model optimisation with PBILh - TEST 2: Final posterior marginal distribution estimates ($\alpha = 0.1$)

Figure 5.29: IC Fault Model optimisation with PBILh - TEST 2: Final cumulative posterior marginal distribution estimates ($\alpha = 0.01$)

Figure 5.30: IC Fault Model optimisation with PBILh - TEST 2: Final cumulative posterior marginal distribution estimates ($\alpha = 0.1$)
Figure 5.31: IC Fault Model optimisation with PBILh - TEST 2: Image of the sampled solution space, constructed by projecting a 3D solution space image onto the three 2D coordinate planes: “fault throw - good quality sand”, “fault throw - poor quality sand” and “good quality sand - poor quality sand”

Figure 5.32: IC Fault Model optimisation with PBILh - TEST 2: Image of the sampled solution space, constructed by projecting a 3D solution space image onto the three 2D coordinate planes: “fault throw - good quality sand”, “fault throw - poor quality sand” and “good quality sand - poor quality sand”
5.3. Results

Figure 5.33: IC Fault Model optimisation with PBILh - TEST 2: 10% of the sampled models are plotted, with misfit value $MF < 4579$ ($\alpha = 0.01$)

Figure 5.34: IC Fault Model optimisation with PBILh - TEST 2: 10% of the sampled models are plotted, with misfit value $MF < 621$ ($\alpha = 0.1$)
PBILh with a smaller learning rate ends up sampling an overall fitter model ($MF < 100$) than that produced in the run with the higher learning rate, as Fig. 5.23 and Fig. 5.24 show. However, percentage of the fitter models within the entire pull of sampled individuals for PBILh with $\alpha = 0.1$ is larger than that for $\alpha = 0.01$. We show this by plotting the simulated production quantities together with the historical data in Fig. 5.33 and Fig. 5.34. Here, and in the remainder of the thesis, the top 10% of the sampled models in a single test are chosen based on their misfit function values. This is done mostly to provide a better graphical illustration of the goodness of the match in the sampled models, and give a large enough selection of the sampled models, in order to implicitly access possible clustering of the sampled solutions within the parameter space. This 10% of the selected models are not a $P10$ estimate of the models’ misfit distribution.

Water breakthrough time was matched by all of the filtered models for both values of $\alpha$. Both oil and water over production seems to be the common problem for the models sampled by PBILh with $\alpha = 0.01$. Such a large fluid displacement within the model can be explained by the increased volumes of water injection (WWIR) which is observed in all the selected models.

Histogram-based PBIL with a higher $\alpha$ was able to sample a larger number of fitter models quicker than the one with smaller learning rate. Here oil overproduction was also present, but was not as significant as in the previous case. Water production is well balanced by the simulated injection rates. Judging by the plotted injection trends, two groups of models can be differentiated: one of which with excessive and another with insufficient injected quantities. Both trends are mirrored in the water production values, therefore oil production stays largely unaffected.
5.3.3 TEST 3 – Influence of different sizes of initial population on the performance of PBILh

Aim of this study was to investigate the influence of the size of initial generation on the convergence speed of the algorithm, quality of the sampled models and algorithm’s ability to accurately estimate the associated uncertainty in model parameters.

For all performed test runs of PBILh we consider initial generation of individuals to be drawn at random, i.e. all three model parameters sampled from uniform distributions and independently from each other. Forward model definition and algorithm setup parameters are given in Table 5.5 and Table 5.6. Two identical tests are performed for the selected values of \( \text{SizeFirstGen} \); these are tests with high and low values of learning rate parameter \( \alpha \). Therefore, here we additionally access the joint effect the two control parameters of PBILh can have on the algorithm’s performance.

In order to assess the stability and robustness of the algorithm, 50 identical test runs have been performed for each of the considered system (forward model and algorithm itself) setups. Cumulative distributions of sampled model misfits among the 50 performed runs for each of the sizes of initial generation are plotted in Fig. 5.35 (for PBILh with \( \alpha = 0.01 \)) and in Fig. 5.36 (for PBILh with \( \alpha = 0.1 \)).

As observed in the tests performed in the Section 5.3.2 of this chapter, lower

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of variables to optimise, N</td>
<td>3</td>
</tr>
<tr>
<td>Fault throw ( h ), ft</td>
<td>0…60</td>
</tr>
<tr>
<td>Good quality sand permeability ( k_{\text{good}} ), mD</td>
<td>0…200</td>
</tr>
<tr>
<td>Poor quality sand permeability ( k_{\text{bad}} ), mD</td>
<td>0…5</td>
</tr>
<tr>
<td>Production series to match</td>
<td>Oil production (WOPR)</td>
</tr>
<tr>
<td></td>
<td>Water production (WWPR)</td>
</tr>
<tr>
<td></td>
<td>Water injection (WWIR)</td>
</tr>
</tbody>
</table>

\textbf{Table 5.5: IC Fault Model optimisation with PBILh - TEST 3: Forward model setup}
learning rate yields a much more stable sampling of the algorithm, with models of similar level of quality being produced in all 50 performed test runs. For a larger $\alpha = 0.1$, PBILh ultimately samples models of different quality in each of the 50 performed test runs. Since the algorithm becomes largely driven by (at these early stages) the information provided in the first generation of randomly sampled individuals, it tends to perform more like a local optimisation technique.

However whether the spread is small (for $\alpha = 0.01$) or large (for $\alpha = 0.1$), the overall algorithm’s dynamics is not in any way influenced by the choice of the size of initial generation of models. The three plots presented in Fig. 5.35 are almost identical, same can be observed in Fig. 5.36.

Since we can now claim that given the adopted asynchronous Master/Slave type parallelisation scheme, convergence properties of PBILh and quality of the sampled models are not affected by the choice of the initial randomly sampled population size, it is only natural to investigate the effect $SizeFirstGen$ may have on the ability of the algorithm to accurately approximate the posterior marginal probability distributions of model parameters.

Fig. 5.37, Fig. 5.39 and Fig. 5.41 present statistics over the estimates of the posterior marginal probability distributions of model parameters, which were derived from 50 performed runs for each of the considered sizes of initial generation and learning rate $\alpha = 0.01$. Fig. 5.38, Fig. 5.40 and Fig. 5.42 are plotted for the case of

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of variables to optimise, $N$</td>
<td>3</td>
</tr>
<tr>
<td>Number of bins in histograms $H_i$, $i \in 0 \ldots (N - 1)$</td>
<td>20</td>
</tr>
<tr>
<td>Learning rate $\alpha$</td>
<td>0.01 and 0.1</td>
</tr>
<tr>
<td>Size of initial generation $SizeFirstGen$</td>
<td>25, 50 and 75</td>
</tr>
<tr>
<td>Minimal Size of generation $SizeGen$</td>
<td>10</td>
</tr>
<tr>
<td>Size of offspring</td>
<td>13</td>
</tr>
<tr>
<td>Size of entire population $M_{max}$</td>
<td>1000</td>
</tr>
<tr>
<td>Fitness scaling coefficient $C_{scale}$</td>
<td>1000</td>
</tr>
</tbody>
</table>

Table 5.6: IC Fault Model optimisation with PBILh - TEST 3: Algorithm setup
5.3. Results

Figure 5.35: IC Fault Model - TEST 3: Influence of the size of initial generation \( \text{SizeFirstGen} \) on the quality of sampling (\( \alpha = 0.01 \)). Cumulative distributions of model misfit over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter.

Figure 5.36: IC Fault Model - TEST 3: Influence of the size of initial generation \( \text{SizeFirstGen} \) on the quality of sampling (\( \alpha = 0.1 \)). Cumulative distributions of model misfit over 50 separate runs of the algorithm are plotted for each of the considered values of the parameter.
$\alpha = 0.1$.

First, let us consider the results produced in the runs performed with lower learning rate values (Fig. 5.37, Fig. 5.39 and Fig. 5.41). For all the IC Fault model parameters the final posterior marginal distribution estimates, for different sizes of initial population $SizeFirstGen$, came to a very close agreement with each other. For all the considered $SizeFirstGen$ values PBILh was able to correctly identify the ranges corresponding to both global and local optima locations, in the case of two out of three model parameters. Highest likelihood estimates were produced for ranges corresponding to bin #4 for fault throw parameter $h$ and bin #14 for good quality sand permeability parameter $k_{good}$, which actually both correspond to the location of a global optimum. The marginal posterior distribution of the poor quality sand permeability $k_{bad}$ proved once again the hardest to approximate. For all considered sizes of the initial generation algorithm has failed to correctly sample the parameter range corresponding to the location of the global optimum ($1.31 \text{ mD}$).

PBILh, which was run with the higher learning rate value of 0.1 (Fig. 5.38, Fig. 5.40 and Fig. 5.42), not only produced a very unstable estimates of the marginal posterior distribution of model parameters (based on the performed 50 sensitivity runs), but they also varied significantly depending on the size of initial generation $SizeFirstGen$. We, however would like to draw the readers attention to two interesting points.

First of all, the good quality sand permeability $k_{good}$ is known to be characterised by the existence of a single optimum solution. For all three considered values of $SizeFirstGen$ PBILh has consistently assigned the highest likelihood to the range of the #14 bin in the parameter histogram, which corresponds to the location of the global optimum. Algorithm was also consistent in resampling the bin #4 of the fault throw parameter histogram most frequently in the 50 performed sensitivity runs for each of the three values of $SizeFirstGen$, although with varying degrees of uncertainty.
Figure 5.37: IC Fault Model optimisation with PBILh - TEST 3: Influence of the size of initial generation on the estimate of posterior marginal probability distribution for the fault throw parameter ($\alpha = 0.01$). Bin height corresponds to the 50% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10% and 90% percentiles respectively.

Figure 5.38: IC Fault Model optimisation with PBILh - TEST 3: Influence of the size of initial generation on the estimate of posterior marginal probability distribution for the fault throw parameter ($\alpha = 0.1$). Bin height corresponds to the 50% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10% and 90% percentiles respectively.
Figure 5.39: IC Fault Model optimisation with PBILh - TEST 3: Influence of the size of initial generation on the estimate of posterior marginal probability distribution for the good quality sand permeability parameter ($\alpha = 0.01$). Bin height corresponds to the 50% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10% and 90% percentiles respectively.

Figure 5.40: IC Fault Model optimisation with PBILh - TEST 3: Influence of the size of initial generation on the estimate of posterior marginal probability distribution for the good quality sand permeability parameter ($\alpha = 0.1$). Bin height corresponds to the 50% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10% and 90% percentiles respectively.
5.4. Summary

Figure 5.41: IC Fault Model optimisation with PBILh - Influence of the size of initial generation on the estimate of posterior marginal probability distribution for the bad quality sand permeability parameter ($\alpha = 0.01$). Bin height corresponds to the 50% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10% and 90% percentiles respectively.

Figure 5.42: IC Fault Model optimisation with PBILh - Influence of the size of initial generation on the estimate of posterior marginal probability distribution for the bad quality sand permeability parameter ($\alpha = 0.1$). Bin height corresponds to the 50% percentile of the distribution of posterior estimates over the 50 separate runs of the algorithm. Lower and upper error bars are plotted for 10% and 90% percentiles respectively.
5.4 Summary

In this chapter we have introduced a new robust parallelisation scheme which can easily be implemented for parallelisation of any evolutionary algorithm, and especially EDAs due to their population based nature. The proposed scheme is based on a so-called Master/Slave parallel setup and is not only able to provide a CPU speed-up to the process, but also effectively manage any possible software or hardware related faults within the optimisation loop. Suggested setup makes it possible to utilize heterogeneous parallel clusters.

We have also considered an improved definition of the misfit function, where we concentrated our attention on the way the measured data are weighted in the process of its calculation. The suggested improvements were aimed predominantly at providing an equally sensitive misfit function response (treatment) to both early and late water breakthrough in the reservoir model.

The IC Fault model was used in this chapter to illustrate an application potential of the histogram-based PBIL for reservoir history-matching optimisation studies. We have studied algorithms ability to perform fast and yet explorative enough search within the parameter space while at the same time being able to accurately approximate the underlying uncertainty in model parameters. The influence of the control parameters for the algorithm such as learning rate and size of the initial population was investigated. Following are the conclusions which could be made after the performed tests:

- When optimising a large size parameter space (Section 5.3.1) algorithm was able to sample a diverse set of history matched models. After around 200 processed generations, estimates of the posterior marginal probability distributions for the IC fault model parameters have evolved into a stable shape, therefore helping PBILh to slowly concentrate its search in the areas of highest likelihood. A low value of the learning parameter $\alpha = 0.01$ was used in this case in order to make sure that the algorithm performs an extensive global
5.4. Summary

search within the solution space. A good proof of the fact that PBILh has in deed performed as a typical global optimisation technique is the 3D solutions space cube presented in Fig. 5.17. The cube is constructed from the three 2D projections of the discretised solution space on the \( h - k_{\text{good}} \), \( h - k_{\text{bad}} \) and \( k_{\text{good}} - k_{\text{bad}} \) parameter planes.

- In Section 5.3.2 it was shown that an increased values of the learning rate parameter \( \alpha \) can positively influence convergence speed of the algorithm and, at times, quality of the sampled model. However high values of \( \alpha \) proved to have a very negative effect on the quality of the posterior marginal distribution estimates of the model parameters. The distribution estimates obtained from PBILh runs with \( \alpha \geq 0.1 \) were unstable and uninformative.

On the other hand, lower values of the learning rate (\( \alpha = 0.01 \)) helped the algorithm to perform a much more thorough sampling within the parameter space slowly learning the topology of the underlying distribution, this of course has also resulted in a much slower convergence rates of the algorithm.

Due to a slow speed of convergence within the common limit of 1000 misfit evaluations, PBILh with \( \alpha = 0.01 \) was not able to produce as many models of acceptable fitness as in the case of PBILh with \( \alpha = 0.01 \).

- Size of the initial population was commonly believed to be one of the most influential control parameters of the Population-Based Incremental Learning algorithm. Tests performed in the Section 5.3.3 showed that, given the implemented asynchronous parallelisation setup, this parameter does not have any influence on the quality of the sampled models. In cases where varying \( \text{SizeFirstGen} \) was used with low values of learning rate parameter \( \alpha \), it did not significantly influence the estimates of posterior marginal probability distributions of model parameters either.

This means that, even with minimal amount of prior information, algorithm is
still able to consistently sample the underlying probability distribution. And in reservoir engineering, and history-matching studies specifically, where accurate prior information is rarely available or is of a very coarse scale, such a property of optimisation algorithm can prove to be very beneficial.
Chapter 6

Algorithm testing: real field application

Two different real-world fields are studied in order to demonstrate the application potential of the histogram-based PBIL for reservoir history-matching optimisation.

Section 6.1 covers PBILh’s application to the history-matching optimisation of an average size oil field with complicated reservoir compartmentalisation due to the presence of a high number of faults as well as vertical flow barriers (induced by partial or continuous shale layers). A total of 24 optimisation parameters are optimised (including aquifer permeability, 17 fault transmissibility and 6 vertical permeability multipliers) in order to history match the oil, water and gas production volumes from 8 producing wells. No prior screening of the selected optimisation parameters is performed.

PBILh’s performance is studied with respect to the value of the learning rate. Specifically, a run with a high value of $\alpha = 0.5$ is performed and its results, from the point of algorithm’s convergence speed, quality of the history match and the ability to accurately estimate the posterior marginal distributions, are analyzed.

The oil field studied in the Section 6.2 represents a large carbonate reservoir, with a distinctive platform structure. While the platform itself is believed to be the main production unit, its flanks are understood to be highly fractured which contributes to the non-matrix permeability of the region. The reservoir is further spatially and temporally subdivided into multiple zones. Given such geological conditions, horizontal and vertical permeability multipliers, as well as vertical transmissibility and pore volume multipliers were selected as optimisation parameters - a total of 23 parameters. A screening Plackett-Burman experimental design is performed in or-
der to identify the heavy-hitters (parameters, whose main effects are dominating the change in the system’s output) and reduce the size of the parameter space being optimised.

Over 340 data points of the static gradient survey tests recorded over the period of over 30 years for 65 producing wells will be matched.

PBILh’s performance and ability to estimate the posterior marginal distributions of the optimisation parameters will be evaluated depending on the values of the learning rate. It will also be compared to that of the standard Genetic Algorithm.

6.1 FIELD 1 case study

Aims of the case study are:

- To apply PBILh to history-matching optimisation of an average size oil field with over 15 years of production history;

- Investigate the efficiency of the algorithm’s sampling in the highly multidimensional parameter space (24 parameters) without prior screening of the heavy-hitters;

- Investigate the consequences of the selection of a high value of the learning rate (0.5) over the lower one (0.2) on the algorithms sampling quality and ability to estimate marginal posterior distributions of optimisation parameters;

- Benchmark obtained production history matched profiles with those obtained in the previous/earlier reservoir HM study. Specifically investigate the influence of the physically-questionable constraints of parameter ranges (which were chosen in the earlier performed history match) on the quality of sampled models and algorithm convergence.
6.1.1 General field description

FIELD 1 is an average size under-saturated oil field that has been in production since early 90s. Structurally it is represented by a highly faulted monoclinal horst with dimensions of about 4 : 1 (EW:NS). Reservoir stratigraphy is Upper and Lower Cretaceous.

Lithologically, the reservoir is represented by a mixture of sandstones and shales. The fault system of the field is quite complicated (both normal and transverse faults) due to the presence of the two separate fault directions, which comes from a multi-stage deformation. This results in a high compartmentalization of the reservoir.

FIELD 1 was developed by 8 producers and one water injector. Production activities started in early 1992 with limited water injection commencing at the end of 1997. All the wells are operated under separator control.

