

A CLASS OF SUCCESSIVE QUADRATIC PROGRAMMING METHODS
FOR FLOWSHEET OPTIMISATION

by

Chung Long, Chen B.Sc., M.Sc., DIC

October, 1988

A thesis submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy of the University of London

Department of Chemical Engineering and Chemical Technology
Imperial College of Science and Technology
London, SW7 2BY

To my parents

Abstract

Chemical process optimisation is an important area of computer aided design within chemical engineering. This work considers several strategies for the application of nonlinearly constrained optimisation methods to current flowsheet simulators, and examines the underlying mathematical techniques in detail. In particular, one of the most effective optimisation algorithms for this class of optimisation problems, the successive quadratic programming (*SQP*) method, is studied.

Firstly, a novel general formulation is presented, which gives some new insights into the *SQP* method and its relation to other optimisation techniques. Several improvements are proposed, aimed at enhancing both its global and local convergence properties. With regards to global convergence, various line search strategies are studied and a trust region algorithm is proposed. Several ways are suggested to improve the local convergence of *SQP* methods. New scaling schemes are proposed, together with strategies for dealing with inconsistent constraints and a generalised projection method for feasibility improvement. The effects of these improvements on the robustness and efficiency of the *SQP* algorithm are demonstrated with numerical results.

Secondly, the application of *SQP* methods to large scale optimisations is examined. An effective decomposition method, named the successive reduced quadratic programming (*SRQP*) method, is proposed. The method partitions the search direction at each step into two components. Each component is generated from the solution of a corresponding reduced quadratic programming subproblem. It is shown that several known algorithms are special cases of this method. In addition, suitable line search strategies, an improved Hessian update and a strategy for dealing with inconsistent constraints are also proposed.

Finally, the application of the *SQP* and *SRQP* methods within a flowsheet simulation system is studied. Particular aspects examined are the optimisation problem formulation, various optimisation strategies (including feasible path, infeasible path and infeasible path hybrid methods), the efficient evaluation of gradients, the initialisation of optimisation calculations, and parametric analysis.

Extensive numerical results are presented on examples of both mathematical and chemical engineering nature. The optimisation algorithms discussed were

implemented into a computer code and interfaced to two commercial process simulators, one of traditional sequential modular architecture, the other an equation oriented simulator, and a variety of industrial problems were solved. The numerical results show that the proposed enhancements to the basic *SQP* algorithm and the new *SRQP* method are very effective in dealing with typical large scale flowsheet optimisation problems.

Table of Contents

<i>CONTENTS</i>	<i>PAGE</i>
Abstract	i-1
Acknowledgements	i-7
List of Tables	i-9
List of Figures	i-11
List of Algorithms	i-12
Nomenclature	i-13
Abbreviations	i-16
Symbols	i-16
Chapter 1 Introduction	
1.1 Backgroud	1
1.2 Outline of the Thesis	5
Chapter 2 Optimisation Methods	
2.1 Introduction	7
2.2 Optimality Conditions	7
2.3 Optimisation Methods	10
2.4 Successive Quadratic Programming Methods	14
2.4.1 Formulation of the <i>QP</i> Subproblem and Its Relation to Other Optimisation Methods	16
2.4.2 Termination Criteria	22
2.4.3 Global Convergence	23
2.4.4 Local Convergence and the Hessian Update	39
2.5 Implementation Considerations	42
2.5.1 Choice of the Initial Hessian of the Lagrangian Function	42
2.5.2 Introducing Step Bounds and Dealing with Inconsistent Constraints	44
2.5.3 Solution of the <i>QP</i> Subproblem	48
2.5.4 Effect of Scaling	48
2.6 Complete <i>SQP</i> Algorithm	50
2.7 Numerical Experiences	51
2.8 Discussion and Conclusions	74

Table of Contents (continued)

<i>CONTENTS</i>	<i>PAGE</i>
Chapter 3 Large Scale Optimisation	
3.1 Introduction	76
3.2 Decomposition Method	79
3.2.1 Construction of Reduced <i>QP</i> Models	80
3.2.2 Termination Criteria	87
3.2.3 Global Convergence	88
3.2.4 Local Convergence and Reduced Hessian Update	92
3.3 Implementation Considerations	95
3.3.1 Formulation of the Optimisation Problem	95
3.3.2 Choice of the Initial Hessian	96
3.3.3 Choice of the Design Variables	97
3.3.4 Construction of the Projection Matrix	98
3.3.5 Strategy for Dealing With Inconsistent Constraints	99
3.3.6 Strategy for Predicting the Likely Active Inequality Constraints	100
3.3.7 Solution of the Reduced <i>QP</i> Subproblem	100
3.4 Complete <i>SRQP</i> Algorithm	100
3.5 Comparison with Other Optimisation Methods	102
3.5.1 Comparison with Feasible Direction Methods	102
3.5.2 Comparison with an Optimisation Code - <i>MINOS</i>	102
3.6 Numerical Experiences	104
3.6.1 Comparison of Reduced Models and Choice of Design Variables	105
3.6.2 Effect of the Choice of the Initial Hessian	109
3.7 Discussion and Conclusions	112

Table of Contents (continued)

<i>CONTENTS</i>	<i>PAGE</i>
Chapter 4 Flowsheet Optimisation	
4.1 Introduction	114
4.2 Flowsheet Optimisation Strategies	116
4.2.1 Feasible Path Approaches	118
4.2.2 Infeasible Path Approaches	122
4.3 Implementation of Flowsheet Optimisation	126
4.3.1 Gradient Evaluation Strategies	126
4.3.2 Initialisation of Process Variables	131
4.3.3 Selection of Tear Streams	133
4.4 Numerical Experiences	135
4.4.1 Process Optimisation with a <i>MO</i> Approach	136
4.4.2 Process Optimisation with an <i>EO</i> Approach	156
4.5 Discussion and Conclusions	171
Chapter 5 Sensitivity Analysis	
5.1 Introduction	174
5.2 Parametric Sensitivity Analysis	175
5.3 Decision Variable Sensitivity Analysis	180
5.4 Warm Start Procedures	182
5.5 Use of Simplified Models	184
5.6 Sensitivity to Model Variations	185
5.7 Numerical Experiences	187
5.8 Discussion and Conclusions	194

Table of Contents (continued)

<i>CONTENTS</i>	<i>PAGE</i>
Chapter 6 Conclusions and Future Research	
6.1 Summary	195
6.2 Contributions	198
6.3 Future Research	199
Appendices	
B.1 Characteristics of Test Problems and Numerical Results on the Mathematical Problems	202
C.1 Alkylation Process	216
C.2 Williams-Otto Process	218
D.1 Choice of the Perturbation Size	224
E.1 Efficient Factorisation	226
References	228

Acknowledgements

I would like to express my sincere gratitude to Dr. S. Macchietto for his guidance, patience and enthusiasm throughout this project.

My great appreciation goes to Professor M. J. D. Powell of University of Cambridge (UK), for kindly providing his valuable quadratic programming code *ZQPCVX*, to Professor Doctor K. Schittkowski of University of Bayreuth (FRG) for his effective optimisation code *NLPQL* and to Professor L. Dixon and Dr. M. C. Bartholomew-Biggs of Hatfield Polytechnic (UK) for part of their precious optimisation library which includes the *OPXRQP* optimisation code.

Thanks are due to the *SPEEDUP* team of Imperial College, Dr. Constantinos C. Pantelides and Mr. Lakis Liberis for providing the interface with the *SPEEDUP* simulator and test problems of a methanol process, a double effect evaporator and a simplified industrial process.

Thanks to Imperial Chemical Industries (*ICI*) for allowing me to access and to test the *SPEEDUP* model of a simplified industrial process.

Special thanks go to International British Petroleum Company, in particular to Mr. Peter Banks, Mr. Vince Paul and Dr. Bryn J. Stenhouse in the System Engineering department for a friendly environment, for allowing me to use the *GENESIS* simulator and for providing the interface and test problems.

I am grateful to members of the staff and faculty of Chemical Engineering Department at the Imperial College for a stimulating research environment and for providing several opportunities for me to attend international conferences where I learnt the state of the art in optimisation techniques. To my colleagues I also express special thanks for their friendship and valuable discussions on both philosophical and technical subject. To three exceptional friends, David Juarez-Romero, Dimitrios Gritsis, and Iqbal Mujtaba, I extend my most heartfelt appreciation.

Drs. David Juarez-Romero, Constantinos C. Pantelides, Ania Grobicki and Bryn J. Stenhouse deserve special commendation for their reading an early version of this thesis and for providing valuable suggestions.

A grant from the British Petroleum Company for the duration of this work is gratefully acknowledged.

I am profoundly indebted to my parents and brothers for their financial support and encouragement.

All errors are mine.

List of Tables

<i>TABLE</i>		<i>PAGE</i>
1.1	Reviews of Flowsheet Optimisation Based on SQP Methods	4
2.1	Major Reviews of Optimisation Methods.	11
2.2	Reviews of Trends in Optimisation Software.	11
2.3	Major Comparison of Optimisation Codes.	12
2.4	Performance Criteria.	52
2.5	Comparison of Two QP Codes.	54
2.6	Effect of the Number of Variables on the QP	55
2.7	Effect of the Number of Active Constraints on the QP	55
2.8	Effect of the Choice of the Initial Hessian.	57
2.9	Effect of Imposing the Step Bounds.	59
2.10	Comparison of Line Search and Trust Region Methods.	64
2.11	Effect of Feasibility Improvements.	67
2.12	Effect of Scaling the Objective Function	69
2.13	Comparison of Three Optimisation Codes on Process Problems.	72
2.14	Statistical Results of the Comparison of Three Optimisation Codes	73
3.1	Effect of the Choice of Design Variables on the Alkylation Process.	110
3.2	Effect of the Choice of Design Variables on the Williams-Otto Process.	110
3.3	Effect of the Choice of the Initial Reduced Hessian on the Alkylation Process.	111
3.3	Effect of the Choice of the Initial Reduced Hessian on the Williams-Otto Process.	112
4.1	Charateristics of Test Problems for a MO approach.	137
4.2	Specifications for a Simple Flash Process.	141
4.3	Specifications for a Simple Distillation Optimisation.	143
4.4	Specifications for a Refrigeration Process.	145
4.5	Specifications for the Cavett's Process.	150

List of Tables (continued)

<i>TABLE</i>		<i>PAGE</i>
4.6	Effect of the Choice of the Initial Hessian.	152
4.7	Effect of the Initialisation of Tear variables.	153
4.8	Comparison of Line Search and Trust Region Methods.	154
4.9	Comparison of Optimisation Strategies for a <i>MO</i> Approach.	155
4.10	Charateristics of Test Problems for an <i>EO</i> approach.	157
4.11	Comparison of Reduced Models.	163
4.12	Comparison of Merit Functions.	164
4.13	Comparison of Optimisation Strategies for an <i>EO</i> Approach.	165
4.14	The CPU Time Distribution on Test problem 12.	167
4.15	Performance of <i>MINOS</i> (V. 5.1).	169
5.1	Reviews of Optimisation on the Williams-Otto Process.	188
5.2	Lagrangian Multiplies to the Design Constraints.	189
5.3	Parametric Optimisation of the Williams-Otto Process.	190
5.4	Specifications for a simple Refrigeration Process.	191
5.5	Lagrangian Multiplies to the Design Constraints.	192
5.6	Parametric Optimisation on a Refrigeration Process.	193
B.1	Characteristics of the Mathematical Test Problems.	202
B.2	Descriptions of Chemical Process Test Problems.	208
B.3	Characteristics of the Chemical Process Test Problems.	209
B.4	Comparison of Three Optimisation Codes on Mathematical Problems.	210

List of Figures

<i>FIGURE</i>	<i>PAGE</i>	
2.1	Correlation of the CPU time in the QP.	56
2.2	Line Search and Trust region Methods.	62
3.1	Alkylolation Process Flowsheet.	106
3.2	Williams-Otto Process Flowsheet.	106
3.3	Comparison of Three Reduced Models.	107
3.4	Jacobian Structure of an Alkylolation Process.	109
3.5	Jacobian Structure of the Williams-Otto Process.	109
4.1	Flowsheet Module.	115
4.2	Feasible Path Approach.	120
4.3	Feasible Path Variant Approach.	121
4.4	Infeasible Path Approach.	124
4.5	Infeasible Path Hybrid Approach.	125
4.6	Simple Flash Process Flowsheet.	140
4.7	Simple Distillation Optimisation.	142
4.8	Refrigeration Process Flowsheet.	144
4.9	Complex Cryogenic Process Flowsheet.	146
4.10	Crude Oil Distillation Optimisation.	147
4.11	Catalytic Reformer Process Flowsheet.	148
4.12	Cavett's Process Flowsheet.	149
4.13	Double Effect Evaporator Process Flowsheet.	158
4.14	Methanol Process Flowsheet.	159
4.15	Flotation Process Flowsheet (a).	160
4.16	Flotation Process Flowsheet (b).	160
5.1	Simple Refrigeration Process Flowsheet.	191

List of Algorithms

<i>ALGORITHM</i>		<i>PAGE</i>
B.I	Basic SQP Algorithm.	15
B.II	Basic Watchdog Algorithm.	27
B.III	Line Search with a Feasibility Step.	33
B.IV	Line Search Followed by Feasibility Steps.	33
B.V	Trust Region Algorithm.	38
B.VI	Complete SQP Algorithm.	50
C.I	Basic Decomposition Algorithm.	79
C.II	Line Search Followed by Feasibility Steps.	92
C.III	Complete SRQP Algorithm.	101
E.I	Warm Start Procedure.	183

Nomenclature

Latins

a	Coefficient vector.
b	Vector of parameters.
c	Set of active constraints.
d_x	Search direction with respect to x .
e	Vector of extra variables.
f	Objective function.
g	Vector of inequality constraints.
h_d	Vector of design equations.
h_p	Vector of process equations or tear equations.
h	Vector of equality constraints.
$\hat{h}$	Vector of nonlinear equality constraints.
m	Number of constraints.
ma	Number of active constraints.
meq	Number of equality constraints.
mi	Number of inequality constraints.
n	Number of variables.
$\hat{n}$	Number of nonlinear variables.
$\bar{n}$	Number of linear variables.
nb	Number of parameters.
p_s	Feasibility step.
p_t	Optimality step.
r	Vector of penalty parameters.
s	Feasibility step.
u	Vector of independent variables.
v	Vector of dependent variables.
w	Vector of total variables.
x	Vector of variables.
$\hat{x}$	Vector of nonlinear variables.
$\bar{x}$	Vector of linear variables.
y	Update basis.
z	Vector of integer variables

Nomenclature (continued)

Latins

- A Coefficient matrix.
- B Approximate Hessian of the Lagrangian function.
- D Diagonal matrix.
- G Positive definite matrix.
- H Hessian approximation of the Lagrangian function.
- H_q Hessian approximation of the quadratic penalty function.
- H_e Hessian approximation of the absolute value penalty function.
- I Identity matrix.
- J Approximate Jacobian matrix.
- J_h Approximate Jacobian matrix of equality constraints.
- J_g Approximate Jacobian matrix of inequality constraints.
- L Lagrangian function.
- K_c Index set of active constraints.
- K_h Index set of active inequality constraints.
- K_g Index set of inactive constraints.
- K_e Index set of most likely active constraints.
- $\hat{O}$ Null matrix.
- P_q Quadratic penalty function.
- P_a Augmented Lagrangian function.
- P_b Modified augmented Lagrangian function.
- P_e Absolute value penalty function.
- P_l Modified Lagrangian function.
- P_m Merit function (line search objective function).
- Q Orthogonal matrix defined in equation (3.7).
- R Upper triangular matrix defined in equation (3.7).
- U_1 A matrix defined in equation (5.5).
- U_2 A matrix defined in equation (5.5).
- U_3 A matrix defined in equation (5.5).
- X Sensitivity matrix of the solution vector.
- Y Range space matrix defined in equation (3.4).
- Z Null space matrix defined in equation (3.4).

Nomenclature (continued)

Greeks

- α Step length parameter.
- β Line search parameter defined in equation (2.25).
- ϵ Tolerance defined in equation (2.23).
- λ Lagrangian multipliers for equality constraints.
- μ Lagrangian multiplier for the trust region bound.
- ν Lagrangian multipliers for inequality constraints.
- ξ Artificial variable.
- π Lagrangian multipliers for active constraints.
- Θ_a Predicted reduction of a merit function.
- Θ_p Actual reduction of a merit function.
- Λ Matrix defined in equation (3.43).
- Ω Trust region radius.

Abbreviations

<i>BFGS</i>	Broyden-Fletcher-Goldfarb-Shanno.
<i>CPU</i>	Central Processing Unit.
<i>CSTR</i>	Continuous Stirred-Tank Reactor.
<i>EO</i>	Equation Oriented.
<i>EQP</i>	Equality Quadratic Programming.
<i>FP</i>	Feasible Path.
<i>GRG</i>	Generalised Reduced Gradient.
<i>IP</i>	Infeasible Path.
<i>IPH</i>	Infeasible Path Hybrid.
<i>IQP</i>	Inequality Quadratic Programming.
<i>MINLP</i>	Mixed Integer Nonlinear Programming.
<i>MO</i>	Modular Oriented.
<i>QP</i>	Quadratic Programming.
<i>REQP</i>	Recursive Equality Quadratic Programming.
<i>RQP</i>	Recursive Quadratic Programming.
<i>SLP</i>	Successive Linear Programming.
<i>Sl₁QP</i>	Successive l_1 Quadratic Programming.
<i>SQP</i>	Successive Quadratic Programming.
<i>SOR</i>	Successive Over-Relaxation.
<i>SRQP</i>	Successive Reduced Quadratic Programming.
<i>STE</i>	Simulation Time Equivalence.
<i>SUMT</i>	Sequential Unconstrained Minimisation Technique.
<i>SVD</i>	Singular Value Decomposition.
<i>MCR</i>	Mixed Composition Refrigerant.

Symbols

$\stackrel{def}{=}$	Defined as.
$\approx$	Approximate.
$\rightarrow$	Converge.
$(\cdot)^T$	The transpose.
$ \cdot _q$	q -norm.
$\frac{\partial}{\partial x_i}$	Partial derivative operator with respect to x_i .
∇_x	Gradient operator $(\dots, \frac{\partial}{\partial x_i}, \dots)^T$.
$\nabla_x^2 P$	The $n \times n$ Hessian matrix of P whose i, j^{th} element is $\frac{\partial^2 P}{\partial x_i \partial x_j}$.
$O(\cdot)$	Of order.
$o(\cdot)$	Negligible relative to $(\cdot)$.

Chapter 1

Introduction

1.1 Background

The computer aided design of a chemical process has advanced significantly over the last two decades. Various computer programmes using flowsheeting techniques, referred to as process simulators, have been successfully written to model chemical processing units, to simulate both the steady state and the dynamic behaviour of a chemical process, and to design such processes [Westerberg and Benjamin, 1985], [Pantelides, 1988 (a)].

The successful application of computers to the analysis of chemical processes encourages engineers further to synthesize, design and operate chemical processes over a wide range of conditions in a profitable manner. This leads to the development of process optimisation which, as its name implies, is intended to select the best of these candidates to achieve one's objective and to satisfy given constraints.

Optimisation at the stage of synthesis of chemical processes, also referred to as structural optimisation, involves the choice of both the type and the number of chemical processing units and of their interconnections to satisfy the designer's objectives. Changing the design conditions for a chemical process of fixed structure results in parametric optimisation. In the course of designing a given chemical process, one should not neglect fluctuations on uncertain empirical or economic parameters. Moreover, it is also required that the process should be controllable at optimal design conditions and operable over a wide range of operation conditions. In addition, the definition of ' optimum ' often involves multiple objectives.

These areas, namely discrete optimisation for process synthesis, continuous optimisation with a given chemical process, optimisation under uncertainty, semi-infinity optimisation, dynamic optimisation and multiple objective optimisation cover the domain of process optimisation [Biegler and Grossmann, 1985]. This thesis is only concerned with optimisation of chemical processes in which the structure is fixed and the process is at steady state. This problem, however, has its own fundamental importance since it is often embedded in many of the other process optimisation classes mentioned above.

Although flowsheet optimisation is a natural and desirable approach to the design of chemical processes, it was not widely employed and accepted in the industry until the mid 1980's due to the lack of suitable tools for the solution of optimisation problems [Westerberg, 1981].