Water broke through in mid 1993 in one of the wells, which encountered the reservoir in a significantly down dip position. The pressure reached bubble point in the middle of 1997.

A successful water management strategy, with overall water-cut not exceeding 25%, was implemented with a low voidage replacement of about 30%. Indications were present of the development of a secondary gas-cap in central area of the reservoir, but stable produced GOR was observed.

Main sources of uncertainties were identified in the previous history-matching studies and included:

- reservoir compartmentalisation by multiple faults, potentially sealing or partially sealing;

- presence of the horizontal flow barriers, represented by partial or continuous shale layers, which were modeled by the reduction of vertical permeability;

- aquifer activity, including both its size and permeability.
6.1.2 Test setup

Misfit function definition

The overall misfit function $MF$ is split into two components (6.1):

- Oil/gas production rate match $MF_{OIL, GAS}$;
- Water production rate match $MF_{WATER}$.

This is done due to the fact, that the field only starts producing water in the later stages of its development. The amount of simulated and historical water produced may vary within large ranges and will require its own specific treatment.

$$MF = MF_{OIL, GAS} + MF_{WATER}$$  \hspace{1cm} (6.1)

Similar to the recommended accuracy chosen for the pressure match (Roggero and Hu (1998)), a 5% error tolerance was chosen for both oil and gas production match.

$$MF_{OIL, GAS} = \sum_{j=1}^{N} \left( \frac{SIM_{OIL, GAS}^{j} - HIST_{OIL, GAS}^{j}}{0.05 \cdot HIST_{OIL, GAS}^{j}} \right)^2$$  \hspace{1cm} (6.2)

For the water match it was decided to scale the difference between historical and simulated water quantities identically for the entire period of production. Historical water production at the beginning of water breakthrough is very small. If we were to use a misfit function definition of the general form from Eq. (5.1), water production mismatch would become the main contributor to the global misfit function. This is something we had to avoid to make sure that we can assess mismatch contributions from all the available production quantities.

$$MF_{WATER} = \sum_{j=1}^{N} \left( \frac{SIM_{WATER}^{j} - HIST_{WATER}^{j}}{10} \right)^2$$  \hspace{1cm} (6.3)
6.1. FIELD 1 case study

Experimental parallel setup

A synchronous version of the Master/Slave experimental parallel setup described in the Section 5.1.2 was used. This meant that apart from the initial generation all sampled and processed generations of individuals were of the same size and only used models sampled in current generation (their fitness) to construct the representative marginal probability models.

Size of the initial generation was set to 33 individuals and for all the following generations - 25 (due to the availability of a 25-cpu parallel cluster).

6.1.3 Results. High vs. low learning rate ($\alpha$)

As noted earlier in this report (Section 4.4.1), learning rate parameter $\alpha$ has a significant effect on the accuracy (global nature) of sampling and convergence speed of the PBIL algorithm in general, and its histogram-based implementation in particular.

Here, FIELD 1 optimisation was performed by running the algorithm with both low ($\alpha = 0.2$) and high ($\alpha = 0.5$) values of learning rate. Remaining settings of the algorithm are provided in Table 6.1.

The interest in conducting such a comparison is to see how high we can push the value of $\alpha$, while still preserving a good level of sampling (generation of a diverse range of low misfit models) and descriptive enough statistics for the optimisation parameters.

Summary of the forward model setup is given in Table 6.2.

Note on the algorithm’s convergence

As expected higher learning rate of 0.5 results in a much quicker convergence than that of 0.2 (Fig. 6.2-6.3).

After around 150 completed model evaluations, the algorithm with a higher $\alpha$ has narrowed down its search area to the cluster of a better performing models. While similar cluster can also be detected in the case of a smaller learning rate, it is
Table 6.1: PBILh algorithm setup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Low $\alpha$ (Run1)</th>
<th>High $\alpha$ (Run2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>0.2</td>
<td>0.5</td>
</tr>
<tr>
<td>Histogram discretisation, bins</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Size of initial generation</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td>Size of generation</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Total size of ensemble</td>
<td>307</td>
<td>307</td>
</tr>
</tbody>
</table>

Table 6.2: Forward model formulation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$\alpha = 0.2$ (Run1)</th>
<th>$\alpha = 0.5$ (Run2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parameters</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>Aquifer permeability, mD</td>
<td>100...300</td>
<td>100...300</td>
</tr>
<tr>
<td>FLTMLT-1...17</td>
<td>0...1</td>
<td>0...1</td>
</tr>
<tr>
<td>PERMZ-multiplier1...6</td>
<td>0.0001...1</td>
<td>0.0001...1</td>
</tr>
<tr>
<td>$C_{scale}$</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 6.1: Comparison of the cumulative distribution of model misfit obtained for different values of the learning rate parameter $\alpha$
clear that in this case the algorithm is still sampling the under-performing areas of the parameter space ($MF > 1200$).

It can be seen in Fig. 6.4 and Fig. 6.5, where generation-wise misfit function evolution is plotted, that PBILh in *Run2* starts converging after the 4th generation, while for *Run1* this only happens after the 6th processed generation, and even then it is hard to judge if the algorithm has reached a steady state of convergence. One can also observe a much faster decrease in diversity of the quality of sampled models (within single generation) towards the final stages of the search for the case of *Run2*.

Cumulative distributions of the sampled models’ misfit values (Fig. 6.1) clearly show that PBILh in the case of *Run2* is able to sample a larger group of fit models (concentrate on sampling the narrow neighbourhood of potential optima location earlier on in the search) than that in the case of *Run1*. However the overall fittest model found in both runs is identical.

Therefore, if the aim of the history-matching optimisation is to achieve a quick convergence to a “fit enough” model, use of the high values of $\alpha$ (even as high as 0.5) can be justified.

**Note on the algorithm’s ability to estimate posterior marginal distributions**

What is of a bigger interest to us, within the framework of the current thesis, is the ability of the algorithm to provide accurate estimates of the final posterior marginal distributions of optimisation parameters. Within the limit of the same number of model evaluations, PBILh with a higher learning rate (*Run2*) produces posterior marginal pdf estimates that are more noisy (spiky) than those resulting from the run with a lower $\alpha$ (Fig. 6.6-6.8).

Following observations can be made:

- In both cases, posterior distribution estimates of a vertical permeability multiplier ($PERMZ – multiplier1$) have clearly identified the location of a single optima for this parameter. In reservoir engineering terms it points to the fact
Figure 6.2: Evolution of the objective function \((\alpha = 0.2)\)

Figure 6.3: Evolution of the objective function \((\alpha = 0.5)\)

Figure 6.4: Mean objective function of a single generation \((\alpha = 0.2)\). Figure legend: lower and upper error bars correspond to Q10 and Q90 quantiles respectively and the red line represents the evolution of an average misfit in generations

Figure 6.5: Mean objective function of a single generation \((\alpha = 0.5)\). Figure legend: lower and upper error bars correspond to Q10 and Q90 quantiles respectively and the red line represents the evolution of an average misfit in generations
Figure 6.6: Final posterior marginal distribution estimate ($\alpha = 0.2$)

Figure 6.7: Final posterior cumulative distribution estimate ($\alpha = 0.2$)
6. Algorithm testing: real field application

Figure 6.8: Final posterior marginal distribution estimate ($\alpha = 0.5$)

Figure 6.9: Final posterior cumulative distribution estimate ($\alpha = 0.5$)
that there is unlikely to be any flow in the vertical direction in the corresponding region of the model, confirming the likely location of the shale layer;

- For a group of optimisation parameters (i.e. $FLTMLT - 5$, $FLTMLT - 9$, $FLTMLT - 1$, $FLTMLT - 16$, $FLTMLT - 17$, $PERMZ - multiplier6$ and $PERMZ - multiplier6$) most of the identified high likelihood/fitness areas are the same, however in cases with a higher learning rate of 0.5 we can see that PBILh will neglect the under-performing ranges much faster. This in turn can result in the existence of the high likelihood parameter ranges (bins) being surrounded by zero-likelihood ones. This is best illustrated in the cases of $FLTMLT - 3$, $FLTMLT - 12$ and $PERMZ - multiplier5$ parameters;

- Interesting behaviour of the posterior distribution estimate can be observed for the aquifer permeability $Aquifer - perm$ parameter. The $pdf$ obtained in case of a higher $\alpha$ not only considerably differs from the one obtained with a lower learning rate value, but is also statistically uninformative and can not be used to accurately assess the uncertainty associated with this parameter.

**Note on the quality of history-matched models**

By imposing a cut-off level of 500 for misfit function values, a larger group of “fit” models is obtained in the case of PBILh with 0.5 learning rate, than that of PBILh with 0.2 learning rate (see Fig. 6.1). Fig. 6.10-6.11 and Fig. 6.12-6.13 illustrate achieved oil, gas and water matches for Well-1 and Well-7 respectively. In both figures the entire ensemble of sampled models (figures $a$) and the selected group of fit models (figures $b$) with $MF < 500$ are plotted.

The spread in simulated production quantities of the history-matched models covers the trend achieved in the earlier history matched benchmark case. However notably most of the sampled models demonstrate overproduction of water at the expense of oil and gas production. After the $MF = 500$ cut-off limit is applied, for the case of Well-1 (Fig. 6.10-6.11, both $b$), the range of simulated production quanti-
Figure 6.10: Matching the oil, water and gas production rates of the Well-1 ($\alpha = 0.2$). Figure legend: production history (red), previously history matched model (green) and PBILh sampled models (blue).

Figure 6.11: Matching the oil, water and gas production rates of the Well-1 ($\alpha = 0.5$). Figure legend: production history (red), previously history matched model (green) and PBILh sampled models (blue).
6.1. FIELD 1 case study

**Figure 6.12**: Matching the oil, water and gas production rates of the Well-1 ($\alpha = 0.2$). Figure legend: production history (red), previously history matched model (green) and PBILh sampled models (blue)

**Figure 6.13**: Matching the oil, water and gas production rates of the Well-1 ($\alpha = 0.5$). Figure legend: production history (red), previously history matched model (green) and PBILh sampled models (blue)
ties is narrowed down to the close neighbourhood of the benchmark case, with only a much slower water breakthrough time simulated in all models compared to the reference case.

Well-7 production performance simulated through PBILh sampling is also comparable to if not better (for some of the sampled models) than that of the reference case.

6.1.4 Results. Realistic vs. unrealistic parameter ranges

FIELD 1 model was previously history matched and this single history matched model was provided as the benchmark to be used in current study.

However it is important to point out that some of the parameter ranges were extended beyond physical values in order to achieve the desired order of the mismatch. It mostly concerns a few of the fault transmissibility multiplier values (Ta-

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Parameter value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLTMLT-1</td>
<td>0.000476808333549947</td>
<td></td>
</tr>
<tr>
<td>FLTMLT-2</td>
<td>0.0115</td>
<td></td>
</tr>
<tr>
<td>FLTMLT-3</td>
<td>6.0</td>
<td>!</td>
</tr>
<tr>
<td>FLTMLT-4</td>
<td>14.0</td>
<td>!</td>
</tr>
<tr>
<td>FLTMLT-5</td>
<td>0.004</td>
<td></td>
</tr>
<tr>
<td>FLTMLT-6</td>
<td>0.006</td>
<td></td>
</tr>
<tr>
<td>FLTMLT-7</td>
<td>2.0</td>
<td>!</td>
</tr>
<tr>
<td>FLTMLT-8</td>
<td>0.38639038809987</td>
<td></td>
</tr>
<tr>
<td>FLTMLT-9</td>
<td>0.00701601664772104</td>
<td></td>
</tr>
<tr>
<td>FLTMLT-10</td>
<td>0.269513049967073</td>
<td></td>
</tr>
<tr>
<td>FLTMLT-11</td>
<td>20.0</td>
<td>!</td>
</tr>
<tr>
<td>FLTMLT-12</td>
<td>6.79574156927935</td>
<td>!</td>
</tr>
<tr>
<td>FLTMLT-13</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>FLTMLT-14</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>FLTMLT-15</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>FLTMLT-16</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>FLTMLT-17</td>
<td>0.001</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.3: Fault transmissibility multipliers in previously history matched model
To be consistent with the approach and justification made for such parameter setting it was decided to perform a test, the aim of which would be to see if we can improve the performance of PBILh and the overall quality of the sampled models by extending our ranges beyond their physically realistic boundaries, as it was done in the case of earlier history-matching study.

**Note on the algorithm’s convergence**

General dynamics of the parameter space sampling and misfit function minimisation were observed to be similar in both cases. However, the model which was constructed using the unrealistic parameter ranges (Run1) did produce fitter individuals than those obtained in the case of Run2 (Fig. 6.14). Fig. 6.17 illustrates the generation-by-generation progress of the algorithm ran with realistic parameter ranges (Run1). Here, fitness of the best performing model in each separate generation improves significantly after already the 3rd processed generation. On the other hand, PBILh run with unrealistic parameter ranges is slowly improving the quality of its best matched models in each generation, achieving a few of the best models (over the entire population) in its 10th and 11th generation (Fig. 6.18).

**Note on the algorithm’s ability to estimate posterior marginal distributions**

Since ranges of some of the optimisation parameters were extended in Run2, it was expected that the parameter space being sampled will increase in size. This could have had a profound effect on the estimates of the posterior distributions of model parameters.

When analyzing the posterior distributions in both runs we can subdivide their main observed characteristics into three groups:

- parameters whose posterior pdf’s were close to identical (for example $PERMZ - multiplier1$, $PERMZ - multiplier6$, $FLTMLT - 9$ and $FLTMLT - 12$);
### Table 6.4: PBILh algorithm setup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Realistic ranges (Run1)</th>
<th>Unrealistic ranges (Run2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Histogram discretization, bins</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Size of initial generation</td>
<td>33</td>
<td>32</td>
</tr>
<tr>
<td>Size of generation</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Total size of ensemble</td>
<td>307</td>
<td>307</td>
</tr>
</tbody>
</table>

### Table 6.5: Forward model formulation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Realistic ranges (Run1)</th>
<th>Unrealistic ranges (Run2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parameters</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>k-aq-1, mD</td>
<td>100 ... 300</td>
<td>100 ... 300</td>
</tr>
<tr>
<td>FLTMLT-3</td>
<td>0 ... 1</td>
<td>0 ... 10</td>
</tr>
<tr>
<td>FLTMLT-4</td>
<td>0 ... 1</td>
<td>0 ... 20</td>
</tr>
<tr>
<td>FLTMLT-7</td>
<td>0 ... 1</td>
<td>0 ... 5</td>
</tr>
<tr>
<td>FLTMLT-11</td>
<td>0 ... 1</td>
<td>0 ... 30</td>
</tr>
<tr>
<td>FLTMLT-12</td>
<td>0 ... 1</td>
<td>0 ... 10</td>
</tr>
<tr>
<td>FLTMLT-other</td>
<td>0 ... 1</td>
<td>0 ... 1</td>
</tr>
<tr>
<td>$C_{scale}$</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

**Figure 6.14:** Comparison of the cumulative distribution of model misfit for physically realistic and unrealistic parameter ranges
6.1. FIELD 1 case study

Figure 6.15: Evolution of the objective function (real. ranges)

Figure 6.16: Evolution of the objective function (unreal. ranges)

Figure 6.17: Mean objective function of a single generation (real. ranges). Figure legend: lower and upper error bars correspond to Q10 and Q90 quantiles respectively and the red line represents the evolution of an average misfit in generations.

Figure 6.18: Mean objective function of a single generation (unreal. ranges). Figure legend: lower and upper error bars correspond to Q10 and Q90 quantiles respectively and the red line represents the evolution of an average misfit in generations.
• for such parameters as \( FLTMLT - 12 \) and \( PERMZ - multiplier4 \) posterior distribution estimates were not only reproduced, but a certain identified optima locations received a much higher likelihood in Run2 that those in the Run1;

• posterior distribution estimates of most of the parameters, whose ranges were modified in Run2, have identified the locations of the highest likelihood to be beyond the physically realistic ranges (\( FLTMLT - 3, FLTMLT - 2 \) and \( FLTMLT - 7 \)). This can partly explain why, in case of the Run1, the algorithm was not able to sample models below the 400 misfit threshold.

**Note on the quality of history-matched models**

The quality and number of the history-matched models is higher in case of Run2. By performing a cutoff of a misfit function at a value of 500 (Fig. 6.14) we obtain a much wider spread of models for this run.

For Well-1, in cases of both realistic and unrealistic parameter ranges, PBILh was able to produce the spread of models which covered the reference history-matched model production data. No clear improvement in matching the water breakthrough time was observed for the case of Run2.

Well-7 on the other hand shows that while the spread of the models (in terms of oil and gas production values) was much wider for the case of realistic parameter ranges (Fig. 6.12, a), it was somewhat narrowed down when parameter ranges were expanded (Fig. 6.26, a).
6.1. FIELD 1 case study

Figure 6.19: Final posterior marginal distribution estimate (real. ranges)

Figure 6.20: Final posterior cumulative distribution estimate (real. ranges)
6. Algorithm testing: real field application

| Aquifer-perm | FLTMLT-1 | FLTMLT-2 | FLTMLT-3 | FLTMLT-4 | FLTMLT-5 | FLTMLT-6 | FLTMLT-7 | FLTMLT-8 | FLTMLT-9 | FLTMLT-10 | FLTMLT-11 | FLTMLT-12 | FLTMLT-13 | FLTMLT-14 | FLTMLT-15 | FLTMLT-16 | FLTMLT-17 |
|-------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 0           | 0        | 0        | 0        | 0        | 0        | 0        | 0        | 0        | 0        | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         |
| 0.5         | 0.05     | 0.1      | 0.15     | 0.2      | 0.2      | 0.2      | 0.2      | 0.2      | 0.2      | 0.2       | 0.2       | 0.2       | 0.2       | 0.2       | 0.2       | 0.2       | 0.2       |
| 1           | 0.05     | 0.1      | 0.15     | 0.2      | 0.2      | 0.2      | 0.2      | 0.2      | 0.2      | 0.2       | 0.2       | 0.2       | 0.2       | 0.2       | 0.2       | 0.2       | 0.2       |

**Figure 6.21:** Final posterior marginal distribution estimate (unreal. ranges)

<table>
<thead>
<tr>
<th>Aquifer-perm</th>
<th>FLTMLT-1</th>
<th>FLTMLT-2</th>
<th>FLTMLT-3</th>
<th>FLTMLT-4</th>
<th>FLTMLT-5</th>
<th>FLTMLT-6</th>
<th>FLTMLT-7</th>
<th>FLTMLT-8</th>
<th>FLTMLT-9</th>
<th>FLTMLT-10</th>
<th>FLTMLT-11</th>
<th>FLTMLT-12</th>
<th>FLTMLT-13</th>
<th>FLTMLT-14</th>
<th>FLTMLT-15</th>
<th>FLTMLT-16</th>
<th>FLTMLT-17</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.05</td>
<td>0.1</td>
<td>0.15</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>0.05</td>
<td>0.1</td>
<td>0.15</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
</tr>
</tbody>
</table>

**Figure 6.22:** Final posterior cumulative distribution estimate (unreal. ranges)
6.1. FIELD 1 case study

Figure 6.23: Matching the oil, water and gas production rates of the Well-1 (real. ranges). Figure legend: production history (red), previously history matched model (green) and PBILh sampled models (blue)

Figure 6.24: Matching the oil, water and gas production rates of the Well-1 (unreal. ranges). Figure legend: production history (red), previously history matched model (green) and PBILh sampled models (blue)
6. Algorithm testing: real field application

Figure 6.25: Matching the oil, water and gas production rates of the Well-1 (real ranges). Figure legend: production history (red), previously history matched model (green) and PBILh sampled models (blue).

Figure 6.26: Matching the oil, water and gas production rates of the Well-1 (unreal ranges). Figure legend: production history (red), previously history matched model (green) and PBILh sampled models (blue).
6.2 FIELD 2 case study

Aims of this study were:

- Test the efficiency of PBILh for history-matching optimisation of a large reservoir, which is geologically complicated and has an extensive amount of historical data to be matched;

- Apply the Experimental Design methodology in order to reduce the number of optimisation parameters prior to the optimisation algorithm run, therefore trying to simplify the inverse problem being solved by reducing the size of parameter space;

- Compare convergence characteristics of histogram-based PBIL to those of standard GA;

- Investigate the ability of PBILh to accurately estimate the posterior marginal distributions of model parameters.

6.2.1 General field description

FIELD 2 is represented by a large carbonate reservoir, within which two main regions can be identified - platform and the flanks (slopes). Reservoir is of Devonian and Carboniferous age without an active aquifer.

Within the grain dominated platform, matrix permeability in mostly controlled by the inter-granular porosity; for the low energy (and low porosity) bound stone deposits in the flank region - fractures are the main contributors of the non-matrix permeability. The platform itself is spatially and temporally subdivided into multiple zones relative to its stratigraphic descriptions. We differentiate four zones, which are (in the order of deposition, top to bottom): Zone-1, Zone-2, Zone-3 and Zone-4.
The field has been in production for around 30 years. Static gradient survey (SGS) data were used for history-matching. Over 340 data points of SGS data are available for a total of 65 producing wells.

Due to the complex nature of the reservoir, such properties as horizontal and vertical permeability multipliers, as well as horizontal and vertical transmissibility multipliers for both platform and flank regions were identified as the most influential in the history-matching process. Pore volume multipliers were recommended to be applied if needed.

### 6.2.2 Test setup

#### Misfit function definition

A general form of the sum of squares rule in Eq. 5.1 commonly used for misfit function evaluation in reservoir history-matching was adopted.

#### Experimental parallel setup

A synchronous version of the Master/Slave experimental parallel setup described in the Section 5.1.2 was also used in this application example.

Sizes of both initial and all the subsequent generations were set to 20 individuals.

#### Experimental design setup

A total of 23 parameters, which included horizontal and vertical permeability and transmissibility multipliers as well as pore volume multiplier, for both platform and flank regions of the reservoir were chosen.

It was suggested to first carry out a sensitivity study of the selected 23 parameters. An experimental design methodology (see Section 2.5) was used in order to screen out the heavy-hitters (parameters, whose main effects are dominating the change in the system’s output).
A folded Plackett-Burman (PB) design was chosen as an adequate screening design. The efficiency of the PB designs is that they ignore the interactions between the factors (variables), therefore reducing the number of experiments required for a full factorial design. A folded PB design provides more accurate estimates of the main effects of a large number of factors (optimisation variables). This is ideal in our case, where the aim of the study was to estimate and screen out the main effects, therefore reducing the number of reservoir parameters taking part in history-matching optimisation process.

PB is a two level fractional factorial design - low (minimum) and high (maxi-
For our case of 23-factorial design, a Plackett-Burman matrix will be formed as shown in Fig. 6.27 in rows 1 through to 24. The last run #24 is formed by all the parameters taking their minimum values. A folded PB design is constructed by appending to the original PB design its inverted mirror image (Fig. 6.27, rows 25 through to 48), such that where there was a high value of the specific factor in an experiment, reflection of that experiment will adopt the low value of the parameter in question. This is best seen in the case of run #48, where all the factors take their maximum values, which is inverse to the setup of experiment #24.

After evaluating all 48 predesigned experiments the main effects of the parameters involved were calculated and the pareto chart displaying the statistics is shown in Fig. 6.28. Length of the bars in the chart corresponds to the scale of the influence a particular factor has on the system’s output. Three clear heavy-hitters can easily be seen, those being \(zt_{\text{mult}}_1\), \(kh_{\text{mult}}_7\) and \(kh_{\text{mult}}_4\). All three parameters correspond to the outer platform regions. The following five parameters correspond to the Zone-4 pore volume \(pv_{\text{mult}}_1\) and vertical permeability \(kv_{\text{mult}}_1\) multipliers, outer platform vertical transmissibility multiplier \(zt_{\text{mult}}_2\) in Zone-2, platform...
horizontal permeability multiplier \( kh_{mult\_3} \) in Zone-3, platform and horizontal permeability multiplier \( kh_{mult\_1} \) in Zone-1, as well as flank horizontal permeability multiplier \( kh_{mult\_5} \) in Zone-1 and Zone-3.

Outer platform Zone-3 horizontal permeability \( kh_{mult\_6} \) and platform Zone-2 horizontal permeability \( kh_{mult\_2} \) multipliers, despite their relatively low main effect estimates, were included in the final set of optimisation variables. These parameters were kept for the HM process due to the fact that they correspond to the platform region, which is believed to be responsible for most of the reservoir production volumes.

So based on the performed sensitivity tests, a total of 11 parameters were chosen to carry out a history-matching study.

### 6.2.3 Results. Histogram-based PBIL vs. standard GA

Following are the setup parameters of two variants of the histogram-based PBIL:

- PBILh was run with two different learning rate values – 0.1 and 0.2;

- In both cases initial as well as each subsequent population consisted of 20 individuals (models). Initial population was sampled at random, uniformly within the given bounds for optimisation variables. Algorithm was allowed to run for 20 generations, sampling a total of 400 individuals (models) for each case;

- For fitness scaling, misfit function scaling coefficient was set to \( C_{\text{scale}} = 10^8 \).

The performance of PBILh and quality of the sampled history-matched models were compared to those produced by a run of a standard genetic algorithm with such settings:

- A standard binary coded genetic algorithm was used for the comparison purposes;

- Mutation and crossover probabilities were set to 0.1% and 90% respectively;
• Crossover type - uniform crossover;

• Selection scheme - roulette wheel selection;

• Scaling scheme - truncated sigma scaling (similar to sigma scaling described in Section 4.4.6);

• Size of the initial generation (generated at random) was 100 individuals and of all subsequent generations - 20 (replacement ratio of 20%).

Note on the algorithm’s convergence

The convergence behaviour (speed) of PBILh and its ability to sample models of high fitness were compared to those of standard GA.

Fig. 6.30 displays the comparison between the misfit function cumulative distributions for sGA, PBIL with 0.1 and 0.2 learning rates. Within the limit of 400 model evaluations sGA is able to produce models of a better fitness, with around 5% of the entire population of individuals being fitter than those sampled by PBILh.

If we were to compare just the two runs of the studied algorithm, as the case with FIELD 1 (Section 6.1) has showed, the quality of the overall fittest model(s) was not significantly influenced by the change in learning rate, however the run with the larger value of \( \alpha \) was able to produce/sample a larger ensemble of “fit” models. For example, around 20% of all sampled models for the case of PBILh(\( \alpha = 0.1 \)) had misfit values lower than \( 1.05 \times 10^7 \), compared to around 40% for PBILh(\( \alpha = 0.2 \)) and just under 50% for standard GA.

Similar to the tests presented for the case of Matlab’s PEAKS function (Section 4.3, Fig. 4.18) we decided to redraw the misfit’s cumulative distribution statistics for only the first half of the processed generation. This was done in order to understand the speed (nature) with which both versions of PBILh as well as sGA explored the parameter space at the early stages of optimisation.

As Fig. 6.29 shows, PBILh with \( \alpha = 0.2 \) samples a larger group of fitter models
(10^8 < MF < 1.1 \times 10^8) than those sampled by PBILh with a lower learning rate and sGA. However, genetic algorithm is still able to sample a small number of models with a smaller misfit (MF < 1.8 \times 10^7).

The percentage of under-performing models sampled by both sGA and PBILh (\alpha = 0.2) is almost identical, which leads us to conclude that the reason why the researched algorithm is not able to sample as fit models as those sampled by GA, is that instead of concentrating on the very narrow neighbourhood of the identified (previously sampled) potential optima areas within the parameter space, it continues sampling a wider region and possibly a number of such regions simultaneously. Also, the selection/replacement scheme implemented in GA means that information from the fittest individuals found “up to date” would be explicitly used to drive the sampling and search in new generations, whereas in PBILh only information from the current generation is explicitly used to construct a probability model representative of that generation. Information of the likelihood of certain values of parameters generating fit individuals in the previous generations is accounted for implicitly by using the prior probability estimates coming from the previous gener-
Note on the algorithm’s ability to estimate posterior marginal distributions

Let us compare the generated/final fitness-based (in case of PBILh) and frequency-based (in case of GA) histograms (estimates of posterior marginal pdfs). One may see that PBILh is able to capture similar solution space topology as does GA (Fig. 6.31), however there are a few differences.

As an example we consider six of the selected 11 optimisation parameters. Marginal probability estimates matched very well for parameters which were characterised by a presence of clear single optimum region, such as in the case with $kh_{mult.4}$, and $zt_{mult.1}$. While demonstrating a near uniform behaviour, the posterior estimates of horizontal permeability multipliers $kh_{mult.1}$, $kh_{mult.3}$ and $kh_{mult.6}$ roughly identify ranges of increased likelihood, that correspond to those indicated by the frequency histograms of the GA run.

Scale of the pore volume multiplier variable $pv_{mult.1}$ was normalised in Fig. 6.31 for presentation purposes, however each of the bins here corresponds to a separate map of pore volume multipliers (applied to the entire reservoir). These 10 maps were previously generated using a range of geostatistical techniques.

Here both algorithms were able to sample a clear multi-modal posterior distribution of this discrete variable, with only one slight difference. While both PBILh and GA have identified realisations (bins) #3,#4,#5,#7 and #8 to be of the higher likelihood of producing good history matches, GA has also identified a single realisation (bin) #1. Though PBILh does estimate a high likelihood values for this bin at the early stages in optimisation process (Fig. 6.48), through the course of optimisation algorithm recognizes the other two potential optima locations to be of a bigger importance (higher fitness).

Similar to the nature of PBILh misfit function evolution behaviour, one can see that for most of the optimisation parameters, the shape of their marginal posterior distribution estimates started to stabilize after around 14-15 processed realisations.
Figure 6.31: Comparison of the frequency-based (GA) and fitness-based (PBILh) final posterior marginal distribution estimates

(Fig. 6.33-6.53). This in turn explains the specifics of sampling in the neighbourhood of all identified potential optima regions at the later stages of parameter space exploration. While GA, on the other hand gradually concentrates its search on the fittest individual found so far.

Note on the quality of history-matched models

In order to illustrate the quality of the produced history matched models, we have randomly chosen 15 out of 65 wells whose static gradient survey data we have matched. Ten fittest models out of the pull of 400 sampled models were selected and their corresponding pressure data are plotted in Fig. 6.54.
The selected history-matched model ensemble, for most of the presented wells has demonstrated considerably narrow uncertainty ranges in pressure estimates. As a rare exceptions Well 7, Well 6, Well 10, Well 13 and most of all Well 8 can be considered. These are the wells that proved to be very sensitive to the changes in the optimisation parameters.

However, another important group of wells are those that apart from having a very narrow spread also had a slightly worse match quality overall. These wells we consider to be relatively insensitive to the changes in global parameters used in the current study, and believe that they would be ideal candidates for possible implementation of additional local changes in further history-matching studies.
6.2. FIELD 2 case study

Figure 6.32: Evolution of the posterior marginal distribution estimate for horizontal permeability multiplier $kh$-mult-1. Scale normalised

Figure 6.33: Evolution of the posterior marginal cumulative distribution estimate for horizontal permeability multiplier $kh$-mult-1. Scale normalised
Figure 6.34: Evolution of the posterior marginal distribution estimate for horizontal permeability multiplier $kh$-mult-2. Scale normalised

Figure 6.35: Evolution of the posterior marginal cumulative distribution estimate for horizontal permeability multiplier $kh$-mult-2. Scale normalised
Figure 6.36: Evolution of the posterior marginal distribution estimate for horizontal permeability multiplier kh-mult-3. Scale normalised

Figure 6.37: Evolution of the posterior marginal cumulative distribution estimate for horizontal permeability multiplier kh-mult-3. Scale normalised
Figure 6.38: Evolution of the posterior marginal distribution estimate for horizontal permeability multiplier $kh$-mult-4. Scale normalised

Figure 6.39: Evolution of the posterior marginal cumulative distribution estimate for horizontal permeability multiplier $kh$-mult-4. Scale normalised
6.2. FIELD 2 case study

Figure 6.40: Evolution of the posterior marginal distribution estimate for horizontal permeability multiplier kh-mult-5. Scale normalised

Figure 6.41: Evolution of the posterior marginal cumulative distribution estimate for horizontal permeability multiplier kh-mult-5. Scale normalised
6. Algorithm testing: real field application

Figure 6.42: Evolution of the posterior marginal distribution estimate for horizontal permeability multiplier \( kh \)-mult-6. Scale normalised

Figure 6.43: Evolution of the posterior marginal cumulative distribution estimate for horizontal permeability multiplier \( kh \)-mult-6. Scale normalised
Figure 6.44: Evolution of the posterior marginal distribution estimate for vertical transmissibility multiplier $z_{t-1}$. Scale normalised

Figure 6.45: Evolution of the posterior marginal cumulative distribution estimate for vertical transmissibility multiplier $z_{t-1}$. Scale normalised
Figure 6.46: Evolution of the posterior marginal distribution estimate for vertical transmissibility multiplier $z_{t-2}$. Scale normalised.

Figure 6.47: Evolution of the posterior marginal cumulative distribution estimate for vertical transmissibility multiplier $z_{t-2}$. Scale normalised.
Figure 6.48: Evolution of the posterior marginal distribution estimate for pore volume multiplier pv-1. Scale normalised

Figure 6.49: Evolution of the posterior marginal cumulative distribution estimate for pore volume multiplier pv-1. Scale normalised
Figure 6.50: Evolution of the posterior marginal distribution estimate for horizontal permeability multiplier $kh$-mult-7. Scale normalised

Figure 6.51: Evolution of the posterior marginal cumulative distribution estimate for horizontal permeability multiplier $kh$-mult-7. Scale normalised
Figure 6.52: Evolution of the posterior marginal distribution estimate for horizontal permeability multiplier $k_z$-mult-1. Scale normalised

Figure 6.53: Evolution of the posterior marginal cumulative distribution estimate for horizontal permeability multiplier $k_z$-mult-1. Scale normalised
Figure 6.54: SGS pressure matches of the 10 best history-matched models for selected wells. Here black data points correspond to historically recorded SGS data and red lines correspond to the simulated static pressures.
6.3 Summary

The overall observation after the performed tests, in terms of algorithm performance, is that while providing a much faster convergence rate, higher values of $\alpha$ jeopardize the ability of the algorithm to accurately estimate and evolve statistically informative posterior distributions for those optimisation parameters which are characterised by a presence of multiple optima.

If the aim of the history-matching study is to sample as fit (as closely matching) models as possible as quickly as possible, without putting a lot of emphasis on the informative nature of the posterior marginal distribution estimates, then as high values of learning rate as $\alpha = 0.5$ can be perfectly justified. However if uncertainty in the input parameters is recognised and considered to be important to estimate (due to the conditions of the available prior data or the lack of it for that matter), lower values of $\alpha$ would do a better job assessing this task.