The development of process simulators with optimisation capability closely parallels the progress of the underlying theoretical optimisation methods. In the early stages, flowsheet optimisation was performed by searching through a feasible region in which the system equations within the flowsheet were satisfied while the value of the objective function was continuously improved. A simulator was used for generating feasible points. This approach is often referred to as a feasible path approach.

Optimisation methods that do not require gradients, e.g. random search methods [see Reklaitis, Ravindran and Ragsdell, 1983], were preferred due to the cost of each simulation. A number of optimisation studies were carried out with this approach on *CHESS* (a simulator developed by Motard, and Lee [1971]), [Friedman and Pinder, 1972], [Gaines and Gaddy, 1976], [Ballman and Gaddy, 1977], [Doering and Gaddy, 1980].

Although this type of direct search optimisation method has the potential to locate a global minimum, it is much too expensive. Numerous search points and extensive computational time are required before reaching an optimal solution.

This difficulty led to the consideration of optimisation methods that require gradients. These methods greatly reduce the number of searching points even though it may be expensive to generate the gradients. It was also observed that the use of (approximate) curvature information of both the objective function and system equations would result in even more efficient optimisation methods. Generalised reduced gradient methods (*GRG*) [Abadie and Carpentier, 1969], [Lasdon *et al.*, 1978] and a projection method [Sargent and Murtagh, 1973] were hence developed.

Grauer *et al.* [1979] optimised a complex benzene chlorination process by the *GRG* method. Parker and Hughes [1981] used a quadratic approximation programming method [Beale, 1967] to optimise an ammonia process on *FLOWTRAN*®, a commercial simulator developed by the Monsanto company. Sarma and Reklaitis [1982] employed the *GRG* method to optimise an ethylene oxide-glycol process.

Although the feasible path approach is easily understood, it is expensive since a flowsheet simulation is required to generate each search point.

The infeasible path approach is therefore motivated, which does not require to perform a flowsheet simulation at each optimisation iteration. An early attempt in using this approach was made by successively linearising all the system equations and the objective function. This was called the method of approximation [Griffith and Stewart, 1961]. This approach has been applied in the refinery industry with some success. However, it may fail to locate an optimal solution for nonlinear systems since no curvature information of both the objective function and system equations is used.

Optimisation methods that introduce a term penalising the violation of some equations with a high cost, e.g. penalty function methods or augmented Lagrangian function methods [cf. Gill, Murray and Wright, 1981 (a)], were also tried. Disappointing results were reported by Jung *et al.* [1971] on the Williams-Otto process [Williams and Otto, 1960], a recycle CSTR system. A similar difficulty in optimising the same process by using an augmented Lagrangian function method was also reported recently by Rangaiah [1985]. The choice of weights for penalising the violation of equations poses a serious difficulty in these methods.

More recently an optimisation method, named the successive quadratic programming (SQP) method [Wilson, 1963], [Han, 1977], [Powell, 1978 (a)], has been developed, which permits the effective solution of a medium sized flowsheet optimisation. Based on this method, various optimisation strategies on different types of simulators have been proposed, as listed in the Table 1.1.

The interesting results obtained by these authors demonstrate the great potential of the SQP methods for flowsheet optimisation.

There are, however, a number of problems with regards to the optimisation strategies and SQP methods. Conflicting results on the efficiency of various optimisation strategies [Biegler, 1985], [Biegler and Hughes, 1985] and on the initialisation of optimisation calculations [Chen and Stadtherr, 1985 (b)] have been reported. The penalty of the infeasible path approach for failing to find a solution may be great since the results at the failed position may be entirely useless. This led to develop an infeasible path hybrid strategy which captures the ideas of both the feasible path and infeasible path strategies [Kisala *et al.*, 1986], [Lang and Biegler, 1987].

Table 1.1 Reviews of Flowsheet Optimisation Based on SQP Methods

Author(s)	Simulator (Type)	Strategy
Jirapongphan [1980]	<i>ASPEN – PLUS</i> [®] (MO)	IP
Biegler and Hughes [1982]	<i>SPAD</i> (MO)	IP
Biegler and Hughes [1983]	<i>SPAD</i> (MO)	IP
Locke and Westerberg [1983]	<i>ASCEND-II</i> (EO)	IP
Kaijaluo [1984]	<i>SSPS</i> (MO)	IP
Chen and Stadtherr [1985]	<i>SIMMOD</i> (MO)	IP
Biegler and Cuthrell [1985]	<i>SPAD</i> (MO)	IP
Biegler and Hughes [1985]	<i>SPAD</i> (MO)	FP
Evans, Trevino-Lozano, Britt, and Boston [1985]	<i>ASPEN – PLUS</i> [®] (MO)	IP
Hutchison, Jackson, and Morton [1986]	<i>QUASILIN</i> (EO)	IP
Kisala, Boston, Britt, and Evans [1986]	<i>ASPEN – PLUS</i> [®] (MO)	IP,FP,IPH
Gallier and Kisala [1987]	<i>ASPEN – PLUS</i> [®] (MO)	IP,FP,IPH
Lang and Biegler [1987]	<i>FLOWTRAN</i> [®] (MO)	IP,FP,IPH
Richard and Pike [1987]	<i>DESIGN/2000</i> (MO)	IP,FP
<i>SPEEDUP</i> group [1988]	<i>SPEEDUP</i> (EO)	FP

Key :

- IP : Infeasible path approach.
- FP : Feasible path approach.
- IPH : Infeasible path hybrid approach.
- MO : Modular oriented approach.
- EO : Equation oriented approach.

The SQP method is also not without major disadvantages. Both its global and local convergence are not fully satisfactory [Fletcher, 1985], [Powell, 1985 (b)]. It is difficult to apply to large scale optimisation problems due to the excessive storage needed to store the curvature information of the objective function and constraints, and considerable computational effort is required to generate search directions [Gill *et al.*, 1985 (b)]. Kisala *et al.* [1986] showed that a decomposition method proposed by Locke *et al.* [1983] for large scale optimisation had difficulties in obtaining an optimal solution. The authors avoided this difficulty by providing a better initial condition. The main issues in the decomposition method are the

generation of search directions and both its global and local convergence properties. These have not been explored in a satisfactory way and need to be tackled in more detail.

These difficulties provide the impetus to study optimisation strategies, the optimisation methods and their applications to large scale optimisation.

The main objectives of this thesis are to present a flexible optimisation strategy which is effectively implemented with the *SQP* type algorithms, to improve the robustness of the *SQP* method and to develop a class of decomposition methods for large scale optimisation. The efficient use of information provided from the *SQP* algorithm for sensitivity analysis will also be examined. The capability of these techniques will be demonstrated through two commercial simulators on several industrial processes.

It is hoped to provide an insight into several optimisation methods and to shed some light on the development of process optimisation.

1.2 Outline of the Thesis

Chapter two considers mathematical methods for the solution of optimisation problems. In particular the study of improvements to the *SQP* method is the core of this chapter. The algorithmic components of the *SQP* method and its relationship to other optimisation methods are stressed.

A trust region algorithm with a second order correction is proposed and compared with various line search algorithms. A generalised projection method for feasibility improvements, scaling schemes and ways to deal with inconsistent constraints are also suggested. Extensive numerical results for various *SQP* algorithms are presented.

A class of decomposition methods with reduced Hessian for large scale optimisation is proposed in Chapter three. The critical roles played by various merit functions in forcing convergence and ways of dealing with inconsistent constraints are explored. In addition, feasibility improvement for this decomposition method is also examined in this chapter.

Chapter four introduces an optimisation facility to current process simulators. Several strategies are discussed for interfacing optimisation methods with process

simulators of different architectures. Ways of generating gradients and the initialisation of optimisation calculations are investigated. After comparing these optimisation strategies, recommendations are provided for the choice of a suitable optimisation strategy. Extensive results on two different types of simulators, namely, a sequential modular simulator and an equation oriented simulator, are also presented.

The effect of parameters, constraints and design variables on the optimum are studied in Chapter five. This chapter also examines the use of approximate models for processing units in order to improve the efficiency of flowsheet optimisation. To enhance the effectiveness of optimisation algorithms, so-called warm start procedures, are also presented in this chapter.

Chapter six summarises the present research and highlights promising topics for further research.

Chapter 2

Optimisation Methods

2.1 Introduction

The success of process optimisation strongly relies on the underlying optimisation techniques. In this chapter, nonlinearly constrained optimisation techniques will be studied in more detail. In particular, the successive quadratic programming (*SQP*) method, one of the most effective optimisation methods in terms of function evaluations, and its relation to other major optimisation methods are critically examined. The difficulties of the method are identified, and possible remedies derived from other methods are proposed. This chapter aims at improving and enhancing the *SQP* method.

This chapter begins with a brief review of the optimality conditions, and then the *SQP* method, its algorithm and its relation to other optimisation techniques are discussed. Several strategies are then proposed and investigated to improve its convergence properties and robustness. Algorithmic components of the method and their implementations are studied. The crucial issues are illustrated by a set of simple mathematical examples. Finally, the effectiveness of the improved method is demonstrated by applying it to extensive test problems.

2.2 Optimality Conditions

We consider a general optimisation problem expressed in the form :

$$\begin{array}{ll}
 \min_x & f(x) \\
 P \ B.I & \text{s.t. } h_i(x) = 0 \quad i = 1, \dots, meq \\
 & g_j(x) \geq 0 \quad j = 1, \dots, mi
 \end{array} \tag{2.1}$$

where

x is an n -dimensional vector of variables.

$f(x)$ is a scalar objective function.

$h_i(x)$ is an equality constraint.

$g_i(x)$ is an inequality constraint.

meq is the number of equality constraints.

mi is the number of inequality constraints.

This formulation $P B.I$ is called the original problem throughout the thesis. In this thesis we are only interested in finding a local solution (local optimum) of the problem $P B.I$ defined as a point at which the variables satisfy the constraints and no small change to the variables improves the objective function and keeps the constraints satisfied [Powell, 1980]. A point is called a feasible point when it satisfies all the constraints. A feasible direction is defined as a direction along which any step starting from a feasible point does not violate the constraints. An inequality constraint $g_i(x)$ is said to be active at a feasible point x when it reduces to an equality constraint, i.e. $g_i(x) = 0$, otherwise it is inactive. Equality constraints are thus active constraints.

The characterisation of a local solution to a constrained optimisation problem leads to the development of the optimality conditions. The importance of these optimality conditions can never be over-emphasized since most effective algorithms are based on these conditions. The following assumptions on the local solution are required in order for the optimality conditions to be valid.

Assumptions

1. The objective function $f(x)$ and constraints $h(x)$ and $g(x)$ are continuously differentiable in the range of interest.
2. The constraints are consistent, i.e. a nonempty set of feasible points exists within these constraints.
3. A local solution of the original problem exists.
4. The gradients of the active constraints are linearly independent in the range of interest (i.e. in the neighbourhood of a local solution). This assumption is often referred to as a constraint qualification.

Only some of the optimality conditions are briefly stated here. More detailed treatments of the optimality conditions can be found in text books [e.g. McCormick, 1983] and a historical view of optimality conditions is available elsewhere [Kuhn, 1976].

Based on these assumptions, the first order necessary optimality conditions can be stated.

Theorem 2.1

(*Karush-Kuhn-Tucker First Order Necessary Optimality Condition*)

If the objective function $f(x)$ and the constraints $h(x)$ and $g(x)$ satisfy the above assumptions, and x^* is a local solution of the original problem $P B.I$ then there exist parameters λ^* and ν^* such that the following conditions hold

$$\begin{aligned}
\nabla_x L(x^*, \lambda^*, \nu^*) &= \nabla_x f(x^*) - \nabla_x h(x^*)\lambda^* - \nabla_x g(x^*)\nu^* = 0 \\
h(x^*) &= 0 \\
g(x^*) &\geq 0 \\
\nu_j^* g_j(x^*) &= 0, \quad j = 1, \dots, m_i. \\
\nu_j^* &\geq 0, \quad j = 1, \dots, m_i.
\end{aligned} \tag{2.2}$$

The function $L(x, \lambda, \nu)$ which includes information about the objective function and constraints, is called the Lagrangian function. This optimality condition implies that only active constraints have influence on the objective function in the neighbourhood of a local solution x^* and no feasible directions can be found to improve the objective function at the neighbourhood of an optimal point. This implication motivates the development of a strategy to predict the set of active constraints.

Theorem 2.2

(*Karush-Kuhn-Tucker Second Order Necessary Optimality Condition*)

If the above assumptions hold, and x^* is a local solution of the original problem $P B.I$ then there exist parameters, λ^* and ν^* , such that the first order necessary condition, theorem 2.1, holds and such that the Hessian of the Lagrangian function $\nabla_x^2 L(x, \lambda, \nu)$ is positive semi-definite on the tangent subspace of the active constraints at the local solution x^* .

The tangent plane of the active constraints is defined as

$$\{ t \mid \nabla_x h(x)^T t = 0, \nabla_x g_j(x)^T t = 0, \text{ for } j \in K_h \} \tag{2.3}$$

where

$$K_h \stackrel{def}{=} \{ j \mid g_j(x) = 0, j = 1, \dots, m_i. \} \tag{2.4}$$

Theorem 2.3

(*Karush-Kuhn-Tucker Second Order Sufficient Optimality Condition*)

If the above assumptions hold, if there exist parameters λ^* and ν^* at a point x^* such that the set of equations (2.2) holds and if the Hessian of the Lagrangian function $\nabla_{xx}^2 L(x^*, \lambda^*, \nu^*)$ is positive definite on the tangent subspace of the active constraints, then the point x^* is a local solution of the original problem.

These theorems provide ways to characterise a local solution of the original problem. However, a local solution may fail to satisfy these theorems if any of the strict assumptions are violated. Ways to deal with linearly dependent constraints will be discussed later.

2.3 Optimisation Methods

A thorough survey and a complete discussion of optimisation methods are neither possible nor intended due to the complexity of the optimisation problem, the variety of optimisation methods and the considerable amount of research in this area. Instead the underlying principles behind the major methods and their relationships are stated.

In general, non-linearly constrained optimisation techniques can be classified as follows :

1. Penalty function methods.
2. Augmented Lagrangian methods (Multiplier methods).
3. Generalised reduced gradient methods and projection methods.
4. Successive quadratic programming methods.

Alternative methods such as random search methods or geometric programming methods [see e.g. Reklaitis, Ravindran and Ragsdell, 1983] will not be considered here either because they are inefficient or because they are only applicable to problems with a special structure.

Excellent reviews of these optimisation methods are listed in Table 2.1. Reviews of optimisation codes and their comparisons are listed in Table 2.2 and 2.3 respectively.

Table 2.1 Major Reviews of Optimisation Methods

Author(s)	Year
Articles	
Sargent	1980
Lasdon	1985
Conference Proceedings	
Gill and Murray (eds)	1974
Dixon, Spedicato, and Szegö (eds)	1980
Mangasarian, Meyer, and Robinson (eds)	1981
Powell (ed.)	1982 (a)
Bachem, Grötschel and Korte (eds)	1983
Boggs, Byrd, and Schnabel (eds)	1985
Schittkowski (ed)	1985
Iserles and Powell (eds)	1987
Text books	
Himmelblau	1972
Gill, Murray and Wright	1981
Bertsekas	1982
Reklaitis, Ravindran and Ragsdell	1983
McCormick	1983
Luenberger	1984
Evtushenko	1985
Edgar and Himmelblau	1988
Fletcher	1988

Table 2.2 Reviews of Trends in Optimisation Software

Author(s)	Year
Waren and Lasdon	1979
Schittkowski	1980
Reklaitis, Ravindran and Ragsdell	1983
Gill, Murray, Saunders and Wright	1984 (a)

Table 2.3 Major Comparisons of Optimisation Codes

Author(s)	Year
Colville	1968
Himmelblau	1972
Eason and Fenton	1974
Sandgren and Ragsdell	1980
Hock and Schittkowski	1981
Minkoff	1981
Mulvey (ed.)	1982
Lootsma	1985

Penalty function methods and augmented Lagrangian methods are based on the transformation of the original problem *P B.I* into an unconstrained optimisation problem or a sequence of unconstrained optimisation problems by adding penalty terms to the objective function (penalty function methods) or to the Lagrangian function (augmented Lagrangian methods). The attraction of these methods is due to the fact that unconstrained optimisation problems are relatively easy to tackle. The penalty terms prescribe a cost for the violation of constraints.

The unconstrained optimisation subproblems are constructed such that under certain conditions, a local solution of the original problem matches a local solution of the unconstrained optimisation problem (exact penalty function methods) or the limit point of a sequence of unconstrained optimisation problems. It is noted that a local solution of the original problem is a saddle point of the Lagrangian function with respect to Lagrangian multipliers and variables. The introduction of a penalty term to the Lagrangian function represents an attempt to convexify the Lagrangian function.

The required property for the construction of the penalty parameters can be found in text books [e.g. Luenberger, 1984]. The following penalty terms are often used,

$$\hat{r} \|h(x)\|_q + \hat{r}' \| \max(0, -g(x)) \|_q$$

The symbol $\|x\|_q$, a *q - norm* of an *n* - dimensional vector *x* with elements x_i , is defined as

$$\|x\|_q \stackrel{def}{=} \left\{ \sum_{i=1}^n |x_i|^q \right\}^{\frac{1}{q}}$$

and the three common values of q are $q = 1, 2$ and ∞ ; the corresponding norms are $\sum_{i=1}^n |x_i|$, $(x^T x)^{\frac{1}{2}}$, and $\max_i |x_i|$.

The success of these methods depends on the choice of the penalty parameters $\hat{r}$ and $\hat{r}'$, i.e. on the weights assigned to a violation of the constraints. In the case of penalty function methods, a heavy weight tends to first direct the points in the optimisation sequence towards the feasible region, and then to move towards the optimum point. This may result in a small search step and hence in an inefficient optimisation. Such an inefficiency can also happen with the augmented Lagrangian function methods with excessively large penalty parameters.

Generalised reduced gradient methods (GRG) [Abadie and Carpentier, 1969], [Lasdon *et al.*, 1978] and projection methods [Sargent, 1974], [Rustem, 1981] tend to reformulate the original problem into an unconstrained optimisation problem by searching through the feasible region for improving the objective function. These methods include a procedure for predicting a set of active constraints, a procedure for defining a reduced sub-space according to the active set of constraints, a procedure for projecting the objective function onto the reduced space of the active constraints and a procedure for maintaining feasibility with respect to the constraints. The difference between generalised reduced gradient methods and projection methods is in the way of projection.

Even though the performance of these two methods may be quite different, the distinction between them is rather artificial [Sargent, 1974]. Since both methods require feasible points, an inefficiency may occur in following the boundary of the active constraints. In addition, a feasible region may not exist at a certain state and precautions must be taken in selecting the set of active constraints to prevent from repeatedly adding and dropping the same constraints (' zigzagging ') or from converging to a non-optimal point (' jamming ').

Successive quadratic programming methods (SQP) [Wilson, 1963], [Han, 1977], [Powell, 1978 (a)] generate at each step a search direction by solving a quadratic programming (QP) subproblem which is an approximation of the original problem. The Hessian of the quadratic programming subproblem is an approximation to the Hessian of the Lagrangian function. A procedure is taken to ensure convergence. The major difficulties of this method are that a quadratic programming subproblem may not always be well-defined and it is not easy to ensure global convergence. Furthermore, the solution methods of general quadratic

programming subproblems is in general demanding in terms of arithmetical operations and storage. These disadvantages may become an obstacle for large scale problems.

Hock and Schittkowski [1981] showed that the *SQP* method is one of the most effective optimisation methods in terms of function evaluations. This is a desirable feature for process optimisation since the function evaluations in chemical processes are generally expensive. However, despite of the emphasis on the *SQP* methods in the current research, the idea of penalty function type methods and projection type methods does not yet seem to become obsolete. The strength of these methods may become a remedy for the weakness of the *SQP* method.