Field 1 history-matching optimisation example have shown us that a blind selection of a high number of optimisation parameters without the performance of additional screening tests may result in a very noisy distribution estimates. This suggestion was backed up by the results of the second history-matching study of the Field 2. Here, with the aid of folded Placket-Burman type experimental design the most influential model parameters were identified and only they took part in the optimisation process. Notable almost for all of them PBILh was able to provide informative uncertainty estimates, which in some cases were able to clearly identify existence of multiple optima regions.

It was also shown, on the example of 10 best history-matched models in the case of Field 1, that changes/optimisation based on the variations of only global reservoir properties may prove inefficient and have little effect on the simulated production performance of some of the wells. Further local changes may need to be applied.
When modelling of oil and gas reservoirs is concerned, one normally considers a wide range of reservoir model parameters, some of which may be dependent on others. Examples include correlation between porosity and permeability values, dependency between the structural model parameters such as, for instance, fault relay ramp geometry (where trade off between both the slope and the extent of the ramp itself can affect the fluid flow across the fault) and its transmissibility.

A reservoir engineer is always aware of the possibility of such interactions being present within the model studied. The logical conclusion then is to try to use this information when performing reservoir simulation and specifically reservoir history-matching related studies.

The core of an efficient HM optimisation technique is its sampling quality. Any extra information capable of guiding the search within the parameter space based on the assumptions of dependence or independence between the optimisation/model parameters should be welcomed into the optimisation process. Currently, apart from the neighbourhood algorithm, none of the stochastic optimisation techniques used in industry, such as simulated annealing, evolutionary algorithms (namely genetic algorithms and evolutionary strategies) explicitly take into account this extra information of parameter interactions during sampling stages, and even in NA this information is not used as a major driving force.

Therefore, the aim of the work presented in the chapter is to make a first step towards giving the engineer an algorithm which is able to sample conditional probability distributions of model parameters and use this information to improve the
quality of solution space exploration.

In Section 7.1 we give a brief overview of the work already done within the domain of EDAs with conditional sampling. Although this class of algorithms have been heavily developed and extended over the past few years, our biggest interest is dedicated to those algorithms capable of handling chain-type pairwise dependencies between model parameters. We will therefore concentrate on their sole description.

In Section 7.2 an introduction is given to the extension of the researched histogram-based Population-Based Incremental Learning algorithm which is based on a pairwise conditional sampling of chain-like dependency structures. We call this extension PBILh-chain.

Preliminary results of the PBILh-chain testing will be presented and discussed in Section 7.3.

7.1 Previous work

Within the framework of Estimation of Distribution Algorithms (EDAs) the idea of conditional sampling is not new. In the binary problem domain MIMIC algorithm introduced by de Bonet et al. (1997) allowed to account for a chain-like dependencies between model parameters, COMIT introduced by Baluja and Davies (1997) was designed to handle tree-like dependencies and BMDA introduced by Pelikan and Mühlenbein (1999) in some ways combined the two previously described types of dependency structures into an advanced forest-like type. Brief description for each of the mentioned techniques can be found in the Chapter 3 (Section 3.2.2).

Much higher level multivariate interactions were a subject of the research carried out by Pelikan et al. (1999). The author later extended his previously developed ideas into the Bayesian Optimisation Algorithm (BOA) and its hierarchical version hBOA in Pelikan (2005). But it is the paper by Bosman and Thierens (1999) that has formed the base line for the research work presented in this chapter. It was mostly
7.1. Previous work

1. Generate a collection of $n$ random vectors 
   \( \{Z^i \mid i \in 0, \ldots, n-1\} \)
2. Evaluate function values of the vectors in the collection 
   \( C(Z^i) (i \in \{0, \ldots, n-1\}) \)
3. $t = 0$
   
   \hspace{1em} \textbf{while} (Termination criteria are not met) \hspace{1em}
   \begin{align*}
   &4. \text{Select } \lfloor \tau n \rfloor \text{ vectors} \\
   &\quad \{Z^{(S)} \mid i \in 0, \ldots, \lfloor \tau n \rfloor - 1\} \leftarrow \text{sel()} \ (\tau \in [1/n, 1]) \\
   &5. \text{Set } \theta_t \text{ to the worst function value among the selected vectors} \\
   &6. \text{Determine the probability distribution } \hat{P}^{\theta_t}(Z) \\
   &7. \text{Generate } m \text{ new vectors (offspring) } O(t) \text{ by sampling from } \hat{P}^{\theta_t}(Z) \\
   &8. \text{Incorporate the new vectors } O(t) \text{ into the collection} \\
   &\quad \text{rep()} \\
   &9. \text{Evaluate the new vectors in the collection } (\subseteq O(t)) \\
   &10. t \leftarrow t + 1
   \end{align*}

\textbf{Figure 7.1: Pseudo-code of the IDEEA framework according to Bosman and Thierens (1999)}

due to the paper’s specific consideration of the sampling of conditional continuous distributions of optimisation parameters.

Bosman and Thierens (1999) introduced a so called Iterated Density Estimation Evolutionary Algorithm (IDEEA) which operates according to the framework presented in Fig. 7.1.

For the case of an iterated optimisation algorithm (which evolutionary algorithms are), given a set of $n$ vectors $Z^i (i \in 0, \ldots, n-1)$ at some iteration $t$, denote the largest function value of the best $\lfloor \tau n \rfloor$ vectors with $\tau \in [1/n, 1]$ by $\theta_t$. Based on the information from the $\lfloor \tau n \rfloor$ selected vectors we approximate the the density $P^{\theta_t}(Z)$. Once the approximation of the distribution is obtained, IDEEA uses it to sample the new generation (offspring) of vectors, which, when evaluated, will hopefully all have function values lower than $\theta_t$. The selection step will then take place again providing the next approximation to the distribution $- P^{\theta_{t+1}}(Z)$ with $\theta_{t+1} \leq \theta_t$. All the above essentially form the basics of the IDEEA framework given in Fig. 7.1.

Step 6 and 7 within the general IDEEA framework are the most important ones, and depending on the variations of these two steps a wide range of algorithm realisations exist. One which is of the biggest interest for us is the IDEEA implementa-
6. Determine the probability distribution $\hat{P}_{\pi,\omega}(Z)$ in two steps

6.1 Find an acyclic PDS $(\pi, \omega)$

6.2 Estimate the density functions

\[ \{ \hat{P}(Z_{\omega_i}|Z_{\pi(\omega_i)_0}, Z_{\pi(\omega_i)_1}, \ldots, Z_{\pi(\omega_i)(\pi(\omega_i)_i-1)} \} \]

7. Generate $m$ new vectors (offspring) $O$ by sampling from $\hat{P}_{\pi,\omega}(Z)$

Figure 7.2: Pseudo-code of the IDEA framework with conditional sampling specifications (modified steps 6 and 7)

...
7.2 Suggested algorithm modification

The amount of information coming from the general principles of PBILh operation (i.e. fitness rather than frequency based nature as well as incremental style of probability model updating) has a greater influence on the operation of the proposed optimisation technique as a whole than that coming from IDEA and BOA principles. We therefore will refer to it as a histogram-based Population-Based Incremental Learning algorithm with pairwise conditional sampling of chain-like dependency structures – PBILh-chain.

We use the IC Fault Model (Appendix C) here to illustrate the main idea behind the PBILh-chain algorithm. The model is characterised by three optimisation parameters: fault throw $h$, good and poor quality sand permeabilities $k_{good}$ and $k_{poor}$.

Three main components of the PBILh-chain algorithm operation are further discussed in this section: representation of joint probabilities by the algorithm, sampling of parameter values from these JPDs and PBILh-inspired evolution of JPDs.

Representation of joint probability distributions

As shown in Chapter 5 (Section 5.3), we suggest constructing a 3D (or ND) parameter space representations which are comprised of the 2D projections of the sampled parameter space onto each of the possible 2D planes (formed by all possible paired combinations of model parameters) in order to visually study the spatial sampling capabilities of the optimisation algorithm. General principles of the construction of such 2D projections were given in Fig. 5.16.

In this thesis we suggest interpreting these 2D fitness-wise projections of the sampled solution space, or to be more accurate – their normalised versions, as a joint
density functions or joint probability distributions (JPD). If we normalize the entire 2D image, the resulting matrix would represent a multivariate joint density function with each cell representing the probability $p(V_i^*, V_j^*)$ of variables $V_i$ and $V_j$ falling within the parameter ranges the cell stands for - $V_i^* \in b_i^*$ and $V_j^* \in b_j^*$. However, since we are interested in chain-type pairwise sampling of the parameter space we can view these 2D images or matrices as a collection of conditional probability distributions.

Fig. 7.3 shows a 3D image of the sampled solution space of the IC Fault Model obtained at the end of a single run of PBILh algorithm with $\alpha = 0.01$ (presented in Chapter 5, Section 5.3.2, Fig. 5.31) overlaid by the estimates of the marginal posterior distributions of model parameters $P(h)$, $P(k_{\text{good}})$ and $P(k_{\text{bad}})$. As the histograms were constructed based on the fitness values of the sampled individuals they correspond very well to the distribution of fitness values observed in the three 2D projections of parameter space. The plotted histograms are the only statistical measures needed when all three parameters are independent of each other (i.e. their PDS is empty and consists of only nodes but no dependency arcs).

Let us assume a predefined PDS of the following form: $h \rightarrow k_{\text{good}} \rightarrow k_{\text{bad}}$. By using the 3D image of the sampled solution space we can derive the conditional distribution of model parameters in the following two steps:

1. Fault throw parameter $h$ was chosen as the head of the chain, therefore its marginal posterior distribution $P(h)$ estimate is preserved unchanged. The good quality sand permeability parameter $k_{\text{good}}$, is dependent on the value of $h$ (according to the chosen PDS) and therefore we suggest viewing the 2D plane $h - k_{\text{good}}$ of the size $(b_h \times b_{k_{\text{good}}})$ as a collection of $b_h$ histograms (with $b_{k_{\text{good}}}$ bins each), which, when normalised, can be interpreted as conditional probability distributions $P(k_{\text{good}}|h)$ (Fig. 7.4);

2. Similar actions can then performed on the $k_{\text{good}} - k_{\text{bad}}$ 2D plane, where the matrix of the size $(b_{k_{\text{good}}} \times b_{k_{\text{bad}}})$ consists of $b_{k_{\text{good}}}$ histograms (with $b_{k_{\text{bad}}}$ bins
7.2. Suggested algorithm modification

each), which, when normalised, can be interpreted as conditional probability distributions \( P(k_{\text{bad}}|k_{\text{good}}) \sim P(k_{\text{bad}}|k_{\text{good}}, h) \) (Fig. 7.5).

Sampling from joint probability distributions

The available unconditional (\( P(h) \)) and joint (\( P(k_{\text{good}}|h) \) and \( P(k_{\text{bad}}|k_{\text{good}}, h) \)) probability distributions information can be used to sample a new candidate solution (individual) \( \text{Ind}^* = (h^*, k_{\text{good}}^*, k_{\text{bad}}^*) \) according to the order specified by a predefined PDS.

1. **Sampling** \( h^* \): Use the marginal probability distribution estimate \( P(h) \) to sample the value of \( h^* \);

2. **Sampling** \( k_{\text{good}}^* \): Given the the sampled value \( h^* \) identify the bin \( b_h^* \) in \( P(h) \) to the range of which the sampled value belongs. Pick out the \( b_h^* \) row of the \( h-k_{\text{good}} \) 2D plane (joint probability distribution matrix) and normalize it (Fig. 7.5). Now it represents the conditional probability distribution estimate \( P(k_{\text{good}}|h^*) \). Use this estimate to sample the \( k_{\text{good}}^* \) value;

3. **Sampling** \( k_{\text{bad}}^* \): Given the the sampled value \( k_{\text{good}}^* \) identify the bin \( b_{k_{\text{good}}^*} \) in \( P(k_{\text{good}}|h^*) \) to the range of which the sampled value belongs. Pick out the \( b_{k_{\text{good}}^*} \) row of the \( k_{\text{good}}-k_{\text{bad}} \) 2D plane and normalize it (Fig. 7.5). Now it represents the conditional probability distribution estimate \( P(k_{\text{bad}}|k_{\text{good}}^*) \) and subsequently \( P(k_{\text{bad}}|k_{\text{good}}^*, h^*) \). Use this estimate to sample the \( k_{\text{bad}}^* \) value (Fig. 7.6).

**PBILh inspired evolution of joint probability distributions**

In PBILh we do not explicitly use population-wise information on the individual’s quality and frequency of resampling (as opposed to IDEEA framework), but instead evolve the probability distribution based on information from the given generation of sampled models. Therefore, for PBILh-chain we need to develop a method of
Figure 7.3: Image of the IC Fault Model solution space overlaid by the estimates of the marginal posterior distributions of model parameters.

Figure 7.4: Sampling strategy of PBILh-chain for a predefined dependency chain structure. Step 1: Given the marginal distribution estimate $P(h)$ of the fault throw parameter, sample a single value $h^*$ and derive the conditional probability distribution $P(k_{good}|h^*)$ by interpreting the information from the underlying $h - k_{good}$ 2D plane.
7.2. Suggested algorithm modification

Figure 7.5: Sampling strategy of PBILh-chain for a predefined dependency chain structure.
Step 2: Given the conditional distribution estimate $P(k_{\text{good}}|h^*)$ of the good quality sand permeability parameter, sample a single value $k_{\text{good}}^*$ and derive the conditional probability distribution $P(k_{\text{bad}}|k_{\text{good}}^*, h^*)$ by interpreting the information from the underlying $k_{\text{good}}-k_{\text{bad}}$ 2D plane.

Figure 7.6: Sampling strategy of PBILh-chain for a predefined dependency chain structure.
Step 3: Given the conditional distribution estimate $P(k_{\text{bad}}|k_{\text{good}}^*, h^*)$ of the poor quality sand permeability parameter, sample a single value $k_{\text{bad}}^*$.
constructing and updating the joint probability distributions in each iteration of the algorithm given some PDS structure.

First let us set out the background notations:

- We optimise based on a predefined PDS structure, where \( j \in (0, 1, \ldots, n) \) are indexes of the chain positions;

- Ranges of the optimisation parameters \( V_i, i \in (0, 1, \ldots, n) \) are discretised into \( b^{V_i} \) bins;

- A single marginal distribution estimate \( H(V^0) \) of the size \( 1 \times b^{V_0} \) is sampled and evolved for the parameter \( V_i^{j=0} \) representing the head of the chain-type structure (chosen PDS). All the elements (bins) in \( H(V^0) \) are then initialised to \( 1/b^{V_0} \) in order to represent a uniform probability distribution in the first generation \( t = 0 \);

\[
H^{t=0}(V^0) = \begin{pmatrix} h(0, 0) & h(0, 1) & \ldots & h(0, b^{V_0}) \end{pmatrix} = \begin{pmatrix} 1/b^{V_0} & 1/b^{V_0} & \ldots & 1/b^{V_0} \end{pmatrix}
\]

- In order to sample values of subsequent chain elements (conditional parameters) \( V_i^{j} \in (1, \ldots, n-1), n-2 \) matrices (2D parameter space projections equivalents) are initialised as follows: matrix \( H(V^0-V^1) \) of the size \( b^{V_0} \times b^{V_1} \), matrix \( H(V^1-V^2) \) of the size \( b^{V_1} \times b^{V_2} \), matrix \( H(V^{n-2}-V^{n-1}) \) of the size \( b^{V_{n-2}} \times b^{V_{n-1}} \). As it was stated above, rows in the \( H(V^j-V^{j+1}) \) matrices represent conditional probability distributions, which, in analogy with \( H(V^0) \), are each initialised to represent a uniform probability distribution. This is done by setting elements of each of the \( j \) rows to \( 1/b^{V_{j+1}} \).
7.2. Suggested algorithm modification

A typical $H(V^j - V^{j+1})$ at generation (iteration) $t = 0$ will then become

$$H^{t=0}(V^j - V^{j+1}) = \begin{pmatrix}
h(0, 0) & h(0, 1) & \ldots & h(0, b^{V^{j+1}}) \\
h(1, 0) & h(1, 1) & \ldots & h(1, b^{V^{j+1}}) \\
\vdots & \vdots & \ddots & \vdots \\
h(b^{V^j}, 0) & \cdots & \cdots & h(b^{V^j}, b^{V^{j+1}})
\end{pmatrix}$$

and together with $H^{t=0}(V^0)$ will be used to sample first generation of candidate solutions at random, by sampling values from the uniform conditional distributions.

The following condition should hold.

$$\sum_{l=1}^{b^{V^{j+1}}} h(d, l) = 1 \quad \forall \ d \in (0, 1, \ldots, b^j)$$

Drawing on the basic principles of probability model updating in PBILh, once a certain number of individuals (models) have been evaluated and a range of misfit (and therefore fitness) values is available, we are ready to estimate the equivalent to the temporary histograms in PBILh, or conditional likelihood estimates.

Initially, we create temporary histogram $H_{\text{temp}}(V^0)$ of the same size as $H(V^0)$ and a range of temporary matrixes $H_{\text{temp}}(V^j - V^{j+1})$ of the same size as $H(V^j - V^{j+1})$ and assign all their elements to 0. Given the total ensemble of evaluated individuals of the size $N_{\text{eval}}$, for each of the evaluated individuals $Ind_k(V_0, V_1, \ldots, V_{n-1})$, $k \in (0, 1, \ldots, N_{\text{eval}} - 1)$ the following procedure is performed:

1. Starting at the head position in the chain $j = 0$ we locate the bin # in histogram $H_{\text{temp}}(V^0)$ within which value of the parameter $V^0$ falls. Let us denote the identified bin within the range $0 \ldots b^{V^0}$ as $b^{V^0*}$. If the current fitness of the bin is lower than the fitness of the individual $FF(Ind_k)$, the bin is assigned the
new fitness value $FF(Ind_k)$;

2. Given the value of $b^{V_0*}$ from the appropriate temporary matrix $H_{temp}(V^0 - V^1)$ pick a row $h(b^{V_0*}, \ldots)$ and, as in the case with $H_{temp}(V^0)$ construction, locate the bin # in “histogram” $h(b^{V_0*}, \ldots)$ within which value of the parameter $V^1$ falls. Update bin’s fitness accordingly.

3. Perform Step.2 for all the remaining parameters $V^j$, $j \in (2, \ldots, n)$ in a chain;

4. When all the parameters of the current individual $Ind_0(V_0, V_1, \ldots, V_{n-1})$ have been processed, repeat Steps 1-3 for the remaining group of evaluated individuals $Ind_k(V_0, V_1, \ldots, V_{n-1})$, $k \in (1, \ldots, N_{eval} - 1)$

5. When the entire generation of $N_{eval}$ individuals have been processed, histogram $H_{temp}(V^0)$ and all the matrixes $H_{temp}(V^j - V^{j+1})$ updated with the recent fitness information, normalize $H_{temp}(V^0)$ and $H_{temp}(V^j - V^{j+1})$, so that the following conditions hold:

\[
\sum_{l=1}^{b^{V_0}} h_{temp}(V^0)(1, l) = 1
\]

\[
\sum_{l=1}^{b^{V_{j+1}}} h_{temp}(V^j - V^{j+1})(d, l) = 1 \quad \forall \ d \in (0, 1, \ldots, b^{V_j})
\]

6. Now, given a value of the learning rate parameter $\alpha$, information on the prior $(H(.))$ and likelihood (normalised $H_{temp}(.)$) probability model estimates, we use the general updating rule of PBILh (Eq. 4.4), in order to update the probability model. This new updated model can then be interpreted as the posterior probability model estimate and used as a prior in the next generation $t + 1$.

\[
H(t + 1)(.) = (1 - \alpha)H(t)(.) + \alpha H_{temp}(t + 1)(.)
\]

In the current section the theoretical background to the algorithm’s conditional sampling strategy and its approach to probability model updating were outlined.
Next section will demonstrate preliminary results of PBILh-chain application to reservoir history-matching optimisation of a synthetic test model.

### 7.3 Testing results

The IC Fault Model is used here to test the performance of PBILh-chain. Model specifications are provided in Table 7.1. This is a relatively small model with three optimisation parameters: fault throw $h$ as well as good and poor quality sand permeabilities $k_{good}$ and $k_{bad}$. Therefore, when considering a chain-like type dependencies between the three variables, a total of $3! = 6$ different chain orderings can be constructed. These are

<table>
<thead>
<tr>
<th>Chain ID</th>
<th>Chain structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>chain-ordering-123</td>
<td>$h \rightarrow k_{good} \rightarrow k_{bad}$</td>
</tr>
<tr>
<td>chain-ordering-132</td>
<td>$h \rightarrow k_{bad} \rightarrow k_{good}$</td>
</tr>
<tr>
<td>chain-ordering-213</td>
<td>$k_{good} \rightarrow h \rightarrow k_{bad}$</td>
</tr>
<tr>
<td>chain-ordering-231</td>
<td>$k_{good} \rightarrow k_{bad} \rightarrow h$</td>
</tr>
<tr>
<td>chain-ordering-312</td>
<td>$k_{bad} \rightarrow h \rightarrow k_{good}$</td>
</tr>
<tr>
<td>chain-ordering-321</td>
<td>$k_{bad} \rightarrow k_{good} \rightarrow h$</td>
</tr>
</tbody>
</table>

As stated in the previous section, probability density structures (Bayesian networks) can be predefined and stay constant throughout the course of optimisation or alternatively evolve based on the information acquired by the algorithm at each generation. In all reported test runs the PDS were maintained constant.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of variables to optimise, $N$</td>
<td>$3$</td>
</tr>
<tr>
<td>Fault throw $h$, ft</td>
<td>$0 \ldots 60$</td>
</tr>
<tr>
<td>Good quality sand permeability $k_{good}$, mD</td>
<td>$0 \ldots 200$</td>
</tr>
<tr>
<td>Poor quality sand permeability $k_{bad}$, mD</td>
<td>$0 \ldots 5$</td>
</tr>
<tr>
<td>Production series to match</td>
<td>Oil production (WOPR) Water production (WWPR) Water injection (WWIR)</td>
</tr>
</tbody>
</table>

**Table 7.1:** IC Fault Model optimisation with PBILh-chain: Forward model setup
Table 7.2: IC Fault Model optimisation with PBILh-chain: Algorithm setup

For each of the 6 dependency structures we performed 3 identical runs of PBILh-chain. This was done in order to not only demonstrate the general applicability of the proposed algorithm extension but also to have a slightly better understanding of its stability and robustness. Algorithm setup details and control parameters are given in Table 7.3.

We use 3 identical runs of the original PBILh setup as a benchmark. Algorithm was run with exactly the same control parameters and parallelization scheme specification as the one specified in Table 7.3. The history-matched production profiles of the top 10% fittest sampled models in the first of the three selected runs of standard PBILh are displayed in Fig. 7.7.

Note on the algorithm’s ability to estimate and evolve joint probability distributions

As an example, let us consider history-matching optimisation of the IC Fault model by PBILh-chain algorithm with the following PDS: \( k_{\text{good}} \rightarrow h \rightarrow k_{\text{bad}} \). In this case the good quality sand permeability parameter will always be sampled first, independently of the remaining two, therefore a single histogram will be maintained and evolved for \( k_{\text{good}} \). Evolution of this histogram over the total of 88 processed generation is presented in Fig. 7.8.

Fault throw parameter \( h \) is sampled second (depending on the sampled value
7.3. Testing results

Figure 7.7: IC Fault Model optimisation with PBILh - 10% of the sampled models are plotted, with misfit value $MF < 621$. A solid red line represents historical data.

Figure 7.8: IC Fault Model optimisation with PBILh_chain – Chain ordering $k_{\text{good}} \rightarrow h \rightarrow k_{\text{bad}}$. Evolution of the single posterior probability distribution estimate $P(k_{\text{good}})$ for good quality sand permeability parameter.
of \( k_{good} \) using the information derived from the joint probability distributions matrix \( H(k_{good} - h) \), evolution of which is presented in Fig. 7.9. Subsequently the poor quality sand permeability parameter \( k_{bad} \) is sampled from the joint probability distributions matrix \( H(h - k_{bad}) \) given the previously sampled values of \( h \). Fig. 7.10 demonstrates an evolution of the later joint probability distributions matrix.

From Fig. 7.8 it can be seen that PBILh-chain very rapidly (within the limit of the first 5 generations) identifies a clear and stable structure of the underlying distribution for the good quality sand permeability parameter \( k_{good} \). Resampling is almost instantly concentrated on the \#14th bin, which represents the range within which the global optimum is located. Likelihood of the bin being re-sampled reaches 1 after about 40 processed generations.

Evolution of the matrices representing the estimates of joint probability distributions of the remaining two model parameters is shown in Fig. 7.9 and Fig. 7.10.

It is important to understand that while we used the 2D images of the sampled parameter space as an analog in order to come up with a structure for these JPDs, the two images (that of 2D projection and a JPD) plotted for the same group of parameters are not equivalent. In a 2D projection image we map the model fitness distribution within the parameter space. Therefore bright red colours (according to the colour scale provided) correspond to the parameter ranges within which models of the highest fit were sampled. When we plot JPD matrixes in the way it was done in Fig. 7.9 and Fig. 7.10, essentially we would like to have a separate colour scale for each of the matrix rows, with each representing a separate conditional probability distribution. But for the ease of visual representation we provide a single colour scale for each of the JPD matrixes. For both \( h \) and \( k_{bad} \) parameters, JPDs gradually evolve throughout the course of optimisation, this process is accompanied by the relative smoothing of the conditional probability distribution estimates. This can be seen in both figures at the later stages of optimisation (# generations > 30), where bright yellow, orange and red colours (representing a much higher likelihood estimates) are gradually replaced by a more subtle blue tones.
7.3. Testing results

**Figure 7.9**: IC Fault Model optimisation with PBILh chain – Chain ordering $k_{good} \rightarrow h \rightarrow k_{bad}$. Evolution of the conditional posterior probability distribution estimate $P(h|k_{good})$ for fault throw parameter.

**Figure 7.10**: IC Fault Model optimisation with PBILh chain – Chain ordering $k_{good} \rightarrow h \rightarrow k_{bad}$. Evolution of the conditional posterior probability distribution estimate $P(k_{bad}|h, k_{good})$ for poor quality sand permeability parameter.
Figure 7.11: IC Fault Model optimisation with PBILh: Chain – Chain ordering $k_{good} \rightarrow h \rightarrow k_{bad}$. Final posterior conditional distribution estimates.
The final posterior joint probability estimates obtained by PBIL\textsubscript{h}-chain after 88 processed generations of PBIL\textsubscript{h}-chain as well as their practical interpretation are presented in Fig. 7.11.

As stated above the posterior distribution estimate for the good quality sand permeability $P(k_{\text{good}})$ clearly and correctly identifies the range $k_{\text{good}}^{*}$ corresponding to the global optimum value of 137.1 mD. Given this the $k_{\text{good}}^{*}$ value, the estimate of a posterior conditional distribution for fault throw parameter $P(h|k_{\text{good}}^{*})$ is selected from the JPD matrix $H(k_{\text{good}} - h)$. Three bins with the highest likelihood correctly correspond to the locations of the optima solutions, including the global optimum (bin #4). For each of the three bins we subsequently select the estimates of posterior conditional distributions for poor quality sand permeability parameter $k_{\text{bad}}^{*}$—$P(k_{\text{bad}}|h^{*}, k_{\text{good}}^{*}), P(k_{\text{bad}}|h^{**}, k_{\text{good}}^{*})$ and $P(k_{\text{bad}}|h^{***}, k_{\text{good}}^{*})$. Conditional distributions estimates $P(k_{\text{bad}}|h^{*}, k_{\text{good}}^{*})$ and $P(k_{\text{bad}}|h^{**}, k_{\text{good}}^{*})$ both indicate the range corresponding to the location of the global optimum ($k_{\text{bad}} = 1.31 \text{ mD}$), however for the case of $P(k_{\text{bad}}|h^{***}, k_{\text{good}}^{*})$, this range is estimated to have a zero likelihood of being resampled.

**Note on the algorithm’s convergence**

In Fig. 7.12 cumulative distributions of the sampled model misfits for each of the considered chain orderings are plotted. Statistics of all performed tests (i.e. $P_{10}$, $P_{50}$ and $P_{90}$ of models’ misfit as well as the lowest misfit obtained in each of the runs) is given in Table 7.3.

By comparing the $P_{10}$, $P_{50}$ and $P_{90}$ misfit values of the sampled models in each of the chain-type PDS considered, we can make the following observations:

- An original implementation of PBIL\textsubscript{h} algorithm with an “empty” PDS structure was able to produce a $P_{10}$ misfit comparable with the other tests. Its $P_{90}$ values are often better than those for other tests;

- The $P_{50}$ misfit estimates were largely similar for all the chain structures apart
Figure 7.12: IC Fault Model optimisation with PBILh-chain – Influence of the parameters ordering in a chain on the quality of sampling. Cumulative distributions of model misfit over 3 separate runs of the algorithm are plotted for each of the considered chain orderings from those with $k_{good}$ being at the head of the dependency chain. For $k_{good} \rightarrow h \rightarrow k_{bad}$ in all three performed test runs of PBILh-chain, the $P_{50}$ was consistently lower than 4000 and for $k_{good} \rightarrow k_{bad} \rightarrow h$ came to 4106.2 which was still one of the lowest values of $P_{50}$ among the range of PDS structures;

- By introducing a chain-like dependency between model parameters, PBILh-chain has increased the spread of the quality of sampled models compared to that produced by PBILh. This can be seen when comparing the $P_{90}$ misfit function statistics. In the case of one of the runs of $k_{good} \rightarrow h \rightarrow k_{bad} \rightarrow h$ PDS structure the maximum $P_{90}$ misfit value sampled was 50154, with 20083 being the maximum sampled by PBILh;

- In terms of the quality of the overall fittest sampled models, only chain structures with fault throw parameter $h$ at their head have not shown a significant
7.3. Testing results

Table 7.3: IC Fault Model optimisation with PBILh-chain: Run statistics. Marked are the runs used for plotting the history-matched production plots presented in Fig. 7.7 and Fig. 7.13 through to Fig. 7.18

<table>
<thead>
<tr>
<th>Chain ordering</th>
<th>$P_{10}$</th>
<th>$P_{50}$</th>
<th>$P_{90}$</th>
<th>$MF_{\text{min}}$</th>
<th>Fittest model parameters $(h, k_{\text{good}}, k_{\text{bad}})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>No chain</td>
<td>1052.3*</td>
<td>4529.1*</td>
<td>17584*</td>
<td>184.7*</td>
<td>(15.21, 1.29, 2.4)*</td>
</tr>
<tr>
<td></td>
<td>676.4</td>
<td>4104.9</td>
<td>20083</td>
<td>113.5</td>
<td>(10.97, 1.33, 2.79)</td>
</tr>
<tr>
<td></td>
<td>696.9</td>
<td>3536.0</td>
<td>15175</td>
<td>205.7</td>
<td>(21.2, 1.33, 3.65)</td>
</tr>
<tr>
<td>$h \rightarrow k_{\text{good}} \rightarrow k_{\text{bad}}$</td>
<td>1175.6*</td>
<td>4527.9*</td>
<td>22847*</td>
<td>180.52</td>
<td>(6.95, 1.31, 3.93)*</td>
</tr>
<tr>
<td></td>
<td>1118.5</td>
<td>5005.7</td>
<td>33762</td>
<td>141.83</td>
<td>(18.24, 1.32, 3.54)</td>
</tr>
<tr>
<td></td>
<td>973.22</td>
<td>4476.1</td>
<td>24933</td>
<td>148.98</td>
<td></td>
</tr>
<tr>
<td>$h \rightarrow k_{\text{bad}} \rightarrow k_{\text{good}}$</td>
<td>840.28*</td>
<td>4284.1*</td>
<td>26010*</td>
<td>181.74*</td>
<td>(19.92, 1.32, 3.08)*</td>
</tr>
<tr>
<td></td>
<td>739.12</td>
<td>3800.7</td>
<td>32167</td>
<td>112.72</td>
<td>(9.232, 1.345, 2.501)</td>
</tr>
<tr>
<td></td>
<td>1048.99</td>
<td>4746.1</td>
<td>25408</td>
<td>131.37</td>
<td>(15.134, 1.311, 3.228)</td>
</tr>
<tr>
<td>$k_{\text{good}} \rightarrow h \rightarrow k_{\text{bad}}$</td>
<td>600.18*</td>
<td>3660.1*</td>
<td>20235*</td>
<td>77.39*</td>
<td>(18.48, 1.31, 2.67)</td>
</tr>
<tr>
<td></td>
<td>820.82</td>
<td>3951.7</td>
<td>17956</td>
<td>171.67</td>
<td>(9.75, 1.33, 1.24)</td>
</tr>
<tr>
<td></td>
<td>602.64</td>
<td>3767.0</td>
<td>17736</td>
<td>22.88</td>
<td></td>
</tr>
<tr>
<td>$k_{\text{good}} \rightarrow k_{\text{bad}} \rightarrow h$</td>
<td>950.90*</td>
<td>4106.2*</td>
<td>23621*</td>
<td>132.02*</td>
<td>(30.46, 1.34, 2.29)*</td>
</tr>
<tr>
<td></td>
<td>503.35</td>
<td>3627.6</td>
<td>19888</td>
<td>101.66</td>
<td>(11.71, 1.32, 2.26)</td>
</tr>
<tr>
<td></td>
<td>697.31</td>
<td>3945.6</td>
<td>22250</td>
<td>25.83</td>
<td>(9.7, 1.33, 1.62)</td>
</tr>
<tr>
<td>$k_{\text{bad}} \rightarrow h \rightarrow k_{\text{good}}$</td>
<td>1534.6*</td>
<td>6730.4*</td>
<td>29608*</td>
<td>119.05*</td>
<td>(12.27, 1.32, 3.2)*</td>
</tr>
<tr>
<td></td>
<td>1866.5</td>
<td>8454.7</td>
<td>50154</td>
<td>140.62</td>
<td>(30.47, 1.33, 1.98)</td>
</tr>
<tr>
<td></td>
<td>913.17</td>
<td>5433.5</td>
<td>41972</td>
<td>45.06</td>
<td>(10.15, 1.33, 2.03)</td>
</tr>
<tr>
<td>$k_{\text{bad}} \rightarrow k_{\text{good}} \rightarrow h$</td>
<td>938.72*</td>
<td>4387.5*</td>
<td>31920*</td>
<td>12.67*</td>
<td>(10.34, 1.32, 1.62)*</td>
</tr>
<tr>
<td></td>
<td>1379.1</td>
<td>5281.3</td>
<td>40237</td>
<td>189.63</td>
<td>(30.42, 1.33, 2.02)</td>
</tr>
<tr>
<td></td>
<td>876.15</td>
<td>4333.1</td>
<td>36930</td>
<td>35.37</td>
<td>(10.57, 1.32, 1.76)</td>
</tr>
</tbody>
</table>

Improvement. The winners here, once again, are the PDS with $k_{\text{good}}$ being at the head of the dependency chain with such MF values as 22.88 (for the case of $k_{\text{good}} \rightarrow h \rightarrow k_{\text{bad}}$) and 25.82 (for the case of $k_{\text{good}} \rightarrow k_{\text{bad}} \rightarrow h$). Only slightly worse performed the remaining two PDS with $k_{\text{bad}}$ at the head of the chain, with the lowest misfit values of 45.06 and 35.37.