2.4 Successive Quadratic Programming Methods (*SQP*) (Recursive Quadratic Programming Methods, *RQP*)

Introduced by Wilson [1963], revived by Han [1977] and Powell [1978 (a)], and comparatively studied with other existing codes by Hock and Schittkowski [1981], the *SQP* method has been regarded as one of the most effective methods for nonlinearly constrained optimisation problems in terms of function evaluations. The method was motivated by the application of Newton-type methods to the first order optimality conditions of the original problem. By linearising the first order optimality condition, equation (2.2), of the original problem at the k^{th} iteration and ignoring the higher order terms, the resulting system is found to be the first order optimality condition of a quadratic programming (*QP*) subproblem of the form :

$$\begin{aligned}
 \min_{d_x} \quad & \nabla_x f(x_k)^T d_{x_k} + \frac{1}{2} d_{x_k}^T H(x_k) d_{x_k} \\
 \text{s.t.} \quad & h(x_k) + \nabla_x h(x_k)^T d_{x_k} = 0 \\
 & g(x_k) + \nabla_x g(x_k)^T d_{x_k} \geq 0
 \end{aligned}
 \tag{2.5}$$

QP B.I

where

- $\nabla_x f(x)$ is the gradient of the objective function.
- $\nabla_x h(x)$ is the gradient of equality constraints.
- $\nabla_x g(x)$ is the gradient of inequality constraints.
- $H(x)$ is the Hessian of the Lagrangian function.
- d_x is the search direction.

The solution of this *QP* subproblem thus provides a search direction d_x to the solution of the original problem *P B.I* and estimates of the Lagrangian

multipliers of the linearised active constraints. It is observed that as the norm of the search direction, $\|d_x\|$, approaches zero, and if the Hessian of the Lagrangian function is bounded, the first order optimality condition of this *QP* subproblem approaches that of the original problem. Thus the current estimate approaches a local solution of the original problem. This *QP* subproblem has a unique solution when the Hessian of the Lagrangian function is positive-definite and the constraints are consistent and linearly independent. However, the Hessian of the Lagrangian function is not in general known in advance. It may be expensive to obtain and it may not be positive definite. In fact, only the projected Hessian of the Lagrangian function on the active constraints at the optimal point is required to be positive (semi-)definite due to the second order optimality conditions [Theorem 2.2].

Powell [1978 (b)], [1983, (a)] showed that the Hessian of the Lagrangian function can be approximated by a positive definite symmetric matrix without destroying the convergence properties of the method. *QP B.I* can therefore be reformulated as

$$\begin{aligned}
 \min_{d_{x_k}} \quad & \nabla_x f(x_k)^T d_{x_k} + \frac{1}{2} d_{x_k}^T B_k d_{x_k} \\
 \text{s.t.} \quad & h(x_k) + \nabla_x h(x_k)^T d_{x_k} = 0 \\
 & g(x_k) + \nabla_x g(x_k)^T d_{x_k} \geq 0
 \end{aligned} \tag{2.6}$$

QP B.II

where

B_k is an approximation to the Hessian of the Lagrangian function at the k^{th} iteration.

A basic *SQP* algorithm can be established as follows,

Algorithm B.I — Basic SQP Algorithm

1. Given initial conditions.
2. Form a quadratic programming subproblem *QP B.II* and solve it for the search directions and the estimated Lagrangian multipliers.
3. Check the termination criterion. If this criterion is satisfied, then exit. Otherwise the algorithm continues.
4. Employ a procedure to ensure convergence and obtain a new estimate of the solution.
5. Update the Hessian of the Lagrangian function.
6. Go to step 2.

It is necessary to consider four important ingredients of the algorithm : the formulation of the *QP* subproblem, a convergence criterion, convergence to a local solution of optimisation problems from remote starting points (global convergence property), and the convergence rate when the current points tend to converge to a local solution of the optimisation problem (local convergence property).

2.4.1 Formulation of the *QP* Subproblem and its Relation to Other Optimisation Methods

It is interesting to note that when the Hessian of the Lagrangian function remains positive definite, the solution of the *QP* subproblem can be expressed in terms of two search directions p_s and p_t as the following expression with the assumption that the set of active constraints is known a priori.

$$d_{x_k} = p_{s_k} + p_{t_k} \quad (2.7)$$

where p_{s_k} is defined as

$$\begin{aligned} QP \ B.III \quad & \min_{p_{s_k}} \quad \frac{1}{2} p_{s_k}^T B_k p_{s_k} \\ & s.t. \quad c(x_k) + \nabla_x c(x_k)^T p_{s_k} = 0 \end{aligned} \quad (2.8)$$

where

- $c(x_k)$ is the set of active constraints, and is defined as
 $c(x) \stackrel{def}{=} \{ h_i(x) = 0, i = 1, \dots, meq, \text{ and } g_j(x) = 0, j = 1, \dots, mi \}$
- $\nabla_x c(x)$ is the gradients of the active constraints.
- B is a Hessian approximation of the Lagrangian function.

and p_{t_k} is defined as

$$\begin{aligned} QP \ B.IV \quad & \min_{p_{t_k}} \quad \nabla_x f(x_k)^T p_{t_k} + \frac{1}{2} p_{t_k}^T B_k p_{t_k} \\ & s.t. \quad \nabla_x c(x_k)^T p_{t_k} = 0 \end{aligned} \quad (2.9)$$

It is clear that the *SQP* method uses the curvature information of the Lagrangian function to predict a set of active constraints and the search directions

can be divided into two components : a step p_s that defines a step to the nearest point on the linearised constraints, and a step p_t that defines a step to the minimum of the quadratic model of the Lagrangian function in the subspace orthogonal to the constraints normals [Gill *et al.*, 1985 (b)]. The step p_s can hence be regarded as a ' feasibility step ' and the step p_t can be regarded as an ' optimality step '. This representation implies that the SQP method is related to projection methods. When the current point x_k is a feasible point, the length of the step p_s is reduced to zero. i.e.

$$\|p_s\| = 0$$

The search direction d_x generated from the SQP method is then equal to the step p_t which is the same direction as that generated from the projection method provided that the same set of active constraints are predicted and the same Hessian approximation is used. This representation will be used to develop a large scale optimisation method in Chapter 4.

A difficulty arises that QP B.II may not be well defined since the linearised constraints may be inconsistent. For example, Let the constraints,

$$(x_1 - 1)^2 + (x_2 - 1)^2 \geq 8$$

and

$$x_1 + x_2 \leq 3$$

be linearised at the point (5,5). We obtain the inequalities

$$d_{x_1} + d_{x_2} \geq -3$$

and

$$d_{x_1} + d_{x_2} \leq -7$$

which are inconsistent. This phenomenon is not uncommon in practice. This is due to the fact that the nonlinear constraints may not be adequately approximated by a set of linearised constraints. This motivates the consideration of the penalty function methods in which the search directions need not satisfy the linearised constraints. The relationship between the SQP method and the penalty function methods must be established before applying the idea of penalty function methods to the SQP algorithm.

2.4.1.1 Relation of the SQP Method to Other Optimisation Methods

We consider the applications of a class of penalty function methods with a quadratic penalty function of the form,

$$P_q(x, r, r') = f(x) + \sum_{i=1}^{meq} \frac{r_i}{2} h_i(x)^2 + \sum_{j=1}^{mi} \frac{r'_j}{2} \max(0, -g_j(x))^2 \quad (2.10)$$

to solve the original problem. In the quadratic penalty function method, a sequence of unconstrained minimisation subproblems is solved with increasing values of the penalty parameters r and r' . In each subproblem, the quadratic penalty function P_q is minimised with given penalty parameters r and r' . It can be proved that the limit points of the sequence of unconstrained optimisation subproblems will converge to a local solution of the original problem when the penalty parameters r and r' diverge to infinity. This method is well known as the sequential unconstrained minimisation technique (SUMT) [Fiacco and McCormick, 1968].

If a Newton-type method is applied to minimise the quadratic penalty function with fixed parameters r and r' in each subproblem, the resulting subproblem at the k^{th} iteration to be solved has the form of

$$P \ B.II \quad \min_{d_{x_k}} P_q(x_k, r, r') + \nabla_x P_q^T(x_k, r, r') d_{x_k} + \frac{1}{2} d_{x_k}^T \nabla_x^2 P_q(x_k, r, r') d_{x_k} \quad (2.11)$$

or alternatively,

$$\begin{aligned} P \ B.III \quad \min_{d_{x_k}} \quad & f(x_k) + \nabla_x f(x_k)^T d_{x_k} + \frac{1}{2} d_{x_k}^T H_q d_{x_k} + \\ & \sum_{i=1}^{meq} \frac{r_i}{2} (h_i(x_k) + \nabla_x h_i(x_k)^T d_{x_k})^2 + \\ & \sum_{j=1}^{mi} \frac{r'_j}{2} \left(\max(0, -(g_j(x_k) + \nabla_x g_j(x_k)^T d_{x_k})) \right)^2 \end{aligned} \quad (2.12)$$

where

$$\begin{aligned} \nabla_x P_q(x_k, r, r') = \quad & \nabla_x f(x_k) + \sum_{i=1}^{meq} r_i h_i(x_k) \nabla_x h_i(x_k) - \\ & \sum_{j \in K_k} r'_j g_j(x_k) \nabla_x g_j(x_k) \end{aligned} \quad (2.13)$$

$$\begin{aligned} \nabla_x^2 P_q(x_k, r, r') = & \nabla_x^2 f(x) + \sum_{i=1}^{meq} r_i \{ \nabla_x h_i(x_k) \nabla_x h_i(x_k)^T + h_i(x_k) \nabla_x^2 h_i(x_k) \} \\ & + \sum_{j \in K_h} r'_j \{ -\nabla_x g_j(x_k) \nabla_x g_j(x_k)^T + g_j(x_k) \nabla_x^2 g_j(x_k) \} \end{aligned} \quad (2.14)$$

$$H_q = \nabla_x^2 f(x_k) + \sum_{i=1}^{meq} r_i h_i(x_k) \nabla_x^2 h_i(x_k) + \sum_{j \in K_h} r'_j g_j(x_k) \nabla_x^2 g_j(x_k) \quad (2.15)$$

and

$$K_h \stackrel{def}{=} \{ j \mid g_j(x_k) = 0, j = 1, \dots, mi. \}$$

In fact, by the introduction of the extra variables, ε and ε' *P B.III* can be reformulated as

$$\begin{aligned} \min_{d_{x_h}, \varepsilon, \varepsilon'} \quad & \nabla_x f(x_k)^T d_{x_h} + \frac{1}{2} d_{x_h}^T H_{q_h} d_{x_h} + \sum_{i=1}^{meq} \frac{r_i}{2} \varepsilon_i^2 + \sum_{j=1}^{mi} \frac{r'_j}{2} \varepsilon_j'^2 \\ \text{s.t.} \quad & h(x_k) + \nabla_x h(x_k)^T d_{x_h} = \varepsilon \\ & g(x_k) + \nabla_x g(x_k)^T d_{x_h} \geq -\varepsilon' \\ & \varepsilon' \geq 0 \end{aligned} \quad (2.16)$$

It can be proved that when the penalty parameters r and r' diverge to infinity and the current sequence of iterates converges to an optimum of the original problem, the terms $\{ -r_i h_i(x), i = 1, \dots, meq \}$ and the terms $\{ r'_j \max(0, -g_j(x)), j = 1, \dots, mi \}$ will converge to the Lagrangian multipliers $\{ \lambda_i, i = 1, meq \}$ and $\{ \nu_j, j = 1, ..mi \}$ respectively and hence the matrix H_q will converge to the Hessian of the Lagrangian function [Fiacco and McCormick, 1968].

In addition, Broyden and Attia [1984] pointed out that the matrix H_q need not have any large eigenvalues even though the matrix $\nabla_x^2 P_q(x, r, r')$ is ill-conditioned. A similar result was given by Gould [1986].

This result suggested that the Hessian of the quadratic penalty function $\nabla_x^2 P_q(x, r, r')$ becomes ill-conditioned with excessively large penalty parameters r and r' while the matrix H_q need not be ill-conditioned.

Furthermore, the direction generated from the quadratic programming subproblem $QP\ B.V$ is the same as that generated from the quadratic programming subproblem $QP\ B.II$ when the constraints are linearly independent and consistent, the matrix H_q is an approximation to the Hessian of the Lagrangian function, and the penalty parameters diverge to infinity. This result shows the relationship between the SQP method and the penalty function method.

Comparing the QP subproblem $QP\ B.V$ with $QP\ B.II$ implies that the unconstrained penalty function P_q need not be minimised completely for a given penalty parameters r and r' . It is possible to solve instead $QP\ B.V$, at each iteration with penalty parameters r and r' given as :

$$\begin{aligned} r_i &= \frac{-\lambda_i}{h_i(x)}, & i &= 1, \dots, meq. \\ r'_j &= \frac{\nu_j}{\max(0, -g_j(x))}, & j &= 1, \dots, mi; \end{aligned} \quad (2.17)$$

or, alternatively as

$$\begin{aligned} r_i &= \frac{-\lambda_i}{\varepsilon_i}, & i &= 1, \dots, meq. \\ r'_j &= \frac{\nu_j}{\varepsilon'_j}, & j &= 1, \dots, mi. \end{aligned} \quad (2.18)$$

The QP subproblem $QP\ B.V$ is also related to the recursive equality quadratic programming method (REQP) developed by Bartholomew-Biggs [1972], [1982] since the terms $\{ \varepsilon_i, i = 1, \dots, meq \}$ and $\{ \varepsilon'_j, j = 1, \dots, mi \}$ can be replaced by the terms $\{ \frac{-\lambda_i}{r}, i = 1, \dots, meq \}$ and $\{ \frac{\nu_j}{r'}, j = 1, \dots, mi \}$ respectively. The violated constraints can be chosen as active inequality constraints and the Lagrangian multipliers can be chosen to minimise the function,

$$\begin{aligned} \min_{\lambda, \nu} \quad & \| \nabla_x f(x) - \nabla_x h(x)\lambda - \nabla_x g(x) \begin{pmatrix} \tilde{\nu} \\ \nu' \end{pmatrix} \|_2 \\ \text{s.t.} \quad & \tilde{\nu} = 0 \\ & \nu' \geq 0 \end{aligned} \quad (2.19)$$

where

- $\tilde{\nu}$ is a vector of the Lagrangian multipliers corresponding to the predicted active inequality constraints.
- ν' is a vector of the Lagrangian multipliers corresponding to the predicted inactive inequality constraints.

A similar argument can be applied to an absolute value penalty function of the form :

$$P_e = f(x) + \sum_{i=1}^{meq} r_i |h_i(x)| + \sum_{j=1}^{mi} r'_j \max(0, -g_j(x)) \quad (2.20)$$

One of difficulties in using this penalty function is that the penalty function is non-differentiable at points where a constraint is active. This non-differentiable function can be solved by using a linear programming method [Fletcher and Sainz de la Maza, 1987]. It can be proved that a local minimum point of the penalty function P_e is a local solution of the original problem when the penalty parameters r and r' are larger than the Lagrangian multipliers $|\lambda|_\infty$ and $\|\nu\|_\infty$ at the optimal points respectively. This type of penalty function is sometime called a non-differentiable exact penalty function.

Let us consider a subproblem of the form,

$$\begin{aligned}
 \min_{d_{x_k}} \quad & f(x_k) + \nabla f(x_k)^T d_{x_k} + \frac{1}{2} d_{x_k}^T H_{e_k} d_{x_k} + \\
 P.B.IV \quad & \sum_{i=1}^{meq} r_i | h_i(x_k) + \nabla_x h_i(x_k)^T d_{x_k} | + \\
 & \sum_{j=1}^{mi} r'_j \max(0, - (g_j(x_k) + \nabla_x g_j(x_k)^T d_{x_k}))
 \end{aligned} \tag{2.21}$$

With the introduction of slack variables ϵ and ϵ' the subproblem can be transformed into a QP subproblem of the form :

$$\begin{aligned}
 \min_{d_{x_k}, \epsilon, \epsilon'} \quad & \nabla_x f(x_k)^T d_{x_k} + \frac{1}{2} d_{x_k}^T H_{e_k} d_{x_k} + \sum_{i=1}^{meq} r_i \epsilon_i + \sum_{j=1}^{mi} r'_j \epsilon'_j \\
 s.t. \quad & h(x_k) + \nabla_x h(x_k)^T d_{x_k} \geq -\epsilon \\
 QP.B.VI \quad & -h(x_k) - \nabla_x h(x_k)^T d_{x_k} \geq -\epsilon \\
 & g(x_k) + \nabla_x g(x_k)^T d_{x_k} \geq -\epsilon' \\
 & \epsilon \geq 0 \\
 & \epsilon' \geq 0
 \end{aligned} \tag{2.22}$$

If the Hessian H_e is replaced by a positive definite approximation to the Hessian of the Lagrangian function, the search direction is the same as that from QP.B.II when the penalty parameters r and r' are sufficiently large and the constraints are linearly independent and consistent. In practice, the penalty parameters r and r' can be estimated from Lagrangian multipliers predicted from the previous iterations. The QP subproblem P.B.IV is related to the successive l_1 quadratic programming (Sl_1QP) method developed by Fletcher [1981], [1985].

A similar analysis can also be applied to augmented Lagrangian methods. Tapia [1977] showed that for equality constrained optimisation problems, it

is more efficient to apply quasi-Newton methods to an augmented Lagrangian function and update the Lagrangian multipliers with the current estimates than to solve a sequence of unconstrained optimisation subproblem with increasing penalty parameters and obtain a complete solution of each subproblem with a given penalty parameters and Lagrangian multipliers. This method is referred to as a diagonalised multiplier method or multiplier update method. The author also showed that this method is equivalent to the *SQP method* for a certain choice of the Lagrangian multipliers for equality constrained problems [Tapia, 1977].

A recursive equality quadratic programming method based on the augmented Lagrangian function was developed by Bartholomew-Biggs [1987].

In this section, the derivation of the *SQP* method and its relations to other optimisation methods were studied. This analysis implied that a subsequence of *QP* problems based on *QP B.III* can be used to generate a feasible point. It is noted that the *QP* subproblems *QP B.V* and *QP B.VI* but not *QP B.II* are always well defined, and can be used to deal with linearly dependent and inconsistent constraints. These ideas can be effectively implemented and will be discussed later.

2.4.2 Termination Criteria

The termination criteria provide ways to terminate the execution of an algorithm when the limit points approximate the solution of the original problem within prescribed accuracy. An optimisation algorithm can be terminated when the iterate satisfies both the constraints and the optimality conditions. Therefore the termination criterion in the *SQP* algorithm, Tol , at the k^{th} iteration may be chosen as follows

$$Tol \left(\stackrel{def}{=} \max (tol_1, tol_2, tol_3, tol_4) \right) \leq \epsilon \quad (2.23)$$

where

$$\begin{aligned} tol_1 &= \sum_{i=1}^{meq} |h_i(x_k)| + \sum_{j=1}^{mi} \max(0, -g_j(x_k)) \\ tol_2 &= \sqrt{d_{x_k}^T B_k d_{x_k}}, \\ tol_3 &= \nabla_x f(x_k)^T d_{x_k} + \sum_{i=1}^{meq} |\lambda_{ik} h_i(x_k)| + \sum_{j=1}^{mi} \nu_{jk} \max(0, -g_j(x_k)), \\ tol_4 &= \nabla_x L(x_k, \lambda_k, \nu_k)^T d_{x_k} \end{aligned}$$

The expression for tol_1 is to check whether the point satisfy the constraints, and the expressions for tol_2 , tol_3 , and tol_4 are to check whether the first order optimality conditions are satisfied.

For process optimisation problems which are often badly scaled, a relative termination criterion is preferred as follows,

$$Tol = \max(tol_1, tol_2, tol_5, tol_6) \quad (2.24)$$

where

$$tol_5 = \frac{tol_3}{\max\{\eta, |L(x_k, \lambda_k, \nu_k)|\}}$$

$$tol_6 = \frac{tol_4}{\max\{\eta, |L(x_k, \lambda_k, \nu_k)|\}}$$

for some constant η , say, 1.

2.4.3 Global Convergence

There are two ways to ensure that the iterative sequence converges to the solution of the original problem from remote starting points : line search and trust region methods.

2.4.3.1 Line Search Methods

In line search methods, a merit function (line search objective function) P_m is constructed such that it contains information about both the objective function and the constraints, and such that a local solution of the original problem is a local optimal point of the merit function P_m .

In the procedure of line searches, the current search point is regarded as a function of a step length parameter α , i.e.

$$x_{k+1} = x_k + \alpha d_k,$$

so that the merit function in turn also becomes a function of the step length parameter. During the line search, the step length is reduced until either the merit function is minimised over all the range of α (exact line searches) or until a ' sufficient decrease ' in the merit function occurs (inexact line searches). In practice, inexact line search methods are preferred since they are less expensive and since it may not be possible to perform the exact line search in some cases.