Note on the quality and diversity of history-matched models

By applying a $P_{10}$ misfit function cut-off we select groups of fittest models from the first run for each PBILh-chain ordering and plot the simulated production profiles...
in Fig. 7.13 through to Fig. 7.18 (runs statistics can be found in Table 7.3).

First, let us comment on the quality and diversity of the selected $P10$ equivalent groups of the history-matched models for each of the considered chain orderings. As it was observed in the analysis of the algorithm’s convergence properties, parameter chosen to represent the “head” of the dependency chain has a significant influence on the success of PBILh-chain sampling and therefore quality of the history-matched models.

The PBILh-chain, based on PDS with fault throw parameter $h$ in the first position within the dependency chain (Fig. 7.13 and Fig. 7.14), has mostly produced models characterised by an increasing overproduction of water (which was mainly due to the increased water injection quantities). Though, in the case of $h \to k_{bad} \to k_{good}$ chain ordering (Fig. 7.14) one is also able to differentiate a smaller group of models showing levels of water and oil production lower than the historically observed and predicted ones. Statistics of the three performed test runs for the two versions of dependency chain, and specifically that of the indicated overall fittest sampled models, which is given in Table 7.3, shows that PBILh-chain failed to sample close to the location of the global optimum. Most of the six considered fittest models are actually located around the same local optima solution. The algorithm is seen consistently overestimating the poor quality sand permeability $k_{bad}$.

Runs of PBILh-chain with the good quality sand permeability parameter $k_{good}$ at the head of the chain (Fig. 7.15 and Fig. 7.16) were able to produce a good spread of the history matched models. By sampling according to PDS structure of the $k_{good} \to h \to k_{bad}$ type, algorithm has achieved an even spread of production dynamics of the history-matched models with only a minor tendency towards oil and water overproduction. A variety of models with both low and high water injection rates were sampled. For $k_{good} \to h \to k_{bad}$ chain type the resulting spread in cumulative oil production was smaller than in the previous case and most of the sampled models indicated injection quantities to be slightly lower than seen in historical data. First among the three performed runs for $k_{good} \to h \to k_{bad}$ was able
7.3. Testing results

Figure 7.13: IC Fault Model optimisation with PBILh_chain – Chain ordering $h \rightarrow k_{good} \rightarrow k_{bad}$. 10% of the fittest sampled models are plotted. A solid red line represents historical data.

Figure 7.14: IC Fault Model optimisation with PBILh_chain – Chain ordering $h \rightarrow k_{bad} \rightarrow k_{good}$. 10% of the fittest sampled models are plotted. A solid red line represents historical data.
7. PBILh with pairwise conditional sampling

Figure 7.15: IC Fault Model optimisation with PBILh chain – Chain ordering \(k_{\text{good}} \rightarrow h \rightarrow k_{\text{bad}}\). 10% of the fittest sampled models are plotted. A solid red line represents historical data.

Figure 7.16: IC Fault Model optimisation with PBILh chain – Chain ordering \(k_{\text{good}} \rightarrow k_{\text{bad}} \rightarrow h\). 10% of the fittest sampled models are plotted. A solid red line represents historical data.
to sample two models within the closest proximity of the global optimum: 1) \( h = 10.085 \, ft, \, k_{good} = 132.1 \, mD \) and \( k_{bad} = 1.399 \, mD \), 2) \( h = 9.753 \, ft, \, k_{good} = 132.9 \, mD \) and \( k_{bad} = 1.236 \, mD \). Most importantly, as these two models and the posterior conditional distribution estimates shown in Fig. 7.11 demonstrate, PBILh-chain was able to correctly identify the global region of interest for parameter \( k_{bad} \).

The remaining two tests of PBILh-chain based on dependency chain structures with the leading element being poor sand quality permeability \( k_{bad} \) have sampled very different groups of history-matched models. In the case of \( k_{bad} \rightarrow h \rightarrow k_{good} \) chain (Fig. 7.17), similar to the results observed for the \( h \rightarrow k_{good} \rightarrow k_{bad} \) PDS structure in Fig. 7.13, two sets of history matched models can be differentiated. In this particular case though, there is almost an equal number of both over and under performing models (in terms of oil and water production quantities). PBILh-chain with the \( k_{bad} \rightarrow k_{good} \rightarrow h \) PDS structure samples a big cluster of models whose production dynamics comes to a close agreement with the available historical data. Quality of the overall fittest model sampled by the algorithm reaches \( MF = 35.37 \) for the following set of model parameters: \( h = 10.57 \, ft, \, k_{good} = 132.2 \, mD \) and \( k_{bad} = 1.76 \, mD \). Algorithm, overall, still slightly overestimates the values of the poor quality sand permeability, however the overestimation is not as significant as in the case of classical version of PBILh or runs shown in figures Fig. 7.13 and Fig. 7.14.

PBILh-chain implementation with PDS of the form \( k_{good} \rightarrow h \rightarrow k_{bad} \) can therefore be identified as one of the best performing ones. Such conclusion can be made based on its ability to sample close to the location of the global optimum as well as to produce a good spread in sampled models’ production quantities. Let us compare 3D images of the parameter space by classical version of PBILh (Fig. 7.19) and PBILh-chain with the selected PDS structure (Fig. 7.20). The main algorithm parameters such as learning rate \( \alpha \) and parallelization scheme specifications are identical for both runs. We can clearly see that PBILh fails to sample the area of the parameter space corresponding to the global optimum location, this is mostly due to significant
Figure 7.17: IC Fault Model optimisation with PBILh\_chain – Chain ordering $k_{\text{bad}} \rightarrow h \rightarrow k_{\text{good}}$. 10% of the fittest sampled models are plotted. A solid red line represents historical data.

Figure 7.18: IC Fault Model optimisation with PBILh\_chain – Chain ordering $k_{\text{bad}} \rightarrow k_{\text{good}} \rightarrow h$. 10% of the fittest sampled models are plotted. A solid red line represents historical data.
7.3. Testing results

Figure 7.19: Image of the solution space sampled by PBILh with $\alpha = 0.1$, constructed by projecting a 3D solution space image onto the three 2D coordinate planes: “fault throw - good quality sand”, “fault throw - poor quality sand” and “good quality sand - poor quality sand” (as first plotted in Fig. 5.32)

Figure 7.20: IC Fault Model optimisation with PBILh_chain – Chain ordering $k_{good} \rightarrow h \rightarrow k_{bad}$. Image of the sampled solution space represented through its 2D projections
overestimation of poor quality sand permeability $k_{bad}$ values. PBILh-chain with the $k_{good} \rightarrow h \rightarrow k_{bad}$ chain dependency structure on the other hand successfully samples the area of the parameter space in question, while still sampling alternative local optima location at the same time.

7.4 Summary

In the current chapter we have outlined the main ideas and principles of operation of the suggested extension of the researched histogram-based Population-Based Incremental Learning algorithm into the domain of conditional sampling of continuous parameter spaces - PBILh-chain algorithm.

Chain-like dependency structures were used to encode possible pairwise dependencies between model parameters. A synthetic IC Fault model was used in order to study the influence conditional sampling of optimisation variables can have on the performance and success of the automatic history-matching optimisation process.

It was shown that conditional sampling can have a profound effect on the quality, diversity and accuracy of the history-matched models for the specific considered dataset. Given the size of the model (three optimisation parameters), a total of 6 chain-like dependency structures were constructed and used for conditional sampling within the PBILh-chain algorithm framework. It was shown that ordering of the chain elements (order in which model parameters were sampled) can have both positive and negative effect on the results of the history-matching process. The overall trend observed in the performed tests was that PBILh-chain, irrespective of the chain structure used, was able to sample a much wider range of models (in terms of their misfit values) that that sampled by classical version of PBILh. The best results were observed for two PDS chain structures, where head of the chain was assigned to the good quality sand permeability parameter $k_{good}$. In both cases PBILh-chain was not only able to sample models with considerably lower misfit values than those obtained in other performed tests, but also perform an effective
parallel search within the parameter space successfully sampling both global and local optima regions.
Chapter 8

Conclusions

8.1 Discussion of results

In this thesis a first application of the class of Estimation of Distribution Algorithm to the problem of reservoir history-matching optimisation problem was presented.

First, a unified framework for classification of the optimisation algorithms applied for reservoir history-matching optimisation was suggested in Chapter 2. A brief description of the theory and application practice of the most widely applied techniques was given. We have indicated that with an increased availability of the parallel cluster computing, the automatic history-matching optimisation methods are currently receiving much more attention from the industry, especially those based on the evolution principles observed in the natural world.

Among such algorithms, stochastic techniques such as Genetic Algorithms, Evolutionary Strategies and Neighbourhood Algorithm have been covered. These algorithms have had a good track record (GAs – Romero et al. (2000); Romero and Carter (2001); Schulze-Reigert et al. (2004); Velez-Langs (2005); Castellini et al. (2006), ES – Schulze-Riegert et al. (2001, 2002, 2003); Al-Shamma and Teigland (2006); Griess et al.; Choudhary and Ludvigsen (2007) and NA – Christie et al. (2001); Subbey et al. (2002, 2003); Suzuki and Caers (2006); Pickup and Christie (2006); Erbaş (2007)) of successful applications within the industry, and drawing from their main principles of population utilization (implicit search parallelism) in this thesis we proposed a technique, which is somewhat similar to the two stochastic methods.

Taking its origin from and being an alternative to genetic algorithms, EDAs are a group of methods that learn the structure of the problem as they work their way
through it and use this information to ensure a proper mixing and improvement of sampled individuals (reservoir models). The approach is based on probabilistic modelling of promising solutions to guide the exploration of the search space instead of using crossover and mutation like in the case of standard GA.

In Chapter 3 a brief introduction to the class of Estimation of Distribution Algorithms was given. We have introduced algorithms suitable for optimisation of discrete as well as continuous search spaces. It was shown that higher order EDAs are able to encode and sample models with multivariate dependencies between their parameters, however even the univariate-type EDAs have been demonstrated to be a powerful optimisation tool.

Within the framework of reservoir history-matching optimisation, we focused on the class of continuous univariate EDAs based on histogram models. The main focus of this thesis is the continuous version of one of the earliest model-based EDAs – a histogram-based Population-Based Incremental Learning Algorithm (PBILh). We consider PBILh to be more accurate in reflecting (approximation) the true probability distribution over the space of all possible solutions due to the way it approximates the generation-specific underlying likelihood distributions. Two major positive characteristics of such approach were highlighted. First being the absence of the selection step, this means that the algorithm utilizes all the available information from the processed candidate solution, even those with low fitness. Second, and the most important characteristic, is that an individual’s fitness is used to construct and update the probability model. In this way we believe PBILh has better chances of accurately approximating the true underlying probability distribution than a frequency-count-based technique.

Although in this thesis we have concentrated on the continuous version of the researched algorithm, in Chapter 4 a background theory and principles of operation were given for both the original binary representation of the algorithm (PBILb) and it’s continuous histogram-based extension. Two synthetic optimisation problems (mathematical functions) were used to illustrate a step-by-step workflow of
the algorithms studied. It was shown that both algorithms can be easily adopted for optimisation within both discrete (for example binary) and continuous solution spaces. However, in cases where one seeks to optimise continuous problem domain, results of the conducted tests demonstrated that PBILh considerably outperformed PBILb in terms of convergence speed, while still providing good quality of sampled solutions. PBILh has shown an ability to sample all the existing optima within the solution space (for continuous function optimisation example), while PBILb has eventually concentrated its search in only one of the optima regions.

Chapter 4 also presented results of an extensive study of the algorithm’s control parameters such as learning and search rate, size of initial and current population, problem domain discretisation (length of the binary coded solution string for PBILb and histogram discretisation in PBILh), as well as fitness function scaling strategy. For PBILh it was noted that such control parameters as learning rate and fitness function scaling coefficient, together with histogram discretisation have proved to be the most influential tuning parameters of the algorithm. Important observation was also made in regards to the size of initially sampled generation of individuals. Test results have shown that PBILh is relatively insensitive to the size of the sample drawn in first generation, typically uniformly, in the way that it does not influence algorithm’s ability to accurately approximate the underlying probability distribution.

In order to improve the overall efficiency of the reservoir history-matching optimisation process two issues were tackled in this thesis: 1) parallelization of the optimisation algorithm and 2) adequate definition of the misfit function used to evaluate fitness of the sampled individuals (reservoir model realisations). In Chapter 5 we have introduced a new robust parallelization scheme which can easily be implemented for parallelization of any evolutionary algorithm, and especially EDA due to their population-based nature. The proposed scheme is based on an asynchronous Master/Slave parallel setup and is not only able to provide an expected CPU speed-up to the process, but also effectively manage any possible software or
hardware related faults within the optimisation loop. The suggested setup makes it possible to utilize heterogeneous parallel clusters.

We have suggested an improved definition of the misfit function, where the attention was concentrated on the way the measured data are weighted in the process of misfit function calculation. The proposed improvements were aimed predominantly at providing an equally sensitive misfit function response (treatment) to both early and late water breakthrough in the reservoir model.

The researched algorithm was tested on one synthetic and two real-world reservoir history-matching optimisation problems. In Chapter 5

As a synthetic dataset, the IC Fault model was used to illustrate the application potential of the histogram-based PBIL for reservoir history-matching optimisation studies. We have studied the algorithm’s ability to perform fast and yet explorative enough search within the parameter space while at the same time being able to accurately approximate the underlying uncertainty in model parameters. The influence of the control parameters for the algorithm such as learning rate and size of the initial population was investigated.

When optimising a large size parameter space, the algorithm was able to sample a diverse set of history matched models. During the course of optimisation estimates of the posterior marginal probability distributions for the IC Fault Model parameters have evolved into stable shape distributions, therefore helping PBILh to slowly concentrate its search in the areas of highest likelihood. The algorithm has demonstrated performance of a typical global optimisation technique by identifying and simultaneously resampling a number of such areas.

The performed tests have shown that an increased value of the learning rate parameter $\alpha$ can positively influence convergence speed of the algorithm and, at times, the quality of the sampled model. However high values of $\alpha$ proved to have a very negative effect on the quality of the posterior marginal distribution estimates of the model parameters. The distribution estimates obtained from PBILh runs with high values of $\alpha$ were unstable and uninformative. On the other hand, lower values of
the learning rate helped the algorithm to perform a much more thorough sampling within the parameter space slowly learning the topology of the underlying distribution; this of course has also resulted in much slower convergence rates of the algorithm.

The size of the initial population, which was commonly believed to be one of the most influential control parameters of the Population-Based Incremental Learning algorithm, did not show any significant influence on the optimisation outcome judging by the results of the performed tests. It was understood that, given the implemented asynchronous parallelisation setup, this parameter does not have any influence on the quality of the sampled models. In cases where varying size of initial generation of individuals was used with low values of learning rate parameter $\alpha$, it did not significantly influence the estimates of posterior marginal probability distributions of model parameters either. This means that, even with minimal amount of prior information, the algorithm is still able to consistently sample the underlying probability distribution. And in reservoir engineering, and history-matching studies specifically, where accurate prior information is rarely available or is of a very coarse scale, such a property of optimisation algorithm can prove to be very beneficial.

The overall observation after the performed tests on a synthetic data set, in terms of algorithm performance, is that while providing a much faster convergence rate, higher values of $\alpha$ jeopardize the ability of the algorithm to accurately estimate and evolve statistically informative posterior distributions for those optimisation parameters which are characterised by a presence of multiple optima. If the aim of history-matching study is to sample as fit models as possible as quickly as possible, without putting a lot of emphasis on the informative nature of the posterior marginal distribution estimates, then as high values of learning rate as $\alpha = 0.5$ can be perfectly justified. However if uncertainty in the input parameters is recognised and considered to be important to estimate (due to the conditions of the available prior data or the lack of it for that matter), a lower values of $\alpha$ would do a better job.
assessing this task.

A first real field history-matching optimisation example of Field 1 (Chapter 6, Section 6.1) has shown us that a blind selection of a high number of optimisation parameters without the performance of additional screening tests may result in a very noisy distribution estimates. This suggestion was backed up by the results of the second history-matching study of Field 2 (Capter 6, Section 6.2). Here, with the aid of folded Placket-Burman type experimental design only the most influential model parameters took part in the optimisation process. PBILh was able to provide informative uncertainty estimates, which in some cases were able to clearly identify existence of multiple optima regions. It was also shown, on the example of the 10 best history-matched models in the case of Field 1, that optimisation based on the variations of only global reservoir properties may prove inefficient and have little effect on the simulated production performance of some of the wells. Further local changes may need to be applied.

After testing the researched optimisation algorithm on both real-world reservoir models it became clear that relationships between optimisation parameters and sensitivity of the forward modelling output to changes in these parameters can have a profound effect on the success of the history-matching optimisation process.

The core of an efficient history-matching optimisation technique is its sampling quality. Any extra information capable of guiding the search within the solution space based on the assumptions of dependence or independence between the optimisation/model parameters should be welcomed into the optimisation framework. Although reservoir engineer is always aware of the possibility of such interactions being present within the studied model, currently, none of the stochastic optimisation techniques used in the industry, such as simulated annealing and evolutionary algorithms (namely genetic algorithms and evolutionary strategies), explicitly take into account this extra information of possible parameter interactions.