The initial step length parameter is often chosen as one. The sufficient decrease requirement may be satisfied by the Goldstein-Armijo rule :

$$0 \leq P_m(x_k, \cdot) - P_m(x_k + \alpha_l d_{x_k}, \cdot) \leq -\beta \alpha_k \nabla_x P_m(x_k, \cdot)^T d_{x_k} \quad (2.25)$$

where $0 < \beta \leq 0.5$. It is noted that the search direction does not change and only the step length is restricted during a line search. The step length parameters may be generated by a simple rule, e.g. $\{ (\frac{1}{2})^k, k = 1, \dots, nk \}$, or by a quadratic extrapolation, i.e. solving the following problem,

$$\min_{\alpha} P_m(\alpha) \quad (= \phi_1 \alpha^2 + \phi_2 \alpha + \phi_3) \quad (2.26)$$

where the parameters ϕ_1 , ϕ_2 , and ϕ_3 are obtained from solving the following equations,

$$\begin{aligned} P_m(0) &= \phi_3 \\ P_m(1) &= \phi_1 + \phi_2 + \phi_3 \\ \frac{\partial P_m}{\partial \alpha}(0) &= \phi_2 \end{aligned}$$

Detailed description of the line search procedures may be found in the text books [e.g. Gill, Murray and Wright, 1981].

It is essential to ensure that the search direction generated from the algorithm is a descent direction for P_m in order for the search procedure to succeed with $\alpha > 0$. Furthermore, the merit function should not impede the convergence rate of the algorithm. As we will discuss later, for the SQP methods a fast convergence rate can be assured only when a full step length is allowed at points near the solution. Desirable properties of merit functions for SQP methods can be found in the literature [e.g. Stoer, 1985].

Candidates for merit functions include

I. those merit functions that need only function evaluations :

1. Absolute value penalty functions.
2. Quadratic penalty functions.
3. Augmented Lagrangian functions.
4. Modified Augmented Lagrangian functions.
5. Modified Lagrangian functions.

II. those merit functions that need function evaluations and gradient evaluations.

1. Fletcher's exact differentiable penalty function [Fletcher, 1973].
2. The exact penalty function of Di Pillo and Grippo [1979], [1985], [1986].

Since merit functions that require gradient information may be prohibitively expensive for process optimisation, only those in the first group are studied here.

• Absolute Value Penalty Functions

A class of absolute value penalty functions has the form :

$$P_e(x) = f(x) + \sum_{i=1}^{meq} r_i |h_i(x)| + \sum_{j=1}^{mi} r'_j \max(0, -g_j(x)) \quad (2.20)$$

Under mild conditions, a local minimum point of the original problem is a local minimum point of this exact penalty function with penalty parameters r and r' larger than Lagrangian multipliers of the original problem at the optimum [see e.g. Gill, Murray and Wright, 1981]. Han [1977] proved that the search direction generated from the QP subproblem QP B.II, is a descent direction of this exact penalty function P_e with sufficiently large penalty parameters r and r' . Powell [1978 (a)] suggested an automatic scheme for calculating the penalty parameters :

$$\begin{aligned} r_{i_{k+1}} &= \max(|\lambda_{i_k}|, \frac{1}{2}(|\lambda_{i_k}| + r_{i_k})), \quad i = 1, \dots, meq \\ r'_{j_{k+1}} &= \max(|\nu_{j_k}|, \frac{1}{2}(|\nu_{j_k}| + r'_{j_k})), \quad j = 1, \dots, mi \end{aligned} \quad (2.27)$$

Unfortunately the use of this exact penalty function as merit function may prevent the algorithm from taking a full step even at points near a solution of the original problem, this resulting some inefficiency. Observed by Maratos [1978], this phenomenon is called the *Maratos effect*. In practice, the penalty parameters r and r' provide the balance between the objective function decrease and constraint violation, and sufficiently large penalty parameters tend to emphasize feasibility of the constraints. This may lead to slow convergence in following nonlinear constraints. This situation is more likely to occur when the current estimates are near the constraint boundary in which case the penalty terms dominate. In addition, Chamberlain [1979] showed that Powell's automatic scheme, equation (2.27), may cycle instead of converging. Several suggestions have been proposed to deal with these difficulties :

1. To use a quadratic approximation to the constraints, or to include a second order correction step to reduce constraint violations [Mayne and Polak, 1982], [Gabay, 1982], [Fletcher, 1982], [Sargent, 1982], [Fukushima, 1987], [Hoza and Stadtherr, 1987], [Maany, 1987].
2. To add a second merit function with safeguards for the acceptance of the full step [Chamberlain *et al.*, 1982].

The first approach is based on the fact that the current estimates must follow more closely the boundary of the feasible region in order to decrease the value of the penalty function with sufficiently large penalty parameters. It may be preferable to solve a subproblem

$$\begin{aligned}
 P \ B.V \quad & \min_{d_{x_h}} \nabla_x f(x_k)^T d_{x_h} + \frac{1}{2} d_{x_h}^T \nabla_{xx}^2 f(x_k) d_{x_h} \\
 & s.t. \quad h_i(x_k) + \nabla_x h_i(x_k)^T d_{x_h} + d_{x_h}^T \nabla_{xx}^2 h_i(x_k) d_{x_h} = 0 \\
 & \quad g_j(x_k) + \nabla_x g_j(x_k)^T d_{x_h} + d_{x_h}^T \nabla_{xx}^2 g_j(x_k) d_{x_h} \geq 0 \\
 & \text{for} \quad i = 1, \dots, meq \text{ and } j = 1, \dots, mi.
 \end{aligned} \tag{2.28}$$

However, this problem is much more expensive to solve as reported by Maany [1987] and storing the second order derivatives of the constraints can cause a serious problem for large scale problems. Alternatively, an additional second correction step is allowed to improve the feasibility of the constraints. This approach will be discussed in the section on feasibility improvement.

In the second approach, a full step is accepted even if it does not result in a reduction of the exact penalty function provided the value of the merit function at the previous iteration is the least that has been calculated and the value of the Lagrangian function is reduced. If a new least value of a merit function does not occur during a prescribed number of iterations, then the estimate of the optimal point is reset to the value that gave the least value of the merit function. This approach is referred to as the watchdog technique and was proposed by Chamberlain *et al.* [1982]. At the k^{th} iteration, a basic watchdog technique without restarting is described as follows : [Chen and Stadtherr, 1985]

Algorithm B.II — Basic Watchdog Algorithm

1. If $k = 0$ then set ρ_k to a large number.
2. If $P_e(x_k, r, r') < \rho_k$, then set $x_{k+1} = x_k + d_k$ if
 $L(x_k + d_k, \lambda_k, \nu_k) < L(x_k, \lambda_k, \nu_k)$.
3. Otherwise perform standard line search.
4. Set $\rho_{k+1} = \min(\rho_k, P_e(x_k, r, r'))$.

This basic watchdog technique without restart schemes used by Chen and Stadtherr [1985] and Renfro [1986] for optimising chemical processes is also employed in this study. The reason for not using restart schemes in this study is that the penalty parameters in the merit function are allowed to reduce.

• Quadratic Penalty Function

The difficulty in using a quadratic penalty function, equation (2.10), as a merit function is that a local optimal point of a quadratic penalty function is not the same as that of the original problem unless the penalty parameters go to infinity. An inefficiency can also occur when penalty parameters are excessively large. Therefore the use of quadratic penalty functions as merit functions will not be considered further.

• Augmented Lagrangian Functions

An augmented Lagrangian function takes the form :

$$P_a(x) = f(x) - \sum_{i=1}^{meq} \lambda_i^+ h_i(x) - \sum_{j=1}^{mi} \nu_j^+ g_j(x) + \frac{1}{2} \left\{ \sum_{i=1}^{meq} r_i h_i(x)^2 + \sum_{j=1}^{mi} r'_j \left[\max(0, -g_j(x) - \frac{\nu_j^+}{r'_j})^2 - \nu_j^{+2} \right] \right\} \quad (2.29)$$

With this merit function, a decision must be made to estimate the Lagrangian multipliers λ^+ and ν^+ during the line search. The multipliers λ^+ and ν^+ of the augmented Lagrangian function P_a can be regarded as variables during line searches as suggested by Yamashita [1982], Schittkowski [1981], [1983], and Gill *et al.* [1986]. In this case, it can be proved that a local solution of the original problem is a local optimal point of this type of augmented Lagrangian function when the estimated Lagrangian multipliers λ^+ and ν^+ converge to the

Lagrangian multipliers of the original problem at the optimum and the penalty parameters r and r' are sufficiently large. Yamashita [1982] chose the Lagrangian multipliers λ^+ and ν^+ at the l^{th} line search iteration as,

$$\begin{aligned}\lambda_l^+ &= \lambda_k + \Delta\lambda^+ \\ \nu_l^+ &= \nu_k + \Delta\nu^+\end{aligned}\tag{2.30}$$

The multipliers λ_k and ν_k are predicted by the QP subproblem QP B.II at the k^{th} iterations and the terms $\Delta\lambda^+$ and $\Delta\nu^+$ are supplied by users. Schittkowski [1983] and Gill *et al.* [1986] update the multipliers λ^+ and ν^+ at the l^{th} line search iteration as

$$\begin{aligned}\lambda_l^+ &= \lambda_{l-1}^+ + \alpha_{l-1} (\lambda_k - \lambda_{l-1}^+) \\ \nu_l^+ &= \nu_{l-1}^+ + \alpha_{l-1} (\nu_k - \nu_{l-1}^+)\end{aligned}\tag{2.31}$$

The parameters λ_k and ν_k are again predicted by the QP subproblem, QP B.II, at the k^{th} iteration. The difficulty with this type of merit functions is that a local solution of the original problem is merely a saddle point of the augmented Lagrangian function with respect to the variables x and multipliers λ^+ and ν^+ unless the penalty parameters are sufficiently large. A special care must be taken in the choice of the multipliers and the penalty parameters.

The advantage of using this merit function is that it allows many more iterations to accept a full step ($\alpha = 1$). However, Powell [1985 (b)] showed that this type of augmented Lagrangian function may accept an incorrect step size and result in inefficiency during the line search. In fact, for equality constrained problems the augmented Lagrangian function, equation (2.29), can be reformulated as

$$\bar{P}_a(x) = f(x) + \sum_{i=1}^{meq} \frac{r_i}{2} \left\{ \left(h_i(x) - \frac{\lambda_i^+}{r_i} \right)^2 - \frac{\lambda_i^{+2}}{r^2} \right\}\tag{2.32}$$

At the feasible point the term $\left\{ \frac{\lambda_i^+}{r_i}, i = 1, ..meq \right\}$ is dominant unless the parameters $\{r_i, i = 1, meq\}$ are excessively large and $\{\lambda_i^+, i = 1, .., meq\}$ are degenerate, and hence a full step is more likely to be accepted. However, this process may not yield a better point unless the penalty parameters r is sufficiently large. A similar argument can be applied to problems with inequality constraints.

To avoid this difficulty, the Lagrangian multipliers λ^+ and ν^+ of the augmented Lagrangian function P_a can be chosen to be functions of the variables x . For equality constrained optimisation problems, the Lagrangian multipliers for

equality constraints λ^+ may be estimated by solving the following problem [Gill and Murray, 1979],

$$\min_{\lambda^+} \|\nabla_x f(x) - \nabla_x h(x)\lambda^+ \|_2 \quad (2.33)$$

This type of augmented functions reduces to a differentiable exact penalty function proposed by Fletcher [1973]. It can be proved that a local minimum of this differentiable exact penalty function is a local solution of the original problem when the penalty parameters are sufficiently large [Fletcher, 1973]. Boggs and Tolle [1980], [1982], [1986] used this type of merit function for equality constrained problems, except that the term $h(x)^T h(x)$ of P_a was replaced by the term $h(x)^T (\nabla_x h(x)^T \nabla_x h(x))^{-1} h(x)$. A difficulty arises that the estimation of the Lagrangian multipliers λ^+ and ν^+ during a line search requires the gradients of the objective function and constraints, which is an expensive procedure. The calculation of gradients of this augmented function requires the second order derivatives of the objective function and constraints, which may not be available. Powell and Yuan [1986] used this augmented Lagrangian function as a merit function for the equality constrained problems. They proposed an approximation to the second derivatives of the Lagrangian function and developed an automatic scheme for updating penalty parameters. The authors showed that this type of differentiable merit functions may be more efficient than the exact penalty function.

For inequality constrained optimisation problems, the Lagrangian multipliers ν^+ may not be a continuous function of variables x since the set of active constraints may change, and this may cause difficulties during the line search. This type of merit functions will not be pursued further, since expensive gradient information during line searches is required.

• Modified Augmented Lagrangian Functions

A modified augmented Lagrangian function proposed by Biegler and Cuthrell [1985] has the form :

$$P_b(x) = f(x) - \sum_{i=1}^{meq} \lambda_i^+ h_i(x) - \sum_{j=1}^{mi} \nu_j^+ g_j(x) + \frac{r}{2} \left\{ \sum_{i=1}^{meq} h_i(x)^2 + \sum_{j=1}^{mi} \max(0, -g_j(x))^2 \right\} \quad (2.34)$$

For an optimisation problem with equality constraints only, this type of modified augmented Lagrangian P_b reduces to the augmented Lagrangian function

P_a . For an optimisation problem with only inequality constraints, this augmented Lagrangian function can be regarded as a penalty function of the form :

$$P_b^+ = f(x) + \sum_{j=1}^{mi} \left[r \max(0, -g_j(x)) + \nu_j^+ \right] \max(0, -g_j(x)) \quad (2.35)$$

It can be observed that a local minimum point of this penalty function is a local solution of the original problem when the penalty parameters $\{ r \max(0, -g_j(x)) + \nu_j^+, j = 1, \dots, mi \}$ are sufficiently large.

• Modified Lagrangian Function

It is noted that equality constraints in the original problem can be viewed as the following inequality constraints

$$\begin{aligned} h(x) &\geq 0 \\ -h(x) &\geq 0 \end{aligned} \quad (2.36)$$

For inequality constrained problems, Sargent [1982] suggested a modified Lagrangian function as a merit function. The modified function takes the form,

$$P_l = f(x) - \alpha \{ \lambda_1 h(x) - \lambda_2 h(x) + \nu g(x) \} \quad (2.37)$$

The advantage of this merit function is that it contains no arbitrary parameters. Sargent [1982] also showed that the use of this modified Lagrangian function does not impair local convergence. This merit function can also be efficiently implemented in conjunction with a Lemke algorithm [Sargent, 1985].

2.4.3.2 Feasibility Improvements

To circumvent difficulties with global convergence, to improve the local convergence and to deal with highly curved constraints, it is proposed to include the second order correction steps in order to follow the constraints more closely. The current estimate x at the $(k + 1)^{th}$ optimisation iteration with (nl) feasibility steps may take the form

$$x_{k+1} = x_k + \alpha d_{x_k} + \alpha^2 \sum_{l=1}^{nl} s_l \quad (2.38)$$

where s_l is the second order correction step. However, the second order correction steps $\sum_{l=1}^{nl} s_l$ may neither yield a decrease in the values of the penalty terms nor reduce the value of the objective function when the current point is far away from

a feasible region. Gabay [1982] suggested that this second order correction step be employed only when the value of the penalty terms is actually reduced. Biegler [1988] suggested that the feasibility step be obtained after the line searches. Since the feasibility step may increase the objective function, the resulting point may not improve the merit function. However, when the penalty terms for the constraints are dominant, any reduction in the values of penalty terms will result in a reduction in the merit function. This may be the case when the penalty parameters are excessively large. In this case, the current estimate x at the $(k+1)^{\text{th}}$ iteration with (nl) feasibility steps takes the form :

$$x_{k+1} = x_k + \alpha d_{x_k} + \sum_{l=1}^{nl} s_l \quad (2.39)$$

In this study, the following QP subproblem is used to obtain the feasibility improvement step s_l at the l^{th} iteration,

$$\begin{aligned} \min_{s_l} \quad & \frac{1}{2} s_l^T \underline{B}_k s_l \\ \text{s.t.} \quad & h(x^+ + s_{l-1}) + J_{h_l}^T s_l = 0 \\ & g(x^+ + s_{l-1}) + J_{g_l}^T s_l \geq 0 \end{aligned} \quad (2.40)$$

where

- $\underline{B}_k$ is a positive definite matrix at the k^{th} optimisation iteration.
- s_l is a feasibility step at the l^{th} step with $s_0 = 0$.
- J_h is a Jacobian approximation to the matrix $\nabla_x h$.
- J_g is a Jacobian approximation to the matrix $\nabla_x g$.
- $x^+ = \begin{cases} x_k + d_k & \text{when this process is made before the line search.} \\ x_k + \alpha_k d_k & \text{when this process is made after the line search.} \end{cases}$

In order to avoid the evaluation of gradients, the subsequent Jacobians J_{h_l} and J_{g_l} can be taken as $\nabla_x h(x^+)$ or $\nabla_x g(x^+)$ respectively, or these Jacobians can be updated by the Broyden update of the form,

$$J_{l+1}^T = J_l^T + \frac{(y_l^+ - J_l^T s_l) s_l^T}{s_l^T s_l} \quad (2.41)$$

where

$$\begin{aligned} J_l^T &= \begin{pmatrix} J_{h_l}^T \\ J_{g_l}^T \end{pmatrix} \\ y_l^+ &= \begin{pmatrix} h(x^+ + s_{l-1}) - h(x^+ + s_{l-2}) \\ g(x^+ + s_{l-1}) - g(x^+ + s_{l-2}) \end{pmatrix} \quad \text{for } l \geq 2 \\ J_1^T &= \begin{pmatrix} \nabla_x h(x^+)^T \\ \nabla_x g(x^+)^T \end{pmatrix} \end{aligned}$$

It can be observed that the QP has a unique solution $s_l = 0$ when the current point is feasible, i.e.

$$h(x^+ + s_{l-1}) = 0 \text{ and } g(x^+ + s_{l-1}) \geq 0.$$

Without line search, there is no guarantee that the solution of QP B.VII, will result in a reduction of the violation of the constraints. This feasibility improvement is used only when

$$\begin{aligned} \|s_1\|_2 &\leq \|d_{x_k}\|_2 \quad \text{and} \\ \|s_l\|_2 &\leq \|s_{l-1}\|_2 \quad \text{when } l \geq 2. \end{aligned} \quad (2.42)$$

The search direction d_k is predicted from the QP subproblem QP B.II. Since there is no intention of satisfying the constraints exactly, this feasibility improvement is terminated either when a prescribed maximum number of iterations is achieved or when the improvement step meets the following condition :

$$\begin{aligned} \|D_1 s_l\|_2 &\leq \epsilon \text{ Tol} \quad \text{or} \\ \|D_2 h(x_{k+1+l})\|_2 + \|D_3 (\max(0, -g(x_{k+1+l}))\|_2 &\leq \epsilon \text{ Tol} \end{aligned} \quad (2.43)$$

where

- D_1, D_2, D_3 are diagonal scaling matrices with positive elements.
- Tol is a termination criterion, equation (2.23).
- ϵ is a small number say 0.01.

The advantage of using QP B.VI is that it easily includes just the equality or the inequality constraints of interest. In addition, the matrix B provides scaling factors for the variables. In this study, the approximation to the Hessian of the Lagrangian function is used for this matrix. It is noted that the QP B.VII can be used for solving a system of equations with inequality constraints.

A different QP formulation to calculate feasibility steps was proposed by Sargent [1982].

The basic line search algorithm with feasibility improvements can be stated as follows

Algorithm B.III — Line Search with a Feasibility Step

1. Define a merit function, $P_m(x_k, \cdot)$.
2. Update parameters for the merit function if necessary.
3. Solve equation (2.40) for the feasibility step s_l .
4. If $\|s_l\| > \|d_{x_k}\|$ then Set $s_l = 0$.
5. If the merit function is an absolute value penalty function then perform Algorithm B.II.
6. Find the step length parameter α_l from equation (2.26) such that
$$P_m(x_k + \alpha_l d_{x_k} + \alpha_l^2 s_l, \cdot) - P_m(x_k, \cdot) \leq \alpha_l \beta \nabla_x P_m(x_k, \cdot)^T d_{x_k}.$$
with $\beta \leq 1$ (say 0.001).
If $\| \alpha_l d_{x_k} \|_2 \leq \epsilon \max(1, \|x\|)$ or $l \geq \max_l$ then exit.
where ϵ is a small number (say 10^{-7}) and $\max_l$ is an integer number (say 5).
7. Set $x_{k+1} = x_k + \alpha_l d_{x_k} + \alpha_l^2 s_l$.