Theoretical background and primary research results presented in the Chapter 7 are a first step towards giving an engineer an algorithm which is capable of sam-
pling conditional probability distributions of model parameters and using this information to improve the quality of the solution space exploration and ultimately of the history-matched models themselves. An extension to the histogram-based Population Based Incremental Learning algorithm capable of handling chain-type pairwise dependencies between model parameters was developed. A new technique was given a notation of PBILh-chain.

A synthetic data set (IC Fault Model) was used to test the performance of PBILh-chain. It was shown that conditional sampling can have a profound effect on the quality, diversity and accuracy of the sampled history-matched models. The proposed algorithm is not only able to sample models with lower misfit values than those obtained from unconditional sampling, but also perform an efficient parallel search within the parameter space successfully sampling both global and local optima regions.

8.2 Main research contributions

- First application of a representative of the class of Estimation of Distribution Algorithms for reservoir history-matching optimisation. Algorithm was successfully applied to optimisation of synthetic as well as real-field history-matching optimisation problems.

- Development and successful implementation of an asynchronous Master/Slave parallelisation setup for the implemented algorithm. The setup can easily be used by any other evolutionary (population-based) optimisation technique and implemented on a heterogeneous parallel cluster. The proposed parallel setup is efficient, robust and able to minimize both software and hardware related CPU losses.

- Development of the theoretical basis and primary testing of the extension to the continuous histogram-based version of the Population-Based Incremental
Learning algorithm capable of handling chain-type pairwise dependencies between model parameters. It was shown that conditional sampling can have a profound effect on the quality, diversity and accuracy of the sampled history-matched models.

8.3 Future research directions

8.3.1 What’s next for the application of EDAs?

- Since the learning rate parameter $\alpha$ is the most influential control parameter of the Population-Based Incremental Learning algorithm (for both its discrete and continuous implementations) it is natural that there exists a range of strategies for the parameter value choice and alterations. One of such strategies is a so-called self adapting $\alpha$. The suggested approach constitutes in the selection of some low base-value of $\alpha$ in the first generation $t = 0$ and then incrementing its value if the average fitness of the last few processed generations to date is consistently improving. By doing this the algorithm may start of by performing a very refined global search within the parameter space. Once it is able to identify a potential area(s) of interest within the parameter space and consecutively sample fitter and fitter individuals from these areas, the decision is made to increment the learning rate value, therefore increasing the size of the steps algorithm takes within the parameter space towards the identified optima locations. This way the initially global search will transform into a more localised one, sampling within the close neighbourhood of the identified optima region(s).

- In the work presented by Pelikan et al. (2002), the authors have argued that the application of the FHH histogram structure with varying size of bins can efficiently and accurately approximate the underlying probability distribution. We suggest using this principle in conjunction with the PBILh framework and
evolve a histogram where both height and width of the bins will be variable. In some ways a mechanism to refine the parameter (bin) ranges can be borrowed from the Neighbourhood Algorithm. The way we suggest of doing it is to refine a bin if its likelihood value, which is a function of fitness of the best sampled model in that range of the bin, is increasing and merge (coarsen) the bin with the nearby ones if it doesn’t. It is expected that by utilising the fitness-based structure of the histogram and with the added flexibility of its discretisation, the PBILh with self-adapting bin width will be able to search parameter space in a similar way that the Neighbourhood Algorithms does, with highly refined histogram ranges corresponding to the potential optima regions within the solution space.

- Based on the ideas used to develop the PBILh extension capable of handling chain-type pairwise dependencies between model parameters, it is also possible to modify the proposed framework to encode and sample from the tree-type dependency structures. In this case a number of parameters will share the same “parent”, a parameter which will represent the “root” of the tree and will be sampled first, independently of other parameters. The subsequent parameters will be sampled from a conditional distribution chosen from the joint probability distribution matrixes based on the sampled value of the “root” parameter.

### 8.3.2 What’s next for reservoir history-matching optimisation?

- Particle Swarm Optimisation (PSO) is one of the prospective, interesting and challenging areas of future research in the area of new optimisation methods for petroleum industry as a whole. PSO is a collective, anarchic (in the original sense of the term), iterative method, with the emphasis on the cooperation; it is partially random and is performed without selection. The basic model and inspiration for PSO is an information exchange between bees. Each particle
chooses the location of its next movement based on the three main choices: its current velocity, the best position it has found up to now and the best position found by its informants. These informants are selected at random in each iteration. The number of informants is a control parameter of the algorithm, just like the total size of the swarm. Clerk (2006) is a good starting point for further reading on the subject of Particle Swarm Optimisation.

In the book, the author argues that even if the swarm is large compared to the number of informants per particle, it can be shown that propagation of information occurs very quickly.

- As highlighted in Chapter 2 most of the optimisation techniques used in the industry are operated by means of evaluating the quality of the generated models by a single objective function value, which is as the sum of squared differences between the historical and the simulated production quantities. Each of the differences is then weighted individually in order to reflect different levels of errors or sensitivities expected for these quantities by engineers. While it is possible to produce and study the output of the contributions made by each of the production quantities’ mismatches, this data is not explicitly used within the optimisation routine.

Multi objective optimisation can and should be the way forward. The approach constitutes in splitting a single objective function into multiple objective functions. The added benefit of such representation is that it is possible to investigate the trade offs between each of them. As pointed out by De Jong (2006), the ultimate aim here is to produce/sample a model for which none of the multiple objectives is dominating, therefore achieving an equal level of match for all the reservoir quantities concerned.

The trade off curve comprised of such multiple objective function evaluation points is referred to as a *Pareto optimal front*. It is assumed that the points which are located closer to the Pareto front dominate points that are farther
8.3. Future research directions

away. Here, one can either estimate the number of points dominated by a
certain member of the population by calculating how many other members of
population it dominates, or alternatively, estimate a dominance of a point by
the size of geometric area below it.

Multi objective optimisation is specifically suited for the joint application with
evolutionary optimization techniques. Since these techniques are population-
based, it is possible to design them in such a way that at each iteration algo-
rithm will provide a selection of evaluated points which can be used to con-
struct a Pareto curve.

- Drawing on the basis of the population-based nature of EAs it was already
  shown in this thesis that they are perfectly suited for parallel computing. A
  variety of parallelisation techniques exist, but most importantly Estimation
  of Distribution algorithms evolve themselves, in order to cater for this avail-
able parallel cluster adaptation. An evolutionary algorithm with multiple si-
multaneously evolved populations is just one such example. One of the ba-
sic examples is a dual-PBIL algorithm, where two populations of individuals
are evolved simultaneously by means of evolving two separate probability
vectors. There, of course, exist algorithms where multiple populations are
evolved simultaneously. Different approaches can be used to exchange the
information between the sub-populations. Randomly selected groups of in-
dividuals can be swapped between sub-populations, or even the probability
models they use.

The important thing to remember here is that while effective utilization of
such finely- and coarsely-grained parallelisation schemes (which were covered
in Chapter 5) in conjunction with evolutionary optimization algorithms, and
namely Estimation of Distribution Algorithms, is important, it is the problem
of convergence to the local minima that we primarily aim to solve by evolving
a number of populations of individuals simultaneously.


Carter, J., Ballester, P., Tavassoli, Z. and King, P.: 2004, Our calibrated model has no predictive value: An example from the petroleum industry, in K. Hanson and


_e-Handbook of Statistical Methods_: 2003, NIST/SEMATECH.


of the Genetic and Evolutionary Computation Conference, Vol. 1, Morgan Kaufmann,
Orlando, Florida, USA, pp. 840–846.

Gao, G., Zafari, M. and Reynolds, A.: 2006, Quantifying uncertainty for the PUNQ-
S3 problem in a bayesian setting with rml and enkf, (SPE 93324).

Geman, S. and Geman, D.: 1984, Stochastic relaxation, gibbs distributions, and
(6), 721–741.

Goldberg, D.: 1983, Computer-aided gas pipeline operation using genetic algorithms and
rule learning, PhD thesis, Univ. of Michigan, Ann Arbor, MI, USA.

Goldberg, D.: 1989, Genetic Algorithms in Search, Optimisation and Machine Learning,
Addison-Wesley Publishing Co. - Reading, Mass.

Gomez, S., Gosselin, O. and Barker, J.: 1999, Gradient-based history-matching with
a global optimisation method, SPE Journal 3(SPE 56756).

González-Rodríguez, P., Kindelan, M., Moscoso, M. and Dorn, O.: 2005, History
matching problem in reservoir engineering using the propagationbackpropaga-
tion method, Inverse Problems (21), 565–590.

techniques for model validation and prediction scenarios for a north african oil
field, 2006(SPE 100193-MS).

Gross, H.: 2006, History Matching Production Data Using Streamlines and Geostatistics,
PhD thesis, Department of Petroleum Engineering, Stanford University.

Gu, Y. and Oliver, D.: 2004, History matching of the punq-s3 reservoir model using
the ensemble kalman filter, (SPE 89942).

matching using the ensemble kalman filter on a north sea field case, (SPE 102430).


Hu, L.: 2002, Combination of dependent realizations within the gradual deformation method, Mathematical Geology 34(8), 953–963.


IC Fault Model web page: n.d.


Appendix A: Example of a single iteration of PBIL

Binary PBIL

For the ease of visual representation a 6-bit long OneMax example for this step-by-step illustrative has been chosen.

Given the PBILb framework (Section 4.1) which starts at the initialization phase and continues through the three iterative steps (sampling, evaluation and updating), here we manually illustrate the single iteration of the algorithm.

Initialisation

Here, a 6-bit prototype vector $P(t)|_{t=0}$ is initialised in the following way

$$P(t) = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$$

The value of prototype (probability) vector positions of 0.5 enables first generation of individuals to be sampled uniformly.
Sampling step

According to the Eq. (4.3), given the current state of the prototype vector $P(t)$ (with elements $p_i$) and $N$ uniformly sampled vectors $R(t)$ (with elements $r_i^j \in 0...N-1$)

$$x_i^j(t) = \begin{cases} 1 & \text{if } p_i(t) > r_i^j(t) \\ 0 & \text{otherwise} \end{cases}$$

a new vector representing a sampled individual $X(t)$ will be generated, as it is shown on the following example:

<table>
<thead>
<tr>
<th>$P(t)$</th>
<th>$R(t)$</th>
<th>$X(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>&gt;</td>
<td>0.01</td>
</tr>
<tr>
<td>0.5</td>
<td>&gt;</td>
<td>0.23</td>
</tr>
<tr>
<td>0.5</td>
<td>&lt;</td>
<td>0.75</td>
</tr>
<tr>
<td>0.5</td>
<td>&gt;</td>
<td>0.12</td>
</tr>
<tr>
<td>0.5</td>
<td>&lt;</td>
<td>0.49</td>
</tr>
<tr>
<td>0.5</td>
<td>&lt;</td>
<td>0.81</td>
</tr>
</tbody>
</table>

Evaluation of individuals and selection of the fittest individual (T=t)

Let us assume that the size of the newly sampled generation of individuals $X_j(t)$ was $N = 6$. Since we aim to maximise the sum of elements (binary positions) in sampled individuals, out of the processed generation individual $X_0(t)$ has the highest function value of $OneMax(X_0(t)) = 5$, therefore it is adopted as the fittest individual of the generation $X_{best}$.

The sampled values of the binary vector positions for each of the sampled individuals are shown below, with their OneMax (sum of all elements) evaluations
Updating the probability model (T=t)

Using the general update rule of PBILb according to Eq. (4.2)

\[ P(t + 1) = (1 - \alpha)P(t) + \alpha X_{\text{best}}(t) \]

and assuming the learning rate value of \( \alpha = 0.1 \) we update our probability model encoded in the prototype vector \( P(t) \) towards the fittest individual in the current generation \( X_0(t) \).

Prototype vector \( P(t + 1) \) is then used to sample new generation of individuals \( T = t + 1 \).
Histogram-based PBIL

We again use a 6-bit long OneMax problem in order to illustrate a single iteration of the histogram-based extension of PBIL.

Initialisation

As it was noted in Section 4.2, for each optimisation variable PBILh creates a separate histogram $H_i \in 0 \ldots (N_{\text{var}} - 1)$. In case of the OneMax problem we consider each of the bit positions to be an optimisation variable $V_i$, therefore $N_{\text{var}} = \text{size}(\text{OneMax}) = 6$.

$$X(t) = \begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ \ldots \\ V_{N_{\text{var}} - 1} \end{bmatrix}$$

Due to the fact that each bit position can only take two possible values 0 and 1, each of the histograms $H_i$ will be formed by only two bins (Fig. A.1, a) and will represent a discrete probability distribution.

A discrete probability distribution relates each value of a discrete variable with its non-zero probability of occurrence. Thus, a discrete probability distribution can always be presented in tabular form. So in the case of binary variable optimisation, one starts by assigning equal probabilities of 0.5 to both 0 and 1 being sampled in current binary vector position.

Sampling step

Variables $V_i$ are then sampled by, first, generating a uniform random number $R_i$, and then, depending on its value, sampling from the discrete probability distribution in
the following way (Fig. A.1, a, b):

\[ V_i = \begin{cases} 
0 & \text{if } 0 < R_i \leq 0.5 \\
1 & \text{if } 0.5 < R_i \leq 1.0 
\end{cases} \]

**Evaluation of individuals and construction of the temporary histogram**

Let us consider sampling the same set of individuals as we had in case of PBILb optimisation of the 6-bit long OneMax vector.

Since we require a fitness estimate for each of the sampled models, we need to define how this fitness will be defined. Given that we know the true value of the global optimum for a 6-bit OneMax problem is 6, we can set up the optimisation problem to be a minimisation problem. Therefore we will seek to minimize the difference between the optimum value and the evaluation of the sampled individual.

\[ MF_i = \max(\text{OneMax}) - \text{Onemax}(X_i) \]

where \( X_i \) is a sampled individual and \( MF_i \) is its misfit. The smaller this difference
$MF_i$ will be the higher will be the fitness of the individual $X_i$.

\[
X_0(t) \quad X_1(t) \quad X_2(t) \quad X_3(t) \quad X_4(t) \quad X_5(t)
\]
\[
\begin{bmatrix}
1 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

\[
\sum_i x_i \\ MF(X_j(t))
\]

5 3 2 1 3 3
1 3 4 5 3 3

Now we go one-by-one through each of the bit positions noting which of the bit (variable) values resulted in which OneMax evaluations.

We identify the lowest misfit obtained by both 0 and 1 being generated in the given binary position, this information is gathered in the Table A.1.

In order to transform our misfit evaluations $MF_i$ into the fitness evaluations $FF_i$ to be used in the process of temporary histogram construction, we use the fitness scaling rule in Eq. (4.20)

\[
FF_i = \exp(-MF_i/C_{scale})
\]

with fitness scaling coefficient $C_{scale} = 1$.

<table>
<thead>
<tr>
<th>bit position</th>
<th>$V_i = 0$ (min misfit)</th>
<th>$V_i = 1$ (min misfit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_0$</td>
<td>0 (3)</td>
<td>1 (1)</td>
</tr>
<tr>
<td>$V_1$</td>
<td>0 (3)</td>
<td>1 (1)</td>
</tr>
<tr>
<td>$V_2$</td>
<td>0 (3)</td>
<td>1 (1)</td>
</tr>
<tr>
<td>$V_3$</td>
<td>0 (1)</td>
<td>1 (3)</td>
</tr>
<tr>
<td>$V_4$</td>
<td>0 (3)</td>
<td>1 (1)</td>
</tr>
<tr>
<td>$V_5$</td>
<td>0 (4)</td>
<td>1 (1)</td>
</tr>
</tbody>
</table>

**Table A.1:** OneMax function optimisation with PBILh - Identification of the lowest misfit obtained by both 0 and 1 being generated in the given binary position
Temporary histograms $H_{temp}^i$ will then be formed in the following way:

$$H_{temp}(V_0 = 0) = \exp(-3) = 0.0498 \quad H_{temp}(V_0 = 1) = \exp(-1) = 0.3679$$

$$H_{temp}(V_1 = 0) = \exp(-3) = 0.0498 \quad H_{temp}(V_1 = 1) = \exp(-1) = 0.3679$$

$$H_{temp}(V_2 = 0) = \exp(-3) = 0.0498 \quad H_{temp}(V_2 = 1) = \exp(-1) = 0.3679$$

$$H_{temp}(V_3 = 0) = \exp(-1) = 0.3679 \quad H_{temp}(V_3 = 1) = \exp(-3) = 0.0498$$

$$H_{temp}(V_4 = 0) = \exp(-3) = 0.0498 \quad H_{temp}(V_4 = 1) = \exp(-1) = 0.3679$$

$$H_{temp}(V_5 = 0) = \exp(-4) = 0.0183 \quad H_{temp}(V_5 = 1) = \exp(-1) = 0.3679$$

Here $H_{temp}(V_i = 0)$ and $H_{temp}(V_i = 1)$ correspond to the first and a second bin in the temporary histogram $H_{temp}$ for a variable $V_i$. The bins’ magnitudes are calculated based on the fitness estimates on the individuals $X_i$ sampled with particular values (0 or 1) in the binary vector positions. So together $H_{temp}(V_i = 0)$ and $H_{temp}(V_i = 1)$ represent an unnormalised likelihood estimates for optimization parameters $V_i$.