Algorithm B.IV — Line Search Followed by Feasibility Steps

1. Define a merit function, $P_m(x_k, \cdot)$.
2. Update parameters for the merit function if necessary.
3. If the merit function is an absolute value penalty function then perform Algorithm B.II.
4. Find the step length parameter α_l from equation (2.26) such that
$$P_m(x_k + \alpha_l d_{x_k}, \cdot) - P_m(x_k, \cdot) \leq \alpha_l \beta \nabla_x P_m(x_k, \cdot)^T d_{x_k}$$
with $\beta \leq 1$ (say 0.001).
5. Solve equation (2.40) for the feasibility step s_l until the termination conditions, equations (2.42) and (2.43), are met.
6. Set $x_{k+1} = x_k + \alpha_l d_{x_k} + \sum_{l=1} s_l$.

2.4.3.3 Trust Region Methods

Trust region methods are based on the idea that the search direction generated from an approximation model is accepted as long as the approximation model adequately reflects the original problem within a certain range. In contrast to line search methods, the search direction generated from trust region methods is allowed to change during the search. Trust region methods are reviewed by Moré [1983].

Gay [1984] proposed a trust region method for linearly constrained optimisation problems. Recently this approach has been applied to equality constrained optimisation problems by many researchers [Byrd, Schnabel and Schultz, 1985], [Vardi, 1985], [Yuan, 1985 (b)], [Celis, Dennis and Tapia, 1985], applied to problems with simple bounds by Conn *et al.* [1988], and applied to both equality and inequality constrained systems by Fan *et al.* [1988]. Fletcher [1981], [1985] proposed a trust region algorithm, termed successive l_1 quadratic programming method. A trust region method with a quadratic approximation of the constraints for highly nonlinear systems was proposed by Maany [1987].

In this study, a trust region method with second order correction steps for constrained optimisation problems is presented, which is an extension to the Sl_1QP method,

With trust region methods, the QP subproblem can be formulated as follows

$$\begin{aligned}
 & \min_{d_{x_k}} \nabla_{x_k} f(x_k)^T d_{x_k} + \frac{1}{2} d_{x_k}^T B_k d_{x_k} \\
 & \text{s.t. } h(x_k) + \nabla_x h(x_k)^T d_{x_k} = 0 \\
 & \quad g(x_k) + \nabla_x g(x_k)^T d_{x_k} \geq 0 \\
 & \quad \|G_k d_{x_k}\| \leq \Omega
 \end{aligned}
 \tag{2.44}$$

where

G is a scaling matrix for the step length.

Ω is a scalar trust region radius.

The inequality constraint, $\|Gd_x\| \leq \Omega$ is called the trust region bound and defines a range for the magnitude of the step, d_x . The norm $\|\cdot\|$ may be arbitrary but generally selected as the 1-norm, 2-norm, or ∞ -norm. Since the step length is restricted within the model, the Hessian need not be positive definite for the above QP to be bounded.

When the 2-norm is employed, there is no guarantee that the resulting subproblem $QP B.VIII$ can be solved within a finite number of iterations due to the presence of a nonlinear constraint. However, in this case, $QP B.VIII$ can be

the radius Ω at the k^{th} iteration chosen as follows

$$\begin{aligned} G_k &= I \quad (\text{an identity matrix}) \\ \Omega_{ik} &= \omega_1 \|d_{k-1}\|_{\infty} \frac{\max(\omega_2, x_{ik})}{\max(\omega_3, x_{\infty})}, \quad i = 1, \dots, n \end{aligned} \quad (2.47)$$

for some positive numbers $\omega_1, \omega_2, \omega_3$. However, when the radius of the trust region is small, the search direction goes to a corner of the trust region. This step no longer becomes a projected steepest descent direction. In addition, there may be no feasible points within the radius of the trust region. Strategies for inconsistent constraints will be discussed in the section of implementation.

It is noted that subproblem *QP B.X* will reduce to *QP B.II* when the Hessian approximation B , is an approximation to the Hessian of the Lagrangian function, and when the trust region bounds are not active. If these conditions hold, it can be expected to obtain the same local convergence property as with *SQP* methods when the optimisation sequence converges to a local solution of the original problem. A local convergence property will be considered later.

Following the method proposed by Fletcher [1985], a merit function is constructed to decide the radius of the trust region Ω . The acceptance of the radius of the trust region is based on the ratio between the actual reduction on the values of the merit function for the original problem and its predicted reduction. The same merit functions used in the line search methods can also be applied here. For a merit function $P_m(x, \cdot)$, the actual reduction Θ_a , takes the form of

$$\Theta_a = P_m(x, \cdot) - P_m(x + d_x, \cdot) \quad (2.48)$$

and its predicted reduction Θ_p takes the form of

$$\Theta_p = P_m(x, \cdot) - (f(x) + P'_m(d_x, \cdot)) \quad (2.49)$$

The penalty function $P_m(x, \cdot)$ is a merit function for the original problem while $P'_m(d_x, \cdot)$ is a merit function for *QP B.II* in terms of search directions. The ratio $\{\frac{\Theta_a}{\Theta_p}\}$ is used to measure the agreement between the predicted and the actual reduction of the value of the merit function. When the ratio is less than a certain value, the radius of the trust region is reduced and *QP B.X* is re-solved with a reduced radius. On the other hand, the search direction from *QP B.X* is accepted and the radius of the trust region can be allowed to increase when this ratio is greater than a certain value.

For example, when the absolute value penalty function P_e , equation (2.20), is chosen as a merit function, the actual reduction of its value Θ_a can be expressed as

$$\Theta_a = P_e(x, r, r') - P_e(x + d_x, r, r')$$

for some penalty parameters r and r' . The predicted reduction Θ_p takes the form

$$\begin{aligned} \Theta_p = & -\nabla_x f(x)^T d_x - \frac{1}{2} d_x^T B d_x + \\ & \sum_{i=1}^{meq} r_i \left\{ |h_i(x)| - |h_i(x) + \nabla_x h_i(x)^T d_x| \right\} + \\ & \sum_{j=1}^{mi} r'_j \left\{ \max(0, -g_j(x)) - \max(0, -g_j(x) - \nabla_x g_j(x)^T d_x) \right\} \end{aligned} \quad (2.50)$$

for some penalty parameters r and r' .

However, Yuan [1985 (b)] and Powell [1986 (a)] showed that the inactivity of the trust region bounds may not be achieved eventually when an absolute value penalty function P_e is used as merit function and an inefficiency similar to the *Maratos effect* may occur. Also when the radius is small, the search direction tends to be the corner of a box formed by the trust region bounds and the constraints tend to become inconsistent. To alleviate these difficulties, a lower bound for the radius of the trust region is imposed. When a radius of the trust region is lower than this lower bound, the feasibility improvements as described earlier are performed.

Yuan [1985 (b)] suggested using a differentiable exact penalty function of Fletcher [1973] as a merit function.

A basic algorithm for the trust region can now be stated as follows.

Algorithm B.V — Trust Region Algorithm

1. Define a merit function $P_m(x_k, \cdot)$.

Set constants $\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, \omega_7, \omega_8, \omega_9$ such that

$0 \leq \omega_4 \leq \omega_5 \leq \omega_6 \leq \omega_7 \leq 1$ and $0 \leq \omega_8 \leq 1 \leq \omega_9$.

Set $\Omega_{i0} = \omega_1 \max \{ \omega_2, |x_{i0}| \}$, $i = 1, \dots, n$.

In this study, these values are chosen as follows

$$\begin{aligned} (\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, \omega_7, \omega_8, \omega_9) = \\ (10^3, 1., 1., 0., 0.001, 0.01, 0.75, 0.5, 2.) \end{aligned}$$

These values mildly affect the algorithm in practice.

2. Calculate the actual reduction Θ_a and the predicted reduction Θ_p on the values of the merit function P_m and their ratios Θ_R . i.e.

$$\begin{aligned} \Theta_a &= P_m(x_k, \cdot) - P_m(x_k + d_k, \cdot) \\ \Theta_p &= P_m(x_k, \cdot) - (f(x) + P_m(d_k, \cdot)) \\ \Theta_R &= \frac{\Theta_a}{\Theta_p} \end{aligned}$$

3. Accept the search direction d_{x_k} generated from a QP subproblem and update the variables when the ratio is no less than ω_4 . i.e.

$$x_{k+1} = \begin{cases} x_k + d_{x_k}, & \text{if } \Theta_R \geq \omega_4; \\ x_k, & \text{otherwise.} \end{cases} \quad (2.51)$$

4. Revise the radii of the trust region as follows :

$$\Omega'_{k+1} = \begin{cases} \omega_8 \|d_k\|_\infty & \text{if } \omega_5 \leq \Theta_R \leq \omega_6; \\ \|d_k\|_\infty & \text{if } \omega_6 \leq \Theta_R \leq \omega_7; \\ \omega_9 \|d_k\|_\infty & \text{if } \Theta_R \geq \omega_7. \end{cases} \quad (2.52)$$

5. Set

$$\Omega_{i, k+1} = \Omega'_{k+1} \frac{\max\{\omega_2, |x_{ik}|\}}{\max\{\omega_3, \|x_k\|_\infty\}}, \quad i = 1, \dots, n.$$

6. Perform feasibility steps when Ω_{k+1} or Θ_R is less than ω_5 .

2.4.4 Local Convergence and the Hessian Update

Local convergence is concerned with the rate of convergence of an algorithm when the current estimates converge to a local solution of the original problem.

Robinson [1974] showed that the *QP* subproblem in the *SQP* method predicts a correct active set of constraints in some neighbourhood of the solution when the Hessian of the *QP* subproblem is the Hessian of the Lagrangian function.

The *SQP* method is *Q*-superlinearly convergent when the estimates converge to a local solution of the original problem, the Hessian approximation of the Lagrangian function is bounded and converges to the true Hessian of the Lagrangian function at the optimal points and a full step generated from the subproblem *QP B.II* is taken [Powell, 1978 (b)], [Garcia-Palomares and Mangasarian, 1976].

The method is two-step *Q*-superlinearly convergent when the Hessian approximation of the Lagrangian function is bounded, the projected Hessian approximation of the Lagrangian function on the tangent plane of the active constraints converges to the projected Hessian of the Lagrangian function at the optimal point and a full step is taken [Powell, 1978 (b)], [Boggs, Tolle and Wang, 1982], [Powell, 1983 (a)], [Coleman and Conn, 1984], [Nocedal and Overton, 1985], [Wright, 1986], [Stoer and Tapia, 1987], [Fontecilla, Steihaug and Tapia, 1987]. Mayne and Polak [1978], Bertsekas [1982] and Gabay [1982] showed that the *SQP* methods with the second order correction are *Q*-superlinearly convergent.

An algorithm is said to be *Q*-superlinearly convergent when the following property holds :

$$\lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0 \quad (2.53)$$

and is said to be *Q*-two-step-superlinearly convergent when

$$\lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x^*\|}{\|x_{k-1} - x^*\|} = 0. \quad (2.54)$$

Even though in practice this convergence behaviour may occur only in the last few iterations, it is of importance and it implies that as long as current estimates tend to converge to a local solution of the original problem, the algorithm will converge to a prescribed accuracy within a finite number of iterations. The requirement of the full step length imposes an extra condition for the choice of the global convergence schemes. These convergence results imply that the convergence rate

is related to the (projected) Hessian approximation of the Lagrangian function and this will be examined next.

2.4.4.1 Update of the Hessian of the Lagrangian Function

It is usual to generate the Hessian approximation B_{k+1} to satisfy the following conditions,

$$\begin{aligned}
B_{k+1} d_k &= y_k \\
&= \nabla_x L(x_{k+1}, \lambda_{k+1}, \nu_{k+1}) - \nabla_x L(x_k, \lambda_{k+1}, \nu_{k+1}) \\
&= \{ \nabla_x f(x_{k+1}) - \nabla_x h(x_{k+1})\lambda_{k+1} - \nabla_x g(x_{k+1})\nu_{k+1} \} - \\
&\quad \{ \nabla_x f(x_k) - \nabla_x h(x_k)\lambda_{k+1} - \nabla_x g(x_k)\nu_{k+1} \} \quad (2.55)
\end{aligned}$$

and

$$B_{k+1}^T = B_{k+1}$$

Among the various update schemes proposed in the literature, the superiority of the *BFGS* (Broyden-Fletcher-Goldfarb-Shanno) update is now accepted [Gill, Murray and Wright, 1981 (a)], [Powell, 1986 (b)]. The *BFGS* update has the form :

$$B_{k+1} = B_k - \frac{B_k d_k d_k^T B_k}{d_k^T B_k d_k} + \frac{\hat{y}_k \hat{y}_k^T}{\hat{y}_k^T d_k} \quad (2.56)$$

where $\hat{y}_k$ is a update basis and can be defined as y_k in equation (2.55). The *BFGS* update has the property of hereditary positive definiteness when the previous matrix is positive definite and

$$\hat{y}_k^T d_k \geq 0 \quad (2.57)$$

However, the true Hessian of the Lagrangian function may not be positive definite. The condition, inequality (2.57), may not be satisfied even when the *SQP* algorithm starts with a positive definite Hessian. There are many ways to tackle such difficulties.

1. No update or restart when the Hessian approximation is not positive definite.
2. Modify the update basis $\hat{y}_k$ to satisfy the condition, inequality (2.57) [Powell, 1978 (a)], [Pantoja, 1984].
3. Modify the update formula whenever the Hessian is not positive definite [Bartholemew-Biggs, 1985].

Our numerical experience showed that the no update and restart scheme may impair convergence rate since in some cases the Hessian may hardly be updated or always be restarted. Powell [1978 (a)] suggested that the basis $\hat{y}_k$ be defined as

$$\hat{y}_k = \theta_k y_k + (1 - \theta) B_k d_k \quad (2.58)$$

The parameter θ can be chosen such that the following condition holds,

$$d_k^T \hat{y}_k = \sigma d_k^T B_k d_k \quad (2.59)$$

for some positive constant σ . Substituting equation (2.58) to equation (2.59) and solving it for θ yields

$$\theta = \begin{cases} 1, & \text{if } d_k^T y_k \geq \sigma d_k^T B_k d_k; \\ \frac{(1-\sigma)d_k^T B_k d_k}{d_k^T B_k d_k - d_k^T y_k}, & \text{if } d_k^T y_k < \sigma d_k^T B_k d_k. \end{cases} \quad (2.60)$$

where σ is a positive number less than 1 (say 0.1 or 0.2). Although this modification performs well in practice, Pantoja [1984], Powell [1985], Bartholomew-Biggs [1985] and Biegler and Cuthrell [1985] showed that it may yield an ill-conditioned or even an unbounded Hessian approximation. Pantoja [1984] proposed to replace the basis $\hat{y}_k$ by y_k^+ of the form :

$$y_k^+ = y_k + \left\{ \sigma' - \frac{y_k^T d_k}{d_k^T d_k} \right\} d_k \quad (2.61)$$

for some constant σ' .

Bartholomew-Biggs [1985] showed that this modification has no great advantages over the Powell's modification. In our experience, it is difficult to choose a suitable constant σ' for the equation (2.61). Biegler and Cuthrell [1985] suggested that the optimisation problem be rescaled when the condition number of the Hessian approximation becomes too high. The authors suggested to use the inverse of the *BFGS* update in order to obtain the inverse of the Hessian approximation for the calculation of the condition number. This strategy requires extra storage for the inverse of the Hessian approximation. In fact, the inverse of the Hessian approximation is implicitly available within the quadratic programming routine.

Bartholomew-Biggs [1985] also suggested to employ the Broyden-Fletcher class of rank-two updated formulae instead of the *BFGS* update formulae when the Hessian is not positive definite.

Since variables in a chemical process have their own physical meaning and the order of magnitude is often known, the choice of an initial Hessian can be based on the physical magnitude of the variables. The initial Hessian B_0 can therefore be taken as a diagonal matrix with the diagonal elements $\{(B_0)_{ii}, i = 1, \dots, n\}$ chosen as follows :

Initial Hessian Scheme I

$$(B_0)_{ii} = 1 \quad (2.64)$$

Initial Hessian Scheme II

$$(B_0)_{ii} = \begin{cases} \frac{1}{10^{(\text{integer}(\log_{10}|x_i^+|))}} & |x_i^+| > \epsilon; \\ \epsilon & |x_i^+| \leq \epsilon. \end{cases} \quad (2.65)$$

Initial Hessian Scheme III

$$(B_0)_{ii} = \max\left(\epsilon, \left| \frac{\partial f(x_0)}{\partial x_i} \right| \frac{1}{x_i^+} \right) \quad (2.66)$$

The values x^+ are chosen such that they reflect the physical magnitude of process variables x . They can be selected as the initial estimates of the variables when reasonable initial estimates are available. When reasonable initial estimates are unavailable, they can be based on the physical bounds of the variables.

2.5.2 Introducing Step Bounds and Dealing with Inconsistent Constraints

Chen and Stadtherr [1984] and Kisala *et al.* [1987] showed that the QP subproblem might produce a very large step length in the first few iterations and that this step may be accepted by the algorithm. This phenomenon can be explained by the fact that the algorithm may produce an incorrect search direction due to the poor Hessian approximation and accept an inappropriate step length due to the inadequate initial penalty parameters. Chen and Stadtherr [1984] restricted the step length by the following factor in the first three iterations,

$$\alpha_k^+ = \min \left\{ 1, \eta \frac{\max(\|x\|_\infty, 10)}{\|d_{xk}\|_\infty} \right\} \quad (2.67)$$

for some constant η .

In fact, this phenomenon can also occur even when the iterates are near the solution. In this study, the step bounds are built directly into the QP and the QP B.II[†] is reformulated as,

$$\begin{aligned} \min_{d_{xk}} \quad & \nabla_x f(x_k)^T d_{xk} + \frac{1}{2} d_{xk}^T B_k d_{xk} \\ \text{s.t.} \quad & h(x_k) + \nabla_x h(x_k)^T d_{xk} = 0 \\ & g(x_k) + \nabla_x g(x_k)^T d_{xk} \geq 0 \\ & (d_{k\ell})_i \leq (d_{xk})_i \leq (d_{kv})_i \quad i = 1, \dots, n. \end{aligned} \quad (2.68)$$

where

$$\begin{aligned} (d_{k\ell})_i &= \begin{cases} -\varsigma_1 \max(\varsigma_2, |x_{ik}|) & \text{for line search methods;} \\ -\varsigma_1 \|d_{xk-1}\|_\infty \frac{\max(\varsigma_2, |x_{ik}|)}{\max(\varsigma_2, \|x_k\|_\infty)} & \text{for trust region methods} \end{cases} \\ (d_{kv})_i &= \begin{cases} \varsigma_1 \max(\varsigma_2, |x_{ik}|) & \text{for line searches;} \\ \varsigma_1 \|d_{xk-1}\|_\infty \frac{\max(\varsigma_2, |x_{ik}|)}{\max(\varsigma_2, \|x_k\|_\infty)} & \text{for trust region methods.} \end{cases} \end{aligned} \quad (2.69)$$

ς_1 is a positive number.

ς_2 is a positive number to prevent the step from becoming too small.

In this study, the values of ς_1 and ς_2 are chosen as 10^3 and 1 respectively unless specified. A similar idea was recently proposed by Kisala *et al.* [1987] in the context of line searches.

In addition, simple bounds on the variables are dealt with directly to ensure that they are always satisfied, i.e. the following bounds are used,

$$\max(x_l - x, d_{xl}) \leq d_x \leq \min(x_u - x, d_{xu}) \quad (2.70)$$

This procedure is helpful for problems with ' hard ' constraints which are not allowed to be violated. This procedure is also useful to prevent the SQP algorithm from divergence in some cases.

Another difficulty arises since subproblem QP B.II may not be well defined since linearised constraints may not be consistent even though the original constraints are consistent. Powell [1978 (a)] suggested the introduction of an extra variable ξ into subproblem QP B.XI . The resulting QP subproblem can be formulated as follows,

$$\begin{aligned} \min_{d_{x_k}, \xi} \quad & \nabla_x f(x_k)^T d_{x_k} + \frac{1}{2} d_{x_k}^T B_k d_{x_k} - \varphi \xi \\ \text{s.t.} \quad & \xi h(x_k) + \nabla_x h(x_k)^T d_{x_k} = 0 \\ & \xi g(x_k) + \nabla_x g(x_k)^T d_{x_k} \geq 0 \\ & d_{x_l} \leq d_{x_k} \leq d_{x_u} \end{aligned} \quad (2.71)$$

where

$$\xi = \begin{cases} 1, & \text{if } g(x_k) \geq 0; \\ \xi, & \text{if } g(x_k) \leq 0. \end{cases}$$

$$0 \leq \xi \leq 1$$

φ is a large number say 10^6 .