Temporary histograms are then normalised to represent probability distributions:

$$H_{temp}(V_0 = 0) = \frac{0.0498}{0.0498 + 0.3679} = 0.1192$$

$$H_{temp}(V_0 = 1) = \frac{0.3679}{0.0498 + 0.3679} = 0.8808$$

$$H_{temp}(V_0 = 0) = \frac{0.0498}{0.0498 + 0.3679} = 0.1192$$

$$H_{temp}(V_0 = 1) = \frac{0.3679}{0.0498 + 0.3679} = 0.8808$$

$$H_{temp}(V_0 = 0) = \frac{0.0498}{0.0498 + 0.3679} = 0.1192$$

$$H_{temp}(V_0 = 1) = \frac{0.3679}{0.0498 + 0.3679} = 0.8808$$
Updating the probability model ($T = t$)

Using the general updating rule of PBILh given in Eq. 4.4:

$$H(t + 1) = (1 - \alpha)H(t) + \alpha H_{temp}(t)$$

the initial uniform probability distributions $H_i(t = 0)$ are updated towards the normalised temporary histograms $H_{temp}(t)$ in order to obtain the estimates of the posterior marginal probability distribution $H_i(t + 1)$.

$$H_0((t + 1) \mid (V_0 = 0)) = (1 - 0.1) \times 0.5 + 0.1 \times 0.1192 = 0.45 + 0.0119 = 0.4619$$

$$H_0((t + 1) \mid (V_0 = 1)) = (1 - 0.1) \times 0.5 + 0.1 \times 0.8808 = 0.45 + 0.0881 = 0.5381$$

$$H_1((t + 1) \mid (V_0 = 0)) = (1 - 0.1) \times 0.5 + 0.1 \times 0.1192 = 0.45 + 0.0119 = 0.4619$$

$$H_1((t + 1) \mid (V_0 = 1)) = (1 - 0.1) \times 0.5 + 0.1 \times 0.8808 = 0.45 + 0.0881 = 0.5381$$
\[ H_2((t + 1) \mid (V_0 = 0)) = (1 - 0.1) \times 0.5 + 0.1 \times 0.1192 = 0.45 + 0.0119 = 0.4619 \]

\[ H_2((t + 1) \mid (V_0 = 1)) = (1 - 0.1) \times 0.5 + 0.1 \times 0.8808 = 0.45 + 0.0881 = 0.5381 \]

\[ H_3((t + 1) \mid (V_0 = 0)) = (1 - 0.1) \times 0.5 + 0.1 \times 0.8808 = 0.45 + 0.0881 = 0.5381 \]

\[ H_3((t + 1) \mid (V_0 = 1)) = (1 - 0.1) \times 0.5 + 0.1 \times 0.1192 = 0.45 + 0.0119 = 0.4619 \]

\[ H_4(t + 1) \mid (V_0 = 3) = (1 - 0.1) \times 0.5 + 0.1 \times 0.1192 = 0.45 + 0.0119 = 0.4619 \]

\[ H_4((t + 1) \mid (V_0 = 1)) = (1 - 0.1) \times 0.5 + 0.1 \times 0.8808 = 0.45 + 0.0881 = 0.5381 \]

\[ H_5((t + 1) \mid (V_0 = 0)) = (1 - 0.1) \times 0.5 + 0.1 \times 0.0474 = 0.45 + 0.0047 = 0.4547 \]

\[ H_5((t + 1) \mid (V_0 = 1)) = (1 - 0.1) \times 0.5 + 0.1 \times 0.9526 = 0.45 + 0.0953 = 0.5453 \]

These estimates of the posterior marginal probability distribution \( H_i(t + 1) \) are then used as prior distributions to sample new generation of individuals \( T = t + 1 \).
Appendix B: OneMax and PEAKS model description

OneMax problem

In the OneMax problem, also known as a BitCounting problem, one seeks to optimise the following function

\[ f_{\text{OneMax}}(X) = \sum_i x_i \]

by finding its maximum value (i.e. when all the positions \( x_i, i \in 0 \ldots (N-1) \) in the binary vector \( X \) are equal to 1).

PEAKS problem

The function is essentially a mixture of multiple Gaussian distributions

\[
\begin{align*}
  f(x, y) &= + 3 \cdot (1 - x)^2 \cdot exp(-(x^2) - (y + 1)^2) \\
  &- 10 \cdot (x/5 - x^3 - y^5) \cdot exp(-x^2 - y^2) \\
  &- 1/3 \cdot exp(-(x + 1)^2 - y^2)
\end{align*}
\]

and is therefore characterised by the presence of multiple optima.

In this particular case we will seek to find a maximum of the absolute value of the PEAKS function \( f^*(x, y) = \text{abs}(f(x, y)) \) (Fig. B.2). Since it is a synthetic problem
we already know the exact location of the maximum:

$$\max (f^*(x, y)) = f^*(0.012, 1.524) = 8.0484$$

The marginal distributions have been estimated by numerical integration using the trapezium rule on a grid of points $1001 \times 1001$ on the interval $x \in (-3, +3)$ and $y \in (-3, +3)$. The results are shown in Fig. B.3.
Appendix C: IC Fault Model description

Geological model

The model consists of six layers of alternating good and poor quality sands (Fig. C.4). The three good quality layers have identical properties, and the three poor quality layers have a different set of identical properties. The thickness of the layers has an arithmetic progression, with the top layer having a thickness of 12.5 feet, the bottom layer a thickness of 7.5 feet, and a total thickness of 60 feet. The width of the model is 1000 feet, with a simple fault at the mid-point, which off-sets the layers. There is a water injector well at the left-hand edge, and a producer well on the right-hand edge. Both wells are completed in all layers, and operated at fixed bottom hole pressures.

Simulation model

The model is $100 \times 12$ grid blocks, with each geological layer divided into two simulation layers with equal thicknesses, each grid block is 10 feet wide. The model is constructed such that the vertical positions of the wells are kept constant and equal, even when different fault throws are considered. The well depth is 8325 feet to 8385 feet. To generate historical data a slightly heterogeneous reservoir realisation was created, with a throw of 10.4 feet. The porosity and permeabilities in each grid block were randomly drawn from uniform distributions with no correlations. The range
for the porosities was ±10% of the mean value, while the range for the permeabilities was ±1% of the mean value. The means for the porosities were: good quality sand, 0.30; poor quality sand, 0.15. The means of the permeabilities were: good quality sand, 137.6 mD; poor quality sand, 1.31 mD.

Production history

The simulation was run using the ECLIPSE simulator (*Eclipse User Guide* (2005a)), for 36 months, with three quantities (water production rate, $Q_{wp}$, oil production rate, $Q_{op}$, and water injection rate, $Q_{wi}$) being recorded at the end of each month (Fig.C.5). To each truth value, random Gaussian noise, with mean zero and standard deviation of 3% of the truth value (a minimum of 0.01 was allowed), was added. The model used to try and match the results was identical, except it was assumed that the good/poor quality sands were homogeneous. The porosities of the two sands were set to 0.30 or 0.15 as appropriate. It was assumed that the two permeabilities and the fault throw were the unknowns that needed to be found.
Figure C.4: IC Fault Model – reservoir model

Figure C.5: IC Fault Model – Oil and water production to be history-matched
Nomenclature

Chapter 1

\(d\) observed response of the model \(m\)
\(f()\) forward problem operator
\(F(m*)\) un-weighted least squares model misfit
\(g()\) inverse problem operator
\(m\) true model of a physical system
\(m*\) estimated model of a physical system, given available data

Chapter 2

\(A\) an event
\(d_i\) is an N-dimensional direction vector
\(D\) observed production data
\(D^N\) and N-dimensional experimental design
\(f(x_i)\) function evaluation at the given location \(x_i\) within the parameter space
\(h_i\) length of the candidate move within the parameter space performed along the direction of vector \(d_i\)
\(i^{(0)}\) initial realisation
\(I_i\) an experiment
\(N()\) normally distributed random number
output of the physical system which is approximated by the response surface

offspring of generation $t$

population of individuals at generation $t$

a marginal distribution of $A$

a probability of some event $A$ occurring given the production data $D$

perturbation parameter

response surface

selected set of individuals from population $P(t)$ in generation $t$

generation count or number of iterations

temperature parameter

coordinates of the spatial location within a reservoir

an approximation to the location of the problem solution within the parameter space

realisation of a Gaussian random field

coefficients of a polynomial

change in energy

size of offspring in evolutionary strategy

number of parents in evolutionary strategy

gradual deformation parameter

mixture coefficient

number of parameters being optimised

histogram constructed based on historical information or histogram representing a prior marginal probability distribution estimate

histogram constructed based on current information or temporary histogram representing a likelihood estimate based on the information obtained from the current generation of evaluated individuals
\( M - 1 \) length of the binary string chosen to code decimal parameter values
\( N \) size of the population of individuals
\( P(t) \) a probability or a prototype vector containing \( p_i, p(x_i) \) probabilities
\( p_i, p(x_i) \) a marginal distribution of the \( i \)th solution string position, probability of 1 being generated in the \( i \)th solution string position
\( p(x_i|x_j) \) conditional probability of sampling the value of the solution string position \( x_i \) given that of previously sampled \( x_j \)
\( V^i_k \) value of the \( i \)th variable of the \( k \)th best individual
\( V^{bin} \) a binary string
\( V^{dec} \) a decimal value
\( V^{dec_{max}} \) a maximal decimal value which can be coded by the binary string \( V^{bin} \)
\( V^{dec_{unscaled}} \) a decimal value coded by the binary string \( V^{bin} \)
\( V^{dec_{scaled}} \) a decimal value coded by the binary string \( V^{bin} \) and further scaled according to the desired decimal value range \( [V_{\text{min}}, V_{\text{max}}] \)
\( V_{\text{min}}, V_{\text{max}} \) ranges within which parameter \( V^{dec_{scaled}} \) is defined
\( x_i \) an \( i \)th solution string position
\( X_{\text{best}} \) solution string which represents the fittest individual sampled in generation
\( x_i^{\text{worst}}, x_i^{\text{best}} \) bit in the \( i \)th position of the best and worst of the two solution strings
\( x^*_i \) a parent of \( x_i \)
\( \alpha \) learning rate parameter
\( \Delta h^i_k \) increment that the \( k \)th best individual will make to the bin in histogram \( H^C(j) \) which it belongs to
\( \mu, \mu(t) \) mean of a Gaussian distribution
\( \mu_{\text{best}}, \mu_{\text{best}} \) statistics extracted from the first, second and worst individuals in the given generation of individuals
\( \sigma, \sigma(t) \) standard deviation of a Gaussian distribution
\( \sigma_{\text{reduce}} \) a reduction factor for standard deviation values
Chapter 4

c constant
\( C_{scale} \) fitness function scaling factor
\( D_n \) a decimal value coded by a binary string
\( D_n^{max} \) a maximal decimal value which can be coded by a binary string containing N bit positions
\( FF \) fitness function
\( g() \) scaling operator
\( h_1, h_2 \) global and local optima values respectively
\( H(t) \) histogram representing a prior marginal probability distribution estimate of generation \( t \)
\( H_{temp}(t), H_{like}(t) \) temporary histogram representing a likelihood estimate based on the information obtained from the current generation of evaluated individuals
\( M_{eval}^{max} \) maximal number of function (forward model) evaluations
\( M_{eval} \) number of function (forward model) evaluations
\( MF \) misfit or objective function
\( MF_{max}(t) \) maximum observed value of the misfit function up to current generation \( t \)
\( MF_{min}(t) \) minimum observed value of the misfit function up to current generation \( t \)
\( N, N_{var} \) number of optimisation variables
\( N_b, nmbBit, B \) length of the binary vector used to code an optimisation variable
\( nmbBins_i \) number of bins in the histogram representative of the \( i \)th optimisation parameter
\( p_i \) a marginal distribution of the \( i \)th solution string position, probability of 1 being generated in the \( i \)th solution string position
\( P(t) \) prototype or probability vector
\( P_{init}, H_{init} \) initial state of the prototype vector \( P(t_0) \) or histogram \( H(t_0) \)
\( r(t), R(t) \) uniformly distributed random number
\( s \) search rate or estimate of standard deviation (Section 4.4.5)
\( SizeFirstGen \) size of the initial generation of individuals (at \( t_0 \))
\( SizeGen \) size of any subsequent generations of individuals (at \( t > 0 \))
parameter which controls the magnitude of the selective pressure during the course of optimisation

\( v \)

optimisation variable defined within boundaries

\( (v_{\text{min}}, v_{\text{max}}) \)

ranges within which parameter \( v \) is defined

\( X_j \)
	solution string which represents the \( j \)th individual from a generation

\( X_{\text{best}} \)
	solution string which represents the fittest individual sampled in generation

\( x_i \)

the \( i \)th element of the binary solution string or model of the physical system or an individual

\( \alpha \)
a multiplier which takes a value of “+1” for maximization and “−1” for minimisation problems or otherwise learning rate parameter (LR)

\( \beta(t) \)

worst fitness value sampled in the last few generations

\( \Delta v \)
discretisation of the optimisation parameter ranges due to the chosen length \( n \) of the binary string used to code the parameter decimal value

\( \varepsilon \)

optimal width of the bin in a histogram

\( \mu^{MF(t)}, \sigma^{MF(t)} \)

are mean and standard deviation of the misfit function values in current generation respectively

\( \omega_1 \)

width of the global optimum

---

**Chapter 5**

\( C_{\text{scale}} \)

fitness function scaling factor

\( h \)
fault throw, ft (for the IC Fault Model)

\( H_i \)
histogram representing a marginal probability distribution estimate for the \( i \)th optimisation parameter

\( \text{Ind}(i), x_i \)
an individual or realisation of a model represented by a vector of parameter values that characterizes a particular solution

\( k_{\text{good}} \)
good quality sand permeability, mD (for the IC Fault Model)

\( k_{\text{bad}} \)
poor quality sand permeability, mD (for the IC Fault Model)

\( M_{\text{eval}}^{\text{max}} \)
maximal number of function (forward model) evaluations
$N_{init}$  
size of the initial generation of individuals (at $t_0$)

$N$  
a minimal size of population necessary to update the probability model (for parallelization scheme specifications) or number of optimisation parameters otherwise

$M$  
isize of the offspring population

$S_j(x_i)$  
a $j$th simulated production quantity of the $x_i$ individual (model)

$SizeFirstGen$  
size of the initial generation of individuals (at $t_0$)

$SizeGen$  
size of any subsequent generations of individuals (at $t > 0$)

$t_j$  
a positive integer

$WOPR$  
well oil production volume

$WOPT$  
well total oil production volume

$WWPR$  
well water production volume

$WWIR$  
well water injection volume

$\alpha$  
a weighting coefficient for misfit function calculation or learning rate otherwise

$\mu_j$  
a $j$th measured (historical) production quantity

$\sigma_j$  
a subjective measure of the perceived importance or the accuracy of the measurement

$\tau_0$  
a positive number

**Chapter 6**

$C_{scale}$  
fitness function scaling factor

$FLTMLT - \ldots 1$  
fault transmissibility multipliers

$HIST_j$  
a $j$th measured (historical) production quantity

$kh_{mult} \ldots 7$  
horizontal permeability multiplier

$kv_{mult} \ldots 1$  
vertical permeability multiplier

$MF$  
misfit or objective function (total)

$MFOIL,GAS$  
misfit or objective function (for oil and gas production volumes)

$MF_{WATER}$  
misfit or objective function (for water production volume)

$PERMZ - \ldots$  
vertical permeability multipliers

$pv_{mult} \ldots 1$  
pore volume multiplier

$SIM_j$  
a $j$th simulated production quantity

$SGS$  
static gradient survey (static pressure)

$WOPR$  
well oil production volume
Chapter 7

$b^p_{par}$ parameter \( par \) histogram discretisation (number of bins)

$b^V_{\pi}$ bin in the histogram representing variable \( V \) within which \( V_\pi \) falls

\( C(Z_t) \) misfit function (or forward model) evaluation

\( Ind \) an individual, model

\( h \) fault throw, ft (for the IC Fault Model)

\( h(d,l) \) elements of the vector representing a marginal distribution estimate \((d = 0)\) or of the matrix representing a conditional distribution estimates \((d > 0)\)

\( h_{temp}(d,l) \) elements of the vector \((d = 0)\) or of the matrix \((d > 0)\)

\( H^t(par^0) \) vector representing a marginal distribution estimate at generation \( t \)

\( H^t(par^1 - par^2) \) matrix representing a posterior conditional distribution estimates at generation \( t \)

\( H^t_{temp}(par^1 - par^2) \) so-called “temporary” matrix representing a conditional likelihood distribution estimates at generation \( t \)

\( k_{good} \) good quality sand permeability, mD (for the IC Fault Model)

\( k_{bad} \) poor quality sand permeability, mD (for the IC Fault Model)

\( MF_{min} \) misfit function of the fittest model sampled in a single run of the algorithm

\( N_{eval} \) number of evaluated individuals

\( O(t) \) the offspring of generation \( t \)

\( \hat{P}^\theta(Z) \) an estimated unconditional probability distribution

\( \hat{P}^\theta_{\pi,\omega}(Z) \) an estimated conditional probability distribution given the PDS structure defined by \((\pi, \omega)\)

\( P(par) \) estimate of the marginal probability distribution
\[ P(\text{par}_1|\text{par}_2) \] estimate of the conditional probability distribution of the parameter \( \text{par}_1 \) given the sampled value of parameter \( \text{par}_2 \)

\( t \) generation count

\( V \) variable, optimisation parameter

\( V^* \) specific value of the variable \( V \)

\( \text{WOPR} \) well oil production volume

\( \text{WOPT} \) well total oil production volume

\( \text{WWPR} \) well water production volume

\( \text{WWIR} \) well water injection volume

\( Z^i, Z^{(S)i} \) randomly sampled vector (solution string)

\( \theta_t \) the largest misfit value observed in generation \( t \)

\( \pi \) index specifying ordering of variables in PDS structure

\( \tau \) selection pressure

\( \omega \) index specifying parents of a given variable (solution string position)
Published work

Conference Proceedings


Other