It is assumed that the original constraints are inconsistent when this modified quadratic programming problem has the only feasible solution $d_{x_k} = 0$ and $\xi = 0$. This assumption is true only when constraints are linear. Alternatively, the QP subproblem may contain only some of the ' worst violated ' constraints; i.e. the inconsistent constraints are ignored [Pshenichny, 1970], [Belegundu and Arora, 1984], [Lim and Arora, 1986], [Pshenichny, 1987], [Tseng and Arora, 1988]. The advantage of this approach is that gradients of only a few constraints are needed. However, an iterative procedure is required in order to delete inconsistent constraints. Sahba [1987] proposed to solve a linear program which ensures existence of a search vector. In fact, it is possible to apply penalty function methods for dealing with inconsistent constraints. Many approaches based on this strategy have been proposed [Schittkowski, 1983], [Tone, 1983] and [Gill et al., 1985 (a)].

In this study, the QP subproblem is reformulated as

$$\begin{aligned}
 & \min_{d_{x_k}} \nabla_x f(x_k)^T d_{x_k} + \frac{1}{2} d_{x_k}^T B_k d_{x_k} + \\
 & \quad \bar{\varphi}' \|h(x_k) + \nabla_x h(x_k)^T d_{x_k}\|_2^2 + \\
 & \quad \varphi_1 \max(0, -g(x_k) - \nabla_x g(x_k)^T d_{x_k}) \\
 & \quad s.t. \quad d_{x_\ell} \leq d_{x_k} \leq d_{x_u}
 \end{aligned} \tag{2.72}$$

From the property of the penalty function, it can be observed that the search directions, d_{x_k} generated from subproblem $QP\ B.XIII$ are the same as those generated from the QP subproblem $QP\ B.XI$ when the constraints are consistent, the penalty parameter $\bar{\varphi}'$ diverges to infinity and the penalty parameter φ_1 is sufficiently large.

In fact, a more convenient form is obtained by transforming the above QP subproblem into the following QP subproblem.

$$\begin{aligned}
 & \min_{d_{x_k}, \xi'} \nabla_x f(x_k)^T d_{x_k} + \frac{1}{2} d_{x_k}^T B_k d_{x_k} + \\
 & \quad \sum_{i=1}^{meq} \bar{\varphi}_i (h_i(x_k) + \nabla_x h_i(x_k)^T d_{x_k})^2 + \varphi_1 \xi' \\
 & \quad s.t. \quad g_j(x_k) + \nabla_x g_j(x_k)^T d_{x_k} \geq -\xi', \quad j = 1, \dots, mi. \\
 & \quad \quad d_{x_\ell} \leq d_{x_k} \leq d_{x_u} \\
 & \quad \quad \xi' \geq 0
 \end{aligned} \tag{2.73}$$

for some constants $\bar{\varphi}_i$ and φ_1 .

The constraints are assumed inconsistent when the extra variable ξ' is unacceptably large, say 10^6 . This formulation, $QP\ B.XIV$, also provides a way to deal with linearly dependent equality constraints and is used in this work.

To alleviate the difficulty in the choice of the penalty parameters φ to increase the efficiency of the algorithm and to avoid unnecessary ill-conditioning of the QP subproblem, these modifications are used only when constraints are actually found to be inconsistent. Thus the QP B.XIV is used only when Powell's modification fails.

With the formulation QP B.XIV the Lagrangian multipliers λ for the equality constraints cannot be calculated directly. However, they can be estimated in the following ways :

$$\lambda = \min_x \|\nabla_x f(x) - \nabla h(x)\lambda - \nabla_x g(x)\nu - \nu_2 + \nu_3\|_2 \quad (2.74)$$

or

$$\lambda_i = -\bar{\varphi}_i h_i(x) \quad i = 1, \dots, meq. \quad (2.75)$$

where

- ν is a vector of the Lagrangian multipliers for the inequality constraints, $\{g_j(x_k) + \nabla_x g_j(x_k)^T d_{x_k} \geq -\xi', \quad j = 1, \dots, mi.\}$
- ν_2 is a vector of the Lagrangian multipliers for the simple lower bounds, $d_x - d_{x\ell} \geq 0$
- ν_3 is a vector of the Lagrangian multipliers for the simple upper bounds, $d_{xu} - d_x \geq 0$

The Lagrangian multipliers ν , ν_2 , and ν_3 are obtained from the solution of the subproblem QP B.XIV. However, in this study, the Lagrangian multipliers for the equality constraints λ are obtained from the previous iteration since the subproblem QP B.XIV is solved only when constraints are inconsistent. The choice of penalty parameter φ_1 is based on the previous estimates of the Lagrangian multipliers for the inequality constraints, i.e.

$$\varphi_{1k} = \max\{(\nu)_i |_{\infty}, i = 1, \dots, k-1\} \quad (2.76)$$

2.5.3 Solution of the QP Subproblem

The efficiency of *SQP* methods strongly depend on the efficiency of the underlying *QP* methods. Reviews of *QP* methods can be found elsewhere [e.g. Gill, Murray and Wrigt, 1981 (a)], [Pang, 1983]. In general, solution methods of a convex *QP* problem may be categorised as primal methods and dual methods. The former methods require a feasible starting point and keep all the iterates feasible with respect to all constraints while changing the predicted set of active constraints until the correct active set is located. The method of Fletcher [1971] belongs to this category.

The latter methods may require an initial estimate of the Lagrangian multiplier with the correct sign and maintain the Lagrangian multipliers with the correct sign while moving the iterates to the feasible region. Based on the dual method, Goldfarb and Idnani [1983] proposed an algorithm for convex quadratic programming problems. These methods can be implemented by transforming the original quadratic programming problem into a linear complementary problem and performing simplex type pivots on basis matrices or tableaux (e.g. Lemke's algorithm [Lemke, 1962]). Efficient versions of the Lemke's algorithm were proposed by Sargent [1978].

Comparing a *QP* subroutine, *ZQPCVX* , [Powell, 1983 (c)] that employs the algorithm of Goldfarb and Idnani, with two widely available *QP* codes that realise the primal feasible point methods, namely *QPSOL* [Gill *et al.*, 1983 (a)] and *VE02A* [Fletcher, 1970], Powell [1985 (a)] concluded that the algorithm of Goldfarb and Idnani is very stable in practice for convex quadratic programming and the performance of the *ZQPCVX* code is comparable to that of the *QPSOL* code. It is noted that the code *VE02A* is able to deal with an indefinite *QP* while the code *ZQPCVX* is only capable of solving a convex *QP*. The *ZQPCVX* code is used for this study with a modification to deal with the simple bounds directly.

2.5.4 Effect of Scaling

There is no doubt that proper scaling of the problem can dramatically improve the efficiency and accuracy of optimisation methods. *SQP* methods are theoretically invariant under scaling of constraints and/or variables even though this nice property can hardly be achieved in practice due to rounding errors and the heuristic parameters within the algorithm. This is because both the *QP* subproblem and the *BFGS* update are invariant under scaling of constraints and/or variables.

Biegler and Cuthrell [1985] gave some reasons why *SQP* methods can be influenced by the scale of the variables and the constraints and suggested a scaling scheme. However, the scale of the objective function or the choice of the initial Hessian can affect the performance of the *SQP* methods. A choice of the initial Hessian has been discussed in the previous section.

In process optimisation, variables and constraints may have their own physical meaning. In practice, it is difficult to devise a suitable automatic scaling procedure for process optimisation without understanding the physical meaning of variables and constraints. The physical meaning of the variables may be lost with improper scaling procedures, e.g. the molar fraction may be scaled to be between 10 and 100. For nonlinear optimisation problems, ill-conditioning may occur during the calculations due to nonlinearity and noise of the functions even when functions are well scaled initially.

It can be observed that for equality constrained problems, scaling the objective function is equivalent to scaling the ' optimality step ' p_t , i.e. the search direction becomes

$$d_x = \gamma p_t + p_s$$

when the objective function is scaled by γ . For inequality constrained problems, scaling the objective function also influences in identification of active constraints. The choice of a suitable scale factor for the objective function depends on the gradients of the objective function and the curvature information of the Lagrangian function.

In practice, the objective function is required to be scaled when it is insensitive to the variables or when its norm is too large (say larger than 10^6). In this study, the objective function will not be scaled unless specified, since the step bounds are imposed and the initial Hessian schemes are proposed to reduce the risk of either an excessively large or a tiny ' optimality step '.

2.6 Complete SQP Algorithm

A complete SQP algorithm can be stated as follows,

Algorithm B.VI — Complete SQP Algorithm

```
Begin  < Initialisation Section >
  Given  an initial estimate  $x_0$  and a tolerance  $\epsilon$ .
  Given  maximum number of iterations  $max_f$  allowed.
  Given  maximum feasibility steps  $max_l$  allowed.
  Choose a merit function as either equation (2.20) or (2.29).
  Choose either a line search method or a trust region method.
  Choose an initial Hessian as equation (2.64), (2.65) or (2.66).
  Set  $k = 0$ .
End
Repeat
  Set  $k = k + 1$ .
  Begin  < Construct and Solve QP >
    Solve QP B.XI
    If Failed Then solve QP B.XII or QP XIV.
  End
  Begin  < Check Convergence Criteria >
    If inequality (2.23) is satisfied Then Exit ( Solution Found ).
  End
  If the Line Search Method is requested Then
    Begin  < Perform Line Search >
      If a Feasibility step before line search is requested
        Then Perform Algorithm B.III.
      If Feasibility steps after line search is requested
        Then Perform Algorithm B.IV.
    End
  Else_If the Trust Region Method is requested Then
    Begin  < Perform Trust Region >
      Perform Algorithm B.V.
    End
  End_If
  Begin  < Update the Hessian >
    Equation (2.55), (2.56), (2.58), and (2.60).
  End
Until  $k \geq max_f$ .
```

2.7 Numerical Experiences

A test program was written to assess the performance of the proposed *SQP* algorithms. The test program is highly structured and modular, each step of the basic *SQP* algorithm being implemented as a module (subroutine). In this way the test program is easily adapted and many algorithms and alternative options for each step of the algorithm can be tested within the same program. Therefore a more meaningful comparative study can be performed. In addition, the program is implemented in a reverse communication structure, i.e. the optimisation routine returns to the calling program whenever function and/or derivative values are required. This feature makes it easy to interface with other programs, e.g. a flowsheet simulator, since no fixed structure of the function and derivatives is assumed. The test program also provides options to calculate derivatives by either two point finite difference or central difference schemes.

The objectives of this study are to understand and examine the proposed algorithmic ideas, to identify areas where further research on the *SQP* algorithm may be required and to assess the robustness and efficiency of the present *SQP* algorithm in view of further application on flowsheet optimisation.

Two types of test problems are used for this study, namely, mathematical problems and chemical process problems. The characteristics of these test problems are listed in appendix B.1. All test problems selected from the literature have been studied by many researchers, and hence a comparative study can be performed.

Many measures of merit have been proposed as the basis of comparison [see e.g. Reklaitis, Ravindran and Ragsdell, 1983], [Lootsma, 1985]. Performance criteria in this study, summarised in Table 2.4, are based on the robustness, accuracy achieved, and efficiency. The number of gradient and function evaluations are used as indicators of the efficiency of the algorithm since these are in general the most expensive part to calculate in process optimisation. It is noted that all these criteria depend on the types of available machine and compilers, the machine precision, the tuning parameters of the optimisation routine and the way of coding the algorithms. Two types of machine are used in this study, *CDC Cyber 180-855* (machine precision 3.55×10^{-15}) and *CDC Cyber 930* (machine precision 7.105×10^{-15}). The tuning parameters are set to the default values and remain unchanged throughout this study unless explicitly stated. Reasons for the failure are examined in order to know how and why the failure occurs.

Table 2.4 Performance Criteria

Criteria	Measures of merit
Robustness	Number of problems solved
Accuracy	Terminal point accuracy achieved
Efficiency	Number of gradient evaluations
	Number of function evaluations
	CPU time

The difficulties in evaluating individual algorithmic components are that a combinatorial problem is involved, e.g. each component may have many alternatives, and furthermore these components may interact, e.g. the performance of the Hessian update depends on the line search procedure. The components of the test algorithm are set to default and only the component of interest at a time is changed when the component alternative is studied.

The default options are

1. The initial Hessian is chosen as an identity matrix.
2. The relative step bounds are 10^3 except for the test problem C3 where 0.1 is employed.
2. The line search method without feasibility improvement and the absolute value penalty function with the watchdog technique (Algorithm B.II) are used.

The gradients for all test problems are generated by the two point finite difference scheme except for the test problems C11, C12, and C13 where the central difference is used. Function evaluations required for generating the gradients are not counted in the statistics presented. However, they are included in the *CPU* time statistics.

The standard starting points cited in the literature are used for all test problems unless specified.

We start by studying the solution methods of the *QP* subproblem since the robustness and efficiency of the *SQP* methods strongly depend on this. The effect

of the initial Hessian, step bounds, line search and trust region methods, feasibility improvements, and the objective function scaling are then examined. The effect of each algorithmic component is illustrated by using a simple mathematical problem and process problems. The behaviour of the proposed algorithm on special cases is also studied. In addition, the performance of current effective optimisation codes are presented to study the performance of the present *SQP* and the related *REQP* methods.

2.7.1 On the Solution of the *QP* Subproblem

Two *QP* subroutines, namely, *ZQPCVX* [Powell, 1983 (c)] and *VE02A* [Fletcher, 1970], are studied. A *SQP* code termed *VF02AD*, developed by Powell [Powell, 1978] and modified by Morison [Morison, 1984], is used.

The selected test problems were representative problems where many additional constraints will become active.

Table 2.5 shows that most of the *CPU* time is spent on the solution of the *QP* subproblem (average 70 %) for these test problems for which the cost of function and gradient evaluations is negligible. In this case, the *SQP* algorithm strongly depends on the efficiency of the *QP* algorithm. The *QP* subroutine *ZQPCVX* is found to be more effective and stable than the subroutine *VE02A* since the latter involves an inverse of a matrix, which can cause numerical instability.

Gradient and function evaluations are in general expensive in chemical process problems and hence the *CPU* time will not be used further as a measure of merit to test the efficiency of the algorithm on these test problems.

To further examine the effect of the problem size, as measured in terms of variables, on the efficiency of the *QP* code *ZQPCVX*, the following simple *QP* problem is constructed.

$$\begin{aligned} \min_{x_i} \quad & \sum_{i=1}^n (x_i - i)^2 \\ \text{s.t.} \quad & -b \leq x_i \leq b \quad i = 1, \dots, n \\ \text{with} \quad & b \geq 0 \end{aligned}$$

The use of this problem is that the number of active constraints can easily be controlled by selecting different values of *b*.

Table 2.5 **Comparison of Two QP Codes**

Problem No.	QP I		QP II	
	NG(NF)	CPU Total (TQP)	NG(NF)	CPU Total (TQP)
M47	21(21)	0.43 (0.36)	21(21)	0.12 (0.06)
M76	5(5)	0.14 (0.12)	5(5)	0.11 (0.08)
M79	6(7)	0.20 (0.17)	6(7)	0.12 (0.08)
M94	16(18)	1.19 (1.11)	16(18)	0.24 (0.16)
M98	F7		17(18)	0.58 (0.51)
M100	8(10)	0.72 (0.62)	8(10)	0.25 (0.13)
M101	51(51)	5.12 (4.67)	51(51)	1.60 (0.99)
M102	30(39)	2.16 (1.95)	30(39)	0.83 (0.60)
M104	16(16)	2.78 (2.62)	16(16)	1.01 (0.83)
C7	30(30)	3.50 (3.37)	30(30)	1.00 (0.87)

Key

QP I	The QP Code VE02A is used.
QP II	The modified QP code ZQPCVX is used.
NG (NF)	Number of gradient (function) evaluations.
F7	Failed because of singularity.
Machine	CDC Cyber 180-855 .
CPU	The Central Processing Unit time (sec).
Total	Total CPU time.
TQP	CPU time spent on QP .

Table 2.6 shows the effect of the number of variables on the above problem with $b = 2$ and Table 2.7 shows the effect of the number of active constraints on the problem with $n = 80$ for the QP code ZQPCVX.

These numerical results show that the efficiency of the algorithm of Goldfarb and Idnani is strongly affected by the number of active constraints. For this problem, the CPU time depends on the number of variables, n , and grows as $O(n^{2.47})$, as shown in Figure 2.1. The efficiency can be greatly improved by the use of an active set predicted from the previous iteration.

Table 2.6 Effect of the Number of Variables on the *QP*

Number of Variables	CPU time (sec)
10	0.008
30	0.095
50	0.353
80	1.268
100	2.342

Key

Machine *CDC Cyber 180-855.*

Table 2.7 Effect of the Number of Active Constraints on the *QP*

<i>b</i>	NACT	CPU time (sec)
2	79	1.268
25	56	1.068
40	41	0.909
60	21	0.651
100	0	0.315

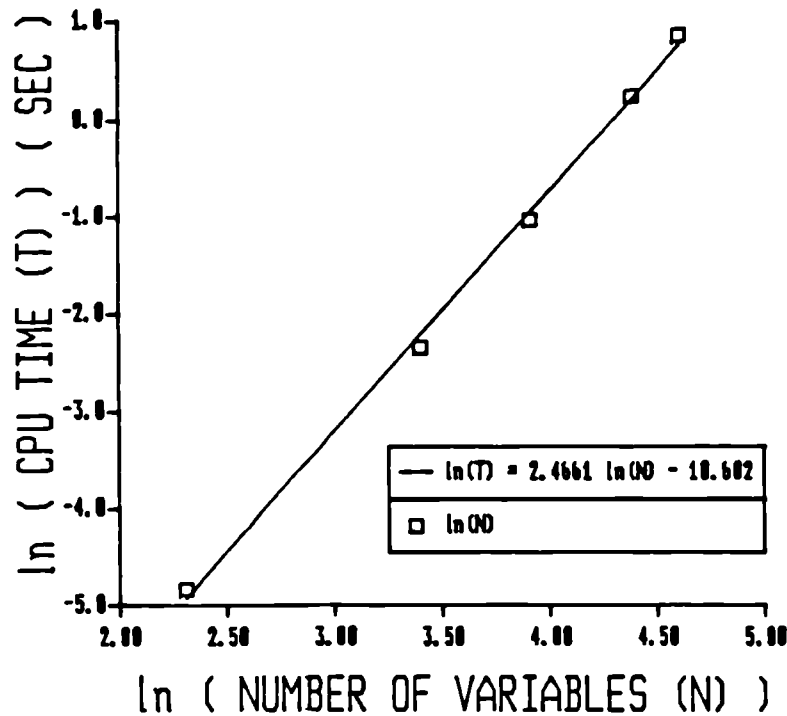
Key

b parameter for the simple bounds

NACT Number of active constraints.

Machine *CDC Cyber 180-855.*

Figure 2.1 Correlation of the CPU time in the QP



2.7.2 On the Choice of the Initial Hessian

Three strategies for the initial Hessian are examined as follows

Strategy I uses an identity matrix, Equation (2.64).

Strategy II is based on the order of magnitude of the variables, Equation (2.65).

Strategy III is based on the order of magnitude of gradients and variables, Equation (2.66).

The process problems are only considered here since they are in general badly scaled. The scaling factors for the initial Hessian are based on the standard starting points since they typically reflect the physical magnitude of the variables for these test problems. Table 2.8 shows that a proper choice of the initial Hessian can dramatically improve the performance of the algorithm especially for badly scaled problems in which the order of magnitude of variables differs in a wide range, e.g. Problem C9, C11, C12, C13, and C17. *Strategy III* is more effective than *Strategy II*. However, *Strategy III* employs the gradient information and this may be sensitive to the function noise when analytical derivatives are not available. *Strategy II* is therefore recommended when the order of magnitude of variables or a reasonable starting point is available.

Table 2.8 Effect of the Choice of the Initial Hessian

Problem No.	HESS I NG (NF)	HESS II NG(NF)	HESS III NG (NF)
C1	11(11)	9(9)	6(6)
C2	11(11)	11(11)	6(6)
C3	23(23)	23(23)	21(21)
C4	8(8)	8(8)	7(7)
C5	9(12)	9(14)	10(13)
C6	56(58)	40(40)	22(24)
C7	26(26)	9(9)	7(7)
C8	68(75)	62(68)	66(77)
C9	29(29)	9(10)	9(9)
C10	25(25)	28(28)	12(12)
C11	136(138)	70(70)	30(32)
C12	141(145)	69(71)	30(32)
C13	98(99)	43(45)	25 (25)
C14	49(49)	16(16)	5(5)
C15	21(23)	23(23)	12(13)
C16	20(20)	8(8)	21(21)
C17	17(17)	15(15)	8(8)

Key

HESS I	The initial Hessian is an identity matrix.
HESS II	The initial Hessian is a diagonal matrix based on Equation (2.65).
HESS III	The initial Hessian is a diagonal matrix based on Equation (2.66).
NG	Number of gradient evaluations.
NF	Number of function evaluations.
Machine	CDC Cyber 930.

2.7.3 On the Effect of the Step Bounds

To illustrate the necessity of imposing the step bounds, let us consider the following example which is a modified version of test problem *M 106*,

$$\begin{aligned}
 & \min_{x_1, x_2} \quad -486x_1 - 640x_2 \\
 M\ 115 \quad & \text{s.t.} \quad 100x_1^2 + 100x_2^2 - 385 = 0 \\
 & \ln(1000 - x_1^3 - x_2^4) - 2 \geq 0
 \end{aligned}$$

The solution of this problem is $(1.1866, 1.5626)^T$. When this problem starts with the point $x^0 = (1, 1)^T$, the *QP* subproblem at the first iteration can be formulated as

$$\begin{aligned}
 & \min_{d_{x_1}, d_{x_2}} \quad -486d_{x_1} - 640d_{x_2} + \frac{1}{2} (d_{x_1}^2 + d_{x_2}^2) \\
 & \text{s.t.} \quad 200d_{x_1} + 200d_{x_2} - 185 = 0 \\
 & \quad -0.003006 d_{x_1} - 0.004008 d_{x_2} + 4.9057 \geq 0
 \end{aligned}$$

The solution of this *QP* subproblem is $(-76.538, 77.462)^T$. It can be observed that the next point, $x^0 + d_x = (-75.538, 78.462)^T$, will cause the second constraint to become arithmetically undefined even though the point satisfies the linearised version of the second constraint. It is noted that this point may also be accepted even if the second constraint is ignored when the penalty parameter for the first constraint is not sufficiently large since the new estimate results in a large reduction of the objective function value. Therefore the new estimate is far away from the solution and an inefficiency may occur. However, when the *QP* is constrained by the following expressions,

$$\begin{aligned}
 -\max(1, |x_{1k}|) & \leq d_{x_{1k}} \leq \max(1, |x_{1k}|) & \text{and} \\
 -\max(1, |x_{2k}|) & \leq d_{x_{2k}} \leq \max(1, |x_{2k}|)
 \end{aligned}$$

the solution is $(-0.075, 1)^T$. The problem can hence be solved within 9 gradient and 9 function evaluations.

Although this problem is deliberately constructed, this behaviour can occur on process problems which are highly constrained by physical properties. Of course an inefficient optimisation may occur when the range of the step bounds is too small, as shown in Table 2.9. Chen and Stadtherr [1984] showed that the optimisation code *VF02AD* failed to solve these test problems in Table 2.9 without reducing the step length at the first three iterations. These results show that it is helpful to include the step bounds for highly constrained problems and is better to include

these bounds directly within the *QP* subproblem. Table 2.9 also shows that for the line search methods, the choice of step bounds as the order of magnitude of the corresponding variables performs satisfactorily for these test problems. It is just a coincidence that the solution of test problem *M 106* lies on a vertex of step bounds and hence the problem is solved within two iterations.

Table 2.9 Effect of Imposing the Step Bounds

Problem No.	BD I NG(NF)	BD II NG(NF)	BD III NG(NF)
M106	46(70)	2(2)	11(11)
M107	56(82)	22(31)	29(37)
M108	54(79)	25(26)	33(43)
M109	67(91)	24(32)	28(34)
M110	68(92)	25(37)	34(42)
M111	46(63)	22(29)	34(34)
M112	48(64)	20(28)	45(46)
M104	16(16)	16(16)	62(62)
C3	F4	19(24)	23(23)
C7	26(26)	26(26)	33(33)
C14	49(49)	38(38)	93(93)

Key

BD I	No step bounds are imposed.
BD II	The relative step bounds of 1 are used.
BD III	The relative step bounds of 0.1 are used.
NG	Number of gradient evaluations.
NF	Number of function evaluations.
F4	Failed because of no progress.
Machine	CDC Cyber 930 .

2.7.4 On Line Search and Trust Region Methods

In this section, four strategies are examined as follows.

- Strategy I implements a line search method with an exact penalty function as a merit function, equation (2.20). A watchdog technique without a restart scheme is introduced to improve local convergence property (Algorithm B.II).
- Strategy II implements a line search method with a modified augmented Lagrangian function, equation(2.34), as a merit function.
- Strategy III uses a trust region method with an exact penalty function, equation (2.20), as a merit function.
- Strategy IV uses a trust region method with an augmented Lagrangian function, equation (2.29), as a merit function.

The conceptual difference of the line search and trust methods is illustrated in Figure 2.2 via problem M115. For convenience, the objective function in *M115* is scaled by a factor of 0.1. It can be observed that the search direction is allowed to change in the trust region but not in line search method and the new estimates generated from the trust region method but not from the line search method satisfy the linearised version of the constraints when the constraints are consistent.

To explain why an augmented Lagrangian function as a merit function may allow more iterations to accept a full step at the feasible point, consider the following example [Powell, 1983 (a)],

$$\begin{aligned}
 & \min_{x_1, x_2} \quad -x_1 + 10(x_1^2 + x_2^2 - 1) \\
 M\ 116 \quad & s.t. \quad x_1^2 + x_2^2 - 1 = 0 \\
 & starting\ at\ (x_1, x_2) = (0.8, 0.6)
 \end{aligned}$$

The solution of this problem is at the point $(1, 0)^T$. When the Hessian of the *QP* starts with an identity matrix, the solution of the *QP* at the first iteration is $(0.36, -0.48)^T$ which gives the next estimate of the solution as $(1.16, 0.12)^T$. The Lagrangian multiplier λ for the constraint is 9.6. It may be argued that this point is a better estimate than the initial point and should be accepted for the next iteration since

$$\|(1.16, 0.12) - (1, 0)\|_2 (=0.2) < \|(0.8, 0.6) - (1, 0)\|_2 (=0.44)$$

It is noted that this estimate of the solution leads to an objective function increase and constraint violation. When the absolute value penalty function (equation (2.20)) is used as a merit function, the function becomes

$$\hat{P}_e(x, r) = -x_1 + 10(x_1^2 + x_2^2 - 1) + r |x_1^2 + x_2^2 - 1|.$$

The full step length is not allowed since

$$\hat{P}_e(x + d, r) (= 2.44 + 0.36r) > \hat{P}_e(x, r) (= -0.8)$$

for $r \geq 0$. It took 45 gradient and 116 function evaluations to reach the solution with the absolute tolerance 10^{-7} when the absolute penalty merit function without relaxations is used. However, when the augmented Lagrangian function (equation (2.29)) is employed for a merit function, the merit function takes the form

$$\hat{P}_a(x, \lambda, r) = -x_1 + 10(x_1^2 + x_2^2 - 1) + \frac{r}{2} \left(x_1^2 + x_2^2 - 1 - \frac{\lambda}{r} \right)^2 - \frac{\lambda^2}{2r}$$

The full step length may be accepted when the Lagrangian multiplier λ starts with 10 since

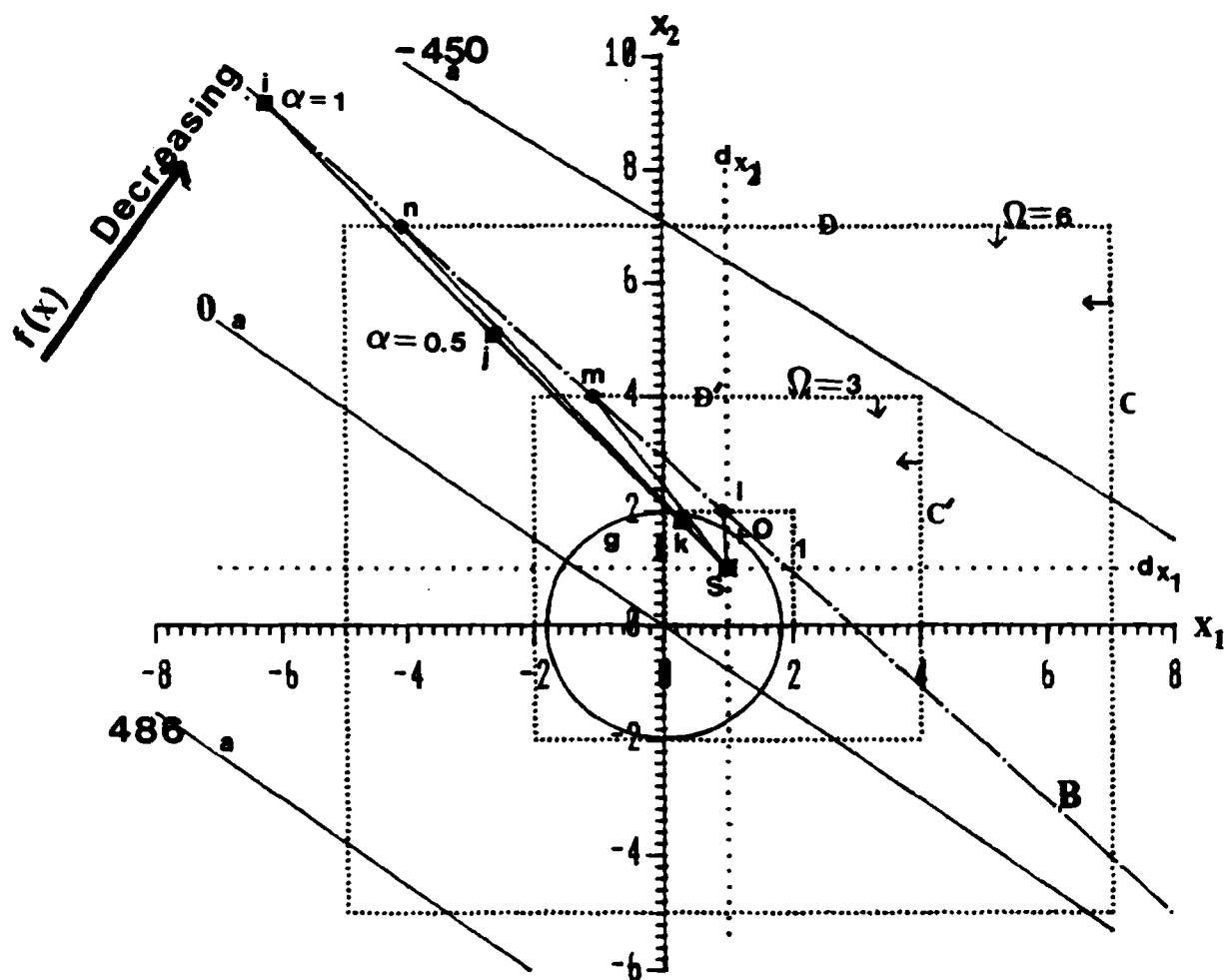
$$\begin{aligned} \hat{P}_a(x + d, 9.6, r) & \left(= 2.44 + \frac{r}{2} \left(0.36 - \frac{9.6}{r} \right)^2 \right) < \\ \hat{P}_a(x, 10, r) & \left(= -0.8 + \frac{r}{2} \left(0 - \frac{10}{r} \right)^2 \right) \end{aligned}$$

for $r < 9.6$. The seven decimals accuracy is obtained in 12 gradient and 19 function evaluations when the augmented Lagrangian merit function is used. This result also shows that it is difficult to devise a suitable merit function for nonlinear optimisation problems.

In general, the performance of an absolute value penalty function with the watchdog technique is comparable to that of the augmented Lagrangian function, as shown in Table 2.10.

The result also shows that trust region methods are comparable to line search methods, which agrees to some degree with the results obtained by Fan *et al.* [1988]. The trust region methods can provide sensible step bounds, which is useful for highly constrained problems, as illustrated previously.

Figure 2.2 Line search and Trust Region Methods



Key (Figure 2.2)

Problem		
line	a	— objective function ($\min_{x_1, x_2}$) $-48.6 x_1 - 64 x_2$
circle	g	— constraints (s.t.) $100 x_1^2 + 100 x_2^2 - 385 = 0$
point	s	— starting point $(x_1^0, x_2^0) = (1, 1)$
point	o	— solution $(x_1^*, x_2^*) = (1.1866, 1.5626)$
QP subproblem		
		— objective function ($\min_{d_x}$) $-48.6 d_{x_1} - 64 d_{x_2} + \frac{1}{2}(d_{x_1}^2 + d_{x_2}^2)$
line	B	— linearised constraints (s.t.) $200 d_{x_1} + 200 d_{x_2} - 185 = 0$
line	C	— trust region bound $x_1 \leq \Omega$
		— trust region bound $-\Omega \leq x_1$
line	D	— trust region bound $x_2 \leq \Omega$
		— trust region bound $-\Omega \leq x_2$
Line search method		
point	i	— new estimate $x^+ = x^0 + \alpha d_x, \Omega = \infty$
point	j	— new estimate <i>when</i> $\alpha = 1$
point	k	— new estimate <i>when</i> $\alpha = 0.5$
		<i>when</i> $\alpha = 0.1$
Trust region method		
point	l	— new estimate $x^+ = x^0 + d_x$
point	m	— new estimate <i>when</i> $\Omega = 1.$
point	n	— new estimate <i>when</i> $\Omega = 3.$
		<i>when</i> $\Omega = 6.$

Table 2.10 Comparison of Line Search and Trust Region Methods.

Problem No.	LS I NG (NF)	LS II NG(NF)	TR I NG (NF)	TR II NG(NF)
C1	11(11)	11(11)	11(11)	8(9)
C2	11(11)	11(12)	9(10)	8(9)
C3	23(23)	23(23)	17(22)	17(22)
C4	8(8)	8(8)	8(9)	8(9)
C5	9(12)	9(12)	13(16)	15(15)
C6	56(58)	53(54)	34(35)	42(63)
C7	26(26)	26(27)	26(27)	28(33)
C8	68(75)	84(88)	70(103)	69(82)
C9	29(29)	29(29)	29(29)	29(30)
C10	25(25)	25(26)	19(19)	25(29)
C11	136(138)	131(136)	141(178)	136(138)
C12	141(145)	141(144)	142(181)	130(134)
C13	98(99)	98(100)	100(197)	95(99)
C14	49(49)	49(49)	50(51)	49(49)
C15	21(23)	23(24)	28(34)	21(23)
C16	20(20)	20(21)	29(31)	25(26)
C17	17(17)	18(19)	17(17)	17(17)

Key

- LS I A line search method with an exact penalty merit function (equation (2.20)).
- LS II A line search method with a modified augmented Lagrangian merit function (equation (2.34)).
- TR I A trust region method with an exact penalty merit function (equaiton (2.20)).
- TR II A trust region method with an augmented Lagrangian merit function (equation (2.29)).
- NG (NF) Number of gradient (function) evaluations.
- Machine CDC Cyber 930.

2.7.5 On the Feasibility Improvement

Two strategies for the feasibility improvement are employed as follows :

Strategy I introduces a feasibility step before the line searches
(Algorithm B.III).

Strategy II imposes feasibility steps after the line searches
(Algorithm B.IV).

Let us consider problem *M 115* again for the first starting point, $(1,1)^T$, to explain the usefulness of the feasibility improvements. It can be observed that the second constraint is well defined when the first constraint is satisfied closely. Suppose the feasibility step on the first constraint is introduced before the line search, the feasibility step is obtained from equation (2.40), i.e. by solving the following QP problem at the point $(-75.538, 78.462)^T$ and the gradient of the first constraint is calculated analytically for convenience.

$$\begin{aligned} \min_{s_{x_1}, s_{x_2}} \quad & \frac{1}{2} (s_{x_1}^2 + s_{x_2}^2) \\ \text{s.t.} \quad & 15307.5 s_{x_1} + 15692.5 s_{x_2} - 1201050.124 = 0 \end{aligned}$$

The solution of this QP subproblem is $(37.75, -39.218)^T$ and hence the next point becomes $(-37.781, 39.2424)^T$. This feasibility procedure can be performed iteratively. After the 6th feasibility step, the iterate reaches the point $(-1.388, 1.44)^T$ at which the second constraint is well defined. After the line search, this point is accepted for the next optimisation iteration. In this way, the problem is solved after 6 optimisation iterations and 16 feasibility steps.

This result shows that it is necessary to satisfy some of the constraints more closely in some cases and that equation (2.40) can be employed to obtain a feasible point. Feasibility steps before line searches are of value especially when problems are highly constrained.

In order to explain how the feasibility improvement reduces the *Maratos effect*, let us consider problem *M116* again for the same starting point. A feasibility step can be introduced before the line search at the first iteration as follows,

$$\begin{aligned} \min_{s_{x_1}, s_{x_2}} \quad & \frac{1}{2} (s_{x_1}^2 + s_{x_2}^2) \\ \text{s.t.} \quad & 2.32 s_{x_1} + 0.24 s_{x_2} + 0.36 = 0 \end{aligned}$$

The solution of this *QP* subproblem is $(-0.1588, -0.1535)^T$. The new estimate becomes $(1.0064, 0.10411)^T$. This procedure can be repeated, the next feasibility step is $(-0.01171, -4.914 \times 10^{-4})^T$ and the new estimate becomes $(0.9946, 0.1036)^T$. This estimate can be accepted when the absolute value penalty function is used, since

$$\hat{P}_e(x + d, r) \quad (= -0.9933 + 1.37 \times 10^{-4} r) < \hat{P}_e(x, r) \quad (= -0.8)$$

for $r < 1400$.

In practice, the feasibility steps may not improve the efficiency of the algorithm, as shown in Table 2.11, since the feasibility step may result in an objective function increase. This implies that it is better to use a feasibility improvement only when it is judged to be necessary, e.g. when encountering a line search difficulty.

Table 2.11 Effect of the Feasibility Improvement

Problem No.	FEA I NG(NF)	FEA II NG(NF)	FEA III NG(NF)
C1	11(11)	11(11)	11(11)
C2	11(11)	6(11)	7(11)
C3	23(23)	23(36)	23(23)
C4	8(8)	8(13)	8(12)
C5	9(12)	10(14)	9(28)
C6	56(58)	51(99)	63(145)
C7	26(26)	32(63)	17(31)
C8	68(75)	96(201)	56(173)
C9	29(29)	30(59)	27(71)
C10	25(25)	18(35)	31(71)
C11	136(138)	136(275)	26(66) ^d
C12	141(145)	140(284)	28(72) ^d
C13	98(99)	94(192)	76(237)
C14	49(49)	33(64)	38(95)
C15	21(23)	21(41)	19(38)
C16	20(20)	17(32)	16(21)
C17	17(17)	28(46)	61(153)

Key

FEA I	No Feasibility step is introduced.
FEA II	A feasibility step is introduced before line searches.
FEA III	Feasibility steps are introduced after line searches.
NG	Number of gradient evaluations.
NF	Number of function evaluations.
<i>d</i>	The code converged to a different optimal point.
Machine	CDC Cyber 930 .

2.7.6 On the Effect of Scaling the Objective Function

Test problem *M 115* also serves as an example to illustrate the effect of scaling the objective function. The search direction d_x is a summation of the 'feasibility step' p_s (from equation (2.8)) and 'optimality step' p_t (from equation (2.9)). Therefore the search direction for the first iteration at the point $(1, 1)^T$ can be expressed as,

$$\begin{aligned} d_x &= p_s + p_t \\ &= (0.4625, 0.4625)^T + (-77, 77)^T \\ &= (-76.5375, 77.4625)^T \end{aligned}$$

It can be observed that the 'optimality step' is dominant because of the large values of the objective function gradient and because of an inappropriate initial Hessian approximation of the Lagrangian function. It may be reasonable for this problem to scale the objective function by 0.01, and the search direction becomes

$$\begin{aligned} d_x &= p_s + 0.01 p_t \\ &= (0.4625, 0.4625)^T + (-0.77, 0.77)^T \\ &= (-0.3075, 1.2325)^T \end{aligned}$$

which is the solution of the following *QP* subproblem,

$$\begin{aligned} \min_{d_{x_1}, d_{x_2}} \quad & -4.86d_{x_1} - 6.40d_{x_2} + \frac{1}{2} (d_{x_1}^2 + d_{x_2}^2) \\ \text{s.t.} \quad & 200d_{x_1} + 200d_{x_2} - 185 = 0 \\ & -0.003006 d_{x_1} - 0.004008 d_{x_2} + 4.9057 \geq 0 \end{aligned}$$

The next point hence becomes $(0.6925, 2.2325)^T$ at which the second constraint is well defined. The problem can hence be solved within 9 gradient and 12 function evaluations.

This result shows that it is more suitable to scale the objective function from information on the gradient of the objective function, the Hessian of the Lagrangian function, and the nonlinearity of the constraints.

Table 2.12 shows the numerical results of an algorithm where the objective function is scaled by an arbitrary scaling factor. It gives an indication of the robustness of the method with badly scaled problems.

Table 2.12 Effect of Scaling the Objective Function

Problem No.	SOB I NG(NF)	SOB II NG(NF)	SOB III NG(NF)
C1	12(12)	10(10)	8(8)
C2	11(11)	11(11)	11(11)
C3	25(25)	24(25)	19(21)
C4	11(11)	6(6)	4(4)
C5	6(7)	9(14)	9(14)
C6	29(32) ^d	50(53)	35(36)
C7	41(41)	22(22)	16(16)
C8	98(102)	45(46)	41(53)
C9	38(42)	22(22)	16(16)
C10	20(20)	12(12)	16(19)
C11	7 (7) ^d	114(114)	101(104)
C12	14(13) ^d	79(80)	61(61)
C13	7(7) ^d	120(122)	103(103)
C14	51(51)	23(23)	16(16)
C15	22(23)	27(27)	26(30)
C16	23(23)	14(14)	12(12)
C17	24(24)	12(12)	16(17)

Key

SOB I	The Objective function is scaled by a factor of 0.1.
SOB II	The Objective function is scaled by a factor of 10.
SOB III	The Objective function is scaled by a factor of 100.
NG	Number of gradient evaluations.
NF	Number of function evaluations.
<i>d</i>	The code converged at a different optimal point.
Machine	<i>CDC Cyber 930</i> .

2.7.7 On a Problem with Linearly Dependent and Inconsistent Constraints

Consider the problem [Bartholomew-Biggs, 1986]

$$\begin{aligned} \min_{x_1, x_2} \quad & x_1 \\ \text{s.t.} \quad & 1 - x_1^2 - x_2^2 = 0, \\ & x_1 + x_2 \geq 0. \end{aligned}$$

The solution of this problem is the point

$$(x_1, x_2)^T = \left(\frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right)^T,$$

and at this point both constraints are active and are linearly independent.

Suppose the iterative process starts with the infeasible point $(-1, -1)^T$. The QP subproblem can be written as

$$\begin{aligned} \min_{d_{x_1}, d_{x_2}} \quad & d_{x_1} + \frac{1}{2}(d_{x_1}^2 + d_{x_2}^2) \\ \text{s.t.} \quad & -1 + 2d_{x_1} + 2d_{x_2} = 0, \\ & -2 + d_{x_1} + d_{x_2} \geq 0. \end{aligned}$$

Clearly, the constraints are inconsistent. The constraints of this QP subproblem are also found to be inconsistent when Powell's modification, equation (2.71), is applied. However, when the penalty function method is used, the QP subproblem can be formulated as

$$\begin{aligned} \min_{d_{x_1}, d_{x_2}} \quad & d_{x_1} + \left(-1 + 2d_{x_1} + 2d_{x_2} \right)^2 + \frac{1}{2}(d_{x_1}^2 + d_{x_2}^2) \\ \text{s.t.} \quad & -2 + d_{x_1} + d_{x_2} \geq 0. \end{aligned}$$

This problem is well defined and has a solution at the point $(0.2727, 1.7272)^T$. The next estimate becomes $(-0.7272, 0.7272)^T$ which is a better estimate. The problem is solved after 7 gradient and function evaluations.

This example shows that the penalty function methods provide a way for dealing with linearly dependent and inconsistent constraints.

2.7.8 On the Performance of Current SQP Optimisation Codes

In order to understand the performance of various SQP optimisation methods, three effective codes are studied, namely, *OPXRQP*, *VMCWD*, and *NLPQL*.

The optimisation code, *OPXRQP*, realises a recursive equality quadratic programming (*REQP*) methods [Bartholomew-Biggs, 1982].

The *VMCWD* code implements the SQP method with an absolute value penalty merit function, equation (2.20), and a watchdog technique with restart schemes [Powell, 1982 (b)], [Powell, 1983 (b)]. The original QP of this code is replaced by the more effective QP code *ZQPCVX*. The strategy for dealing with constraint inconsistency, mentioned in section 2.5.2 is employed. This code was also modified to calculate functions and gradients separately, and a reverse communication is imposed.

The code *NLPQL* [Schittkowski, 1984] employs an SQP method with an augmented Lagrangian function as a merit function. The QP code in this optimisation routine is a version of the *ZQPCVX* code modified by Schittkowski.

The numerical results on the mathematical test problems are presented in the appendix, B.1 and those on the chemical problems are shown in Table 2.13. A summary of the numerical results on all the test problems is listed in Table 2.14.

With regards to the treatment of inequality constraints, there is at present some debate about the relative efficiency between the equality quadratic programming (*EQP*) subproblem and the inequality quadratic programming (*IQP*) subproblem. In the *EQP* formulation, constraints which are violated are regarded as active constraints and hence the QP subproblem only contains equality constraints. In the *IQP* subproblem both mixed equality and inequality constraints are involved and an active set strategy is used to predict the active constraints. In practice, the *EQP* subproblem is much easier to solve than the *IQP* one. In fact, when the estimates are near the solution, a similar performance can be achieved since an active set predicted from previous iterations can be used for the *IQP* subproblem and sufficiently large penalty parameters can be used to force the iterates to satisfy the linearised version of the constraints in the *EQP* subproblem.

The first optimisation code implements the *EQP* strategy while the other two codes realise the *IQP* strategy.

Table 2.13 Comparison of Three Optimisation Codes on Process Problems

Problem No.	OPRQP NG(NF)	VMCWD NG(NF)	NLPQL NG(NF)
C1	39(66)	11(11)	13(13)
C2	F4	6(7))	8(30)
C3	F6	F6	F6
C4	19(19)	4(6)	9(9)
C5	94(648) ^s	11(14)	11(14)
C6	211(605)	42(46)	63(145)
C7	F5	30(31)	45(64)
C8	259(765)	118(178)	69(83)
C9	F5	29(29)	48(78)
C10	25(36)	17(17)	18(18)
C11	F5	F5	F5
C12	F5	F5	F5
C13	F5	123(158)	F5
C14	F4	48(48)	38(95)
C15	38(52)	20(21)	20(21)
C16	190(298)	20(20)	20(20)
C17	83(83) ^s	17(17)	18(34)

Key

OPRQP	The OPRQP optimisation code is used.
VMCWD	The modified VMCWD optimisation code is used.
NLPQL	The NLPQL optimisation code is used.
NG (NF)	Number of gradient (function) evaluations.
F4	Failed because of no progress.
F5	Failed because of more than 500 function evaluations.
F6	Failed because of divergence.
^s	Terminated abnormally at the point near an optimum.
Machine	CDC Cyber 180-855 .

The results in Table 2.13 show that the *REQP* method in general requires more gradient and function evaluations. The code *OPXRQP* is more susceptible to 'hard' constraints since the method belongs, by its nature, to a class of penalty function method.

Table 2.14 Statistical Results of the Comparison of Three Optimisation Codes

The Performance	Number of Problems		
	OPRQP	VMCWD	NLPQL
Convergence obtained			
$NG(NF) \leq 20$	51(40)	90(85)	88(80)
$21 < NG(NF) \leq 50$	39(38)	18(16)	21(16)
$NG(NF) > 50$	16(28)	9(16)	6(19)
Premature termination at point near the solution (s)	10	3	3
Total number of problems Solved (%)	116(88%)	120(92%)	118 (90%)
Converged to a different solution	3	5	5
Failed	12	6	8
Total Number of Problems tested	131	131	131

Key

- NG Number of gradient evaluations.
 NF Number of function evaluations.
 (s) The point is near the solution when $Tol \leq 10^{-4}$.
 Tol Tolerance.

Most of the failures of the modified *VMCWD* code are due to insufficiently large penalty parameters. This failure can be improved by increasing their values automatically whenever necessary. This feature has been included in the test program.

Occasionally the code *NLPQL* failed due to inconsistent constraints and this can be improved by the proposed strategy.

In fact, the robustness of these three optimisation codes is approximately comparable.

Comparing the numerical results in Table 2.13 with those in Table 2.9 and 2.10 shows that the proposed algorithm improves the robustness and efficiency of the current *SQP* algorithm. It is required to introduce the step bound to solve problem C3 and the scaling the initial Hessian of problem C11, C12 and C13 greatly reduces the function and gradient evaluations.

2.8 Discussion and Conclusions

In this chapter, a class of *SQP* methods and various algorithmic components are examined. The relationships between the *SQP* methods and other optimisation methods are discussed. Several strategies and alternatives to enhance the performance of the *SQP* methods are proposed and tested.

The robustness of the *SQP* methods is improved by imposing step bounds and/or by introducing feasibility improvements as illustrated by a simple mathematical problem. Imposing the step bounds ensures the linear approximation to the constraints is valid within the range, while introducing the feasibility step allows the iterates to follow the arc of the nonlinear constraints more closely.

The performance of the trust region methods strongly relies on the validity of the approximation of the subproblem within the range. The proposed strategy for automatic adjusting of the bounds appears to be effective. The performance of the line search methods depends on the underlying merit function. The use of an absolute value penalty function as merit function requires proper parameters which balance the objective function decrease and constraint violation while the use of an augmented Lagrangian function needs appropriate parameters to convexify the Lagrangian function. Feasibility improvements attempt to provide a 'natural' balance between objective function decrease and constraint violations.

Scaling the objective function implicitly tends to balance the 'optimality step' and the 'feasibility step'.

Numerical results show that inefficiencies may occur when improper values for step bounds, trust region bound, the degree of feasibility or penalty parameters for the merit function are used. This is because a small optimisation step may be taken.

In fact, all these problems relate to how good the approximation subproblem is, which is solved at each optimisation iteration. For nonlinear problems, it is not trivial to estimate the approximation correctness and to choose suitable values for step bounds, the degree of feasibility, the trust region bounds, and parameters for the merit function without higher order information. In this study, heuristic rules are employed to choose these parameters and hence the performance depends on the characteristics of the test problems. The method, however, performs well in practice on a large set of problems, using default values of the parameters and of the algorithm options.

Chapter 3

Large Scale Optimisation

3.1 Introduction

The success of optimisation methods for small scale systems has revived the interest in applying these techniques to large scale chemical processes because many process optimisation problems in engineering contain a large number of variables and constraints. However, in general each constraint will contain only few variables. Also the degree of freedom of the system, the difference between the number of variables and equality constraints, is relatively small. Thus optimisation techniques for large sparse systems are needed.

The definition of 'large', of course, depends both on the available machine (eg. available storage and computation time) and on the numerical methods used. The extension of optimisation methods to large sparse systems is not trivial since many effective alternatives for small systems may become prohibitively expensive for large scale systems in terms of storage and arithmetical operations, for example, those methods using curvature information of the objective function and constraints. On the contrary, some options that are not likely to be considered for small problems may become advantageous with large scale problems. This subject was recently reviewed by Gill, *et al.* [1981 (b)] and Toint [1988].

The successive linear programming method (*SLP*, or method of approximation programming) [Griffith and Stewart, 1961] is one of the practical algorithms for large scale optimisation. It consists of the solution of a sequence of linear programming subproblems in which the objective function and the constraints are linearised at the previous solution. Effective tools for the solution of large scale linear programming problems are available, eg. simplex methods [see e.g. Gill, Murray and Wright, 1981] or the Karmarkar algorithm [Karmarkar, 1984]. This method has been applied in the oil industry with some success [Sauer, Colville and Burwick, 1964]. However, its drawback is that it oscillates around the optimum when the optimum does not lie on a vertex. Special care must be taken, for example, by restricting the search direction [Palacios-Gomez, Lasdon and Enquist, 1982]. The convergence properties of the *SLP* method on nonlinear problems have not yet been established.

GRG methods or projected gradient methods, as discussed in Chapter 2, also have the capability to deal with large scale optimisation problems when a large number of active constraints are present [Lasdon *et al.*, 1978], [Gabriele, 1980], [Drud, 1985]. In general, they require a large number of function evaluations.

Much work has been concentrated on the application of the SQP method to large scale systems since the good performance of the method was established on small scale problems, as shown in the previous chapter.

The first problem with the application of the SQP method to large scale systems is that the traditional BFGS update may result in a full and dense Hessian approximation, even though the original Hessian of the Lagrangian function is sparse. Consequently, the Hessian approximation becomes too large to store for problems with more than, say, 1000 variables. A sound Hessian update scheme which can preserve sparsity, symmetry, and positive definiteness at the same time is not presently known. An analytical Hessian of the Lagrangian function can be used when the Hessian is sparse [Burchett, Happ and Vierath, 1984]. For a specific application, the optimisation of a distillation column problem, Lucia and Kumar [1986] suggested the construction of a Hessian approximation by combining an analytical Hessian, if available, with a thermodynamically constrained quasi-Newton update for the rest of non-zero Hessian elements. Their procedure attempts to keep the projected Hessian positive definite on the tangent plane of the active constraints. For general problems, however, an exact Hessian may be expensive to evaluate and constructing a Hessian which is positive definite on the tangent plane of the active constraints is not trivial, since it is not known in advance which constraints will be active. Alternatively, a trust region method may be used to prevent the QP problem from becoming unbounded when the Hessian is not positive definite. This is an active research area [Powell, 1987]. On the other hand, the decomposition of an indefinite matrix may not be stable from a numerical point of view [Gill, Murray and Wright, 1981 (a)].

Particular advantage may be taken of the structure of the system when an optimisation problem is partially separable. In this case the problem can be decomposed into a set of subproblems, each one involving only a small number of variables, and a partitioned BFGS update for the Hessian of the Lagrangian function can be applied [Griewank and Toint, 1982 (a)], [Griewank and Toint, 1982 (b)], [Toint, 1986]. However, a chemical process may contain many nested recycles, and it may be difficult to find such a decomposition. Also, stable implementation techniques for maintaining positive definiteness need further study

and at present it is not clear whether the method can handle the nonconvex problems which characterise general chemical process optimisation problems [Coleman, 1984].

It is also possible to neglect the Hessian matrix altogether, and replace it by an identity matrix [Pshenichnyj, 1970], [Pshenichnyj, 1987], or by a diagonal matrix. This method is equivalent to projecting the negative gradient of the objective function onto the tangent space of the constraints, i.e. in some sense a version of the projected steepest descent method. In fact, the Hessian matrix of the Lagrangian function can be replaced by the Hessian matrix of the objective function when constraints are all linear [Gafni and Bertsekas, 1984]. Slow convergence often occurs for nonlinearly constrained problems since no information of the curvature of the constraints is being used [Powell, 1983 (a)].

The second problem arising in the application of the *SQP* method to large scale problems is that the solution of a large sparse *QP* subproblem may be prohibitively expensive. To reduce this overhead, Dembo and Tulowitzki [1985] developed a truncated (inexact) quadratic programming method in which the *QP* subproblem is solved approximately. A similar approach was proposed by Fontecilla [1985] and Dembo [1987]. Escudero [1986] applied this approach to a nonlinear network problem. Unfortunately, it is difficult to apply this technique to a nonlinearly constrained optimisation problem since the iterates are generally not feasible and it is difficult to devise a global scheme to force convergence. Alternatively, Berna *et al.* [1980] proposed a decomposition technique aimed at reducing the size of the quadratic programming problem and hence the cost of solving it. In their implementation, the Hessian approximation is stored in a factorised form to reduce the excessive storage. A restart scheme after a few iterations is necessary for this approach since the storage requirements increase with the number of iterations [Hutchison, Jackson and Morton, 1986]. Renfro [1986] showed that this restart strategy does not affect the global convergence. However, it may slow down the convergence rate and affect the overall efficiency of the method.

On the other hand, the projected Hessian onto the tangent plane of the active constraints may be updated directly. Based on this idea, various algorithms were suggested by Coleman and Conn [1982], Gaddy [1982], Locke *et al.* [1983 (b)], Nocedal and Overton [1985], Gill *et al.* [1985 (b)], Skrobański [1986] and Vasantharajan and Biegler [1987]. Locke *et al.* [1983 (b)] proposed a way to deal with inequality constraints and developed a decomposition technique to implement this approach. Irregular convergence and discontinuity properties

embedded within the method that uses the projected Hessian were observed by several researchers [Coleman and Sorensen, 1984], [Yuan, 1985 (a)], [Byrd, 1985 (a)], [Gill et al., 1985 (a)], [Byrd and Schnabel, 1986].

It is useful to re-examine in more detail this decomposition technique that uses a reduced Hessian, since it has a potential to handle large scale systems with few degrees of freedom and relatively few numerical results have been published [kisala *et al.*, 1987], [Chan and Prince, 1986].

In the following, a class of decomposition methods that use a reduced Hessian is proposed. This chapter starts by describing a framework for the decomposition algorithm. Following the derivation of the decomposition method, its algorithmic components are examined. These include a procedure for forcing convergence and the update of the reduced Hessian. The implementation of this decomposition method is then discussed and a complete algorithm is presented. The decomposition method is then compared with current large scale optimisation methods and numerical results are also presented.

3.2 Decomposition Method

A basic decomposition algorithm for a large sparse system can be stated as follows :

Algorithm C.I — Basic Decomposition Algorithm

1. Given initial conditions.
2. Construct and solve a reduced quadratic programming subproblem. Calculate the search direction and estimate the Lagrangian multipliers.
3. Check the termination criterion. If it is satisfied, then stop. Otherwise, the algorithm continues.
4. Apply procedures to force convergence and improve feasibility, and obtain the new estimate of the solution.
5. Update the reduced Hessian, and go to step (2).

The critical points in this algorithm are the construction and solution of a reduced model in step (2), the procedure to force the convergence in step (4), and the update of the reduced Hessian in step (5). Each of these will be examined in detail.

3.2.1 Construction of Reduced QP Models

For convenience, the optimisation problem to be considered is restated as

$$\begin{array}{ll}
 \min_x & f(x) \\
 \text{P B.I} & \text{s.t. } h_i(x) = 0 \quad i = 1, \dots, meq \\
 & g_j(x) \geq 0 \quad j = 1, \dots, mi
 \end{array} \quad (2.1)$$

where

x is an n -dimensional vector of variables.

$f(x)$ is a scalar objective function.

$h_i(x)$ is an equality constraint.

$g_i(x)$ is an inequality constraint.

meq is the number of equality constraints.

mi is the number of inequality constraints.

It is assumed that the number of variables and constraints in Problem P B.I is large and the Jacobian matrices $\nabla_x h$ and $\nabla_x g$ are sparse. As discussed in Chapter 2, a QP sub-problem for the SQP method at the k^{th} iteration can be formulated as,

$$\begin{array}{ll}
 \min_{d_{x_k}} & \nabla_x f(x_k)^T d_{x_k} + \frac{1}{2} d_{x_k}^T B_k d_{x_k} \\
 \text{QP B.II} & \text{s.t. } h(x_k) + \nabla_x h(x_k)^T d_{x_k} = 0 \\
 & g(x_k) + \nabla_x g(x_k)^T d_{x_k} \geq 0
 \end{array} \quad (2.6)$$

where

$\nabla_x f(x)$ is the gradient of the objective function.

$\nabla_x h(x)$ is the gradient of the equality constraints.

$\nabla_x g(x)$ is the gradient of the inequality constraints.

B is an approximation to the Hessian of the Lagrangian function.

d_x is a search direction.

As observed in the previous chapter, in the SQP method the search direction d_x is the summation of two steps : a ' feasibility step ' p_s and an ' optimality step ' p_t . The search direction d_x can hence be expressed as

$$d_x = p_s + p_t. \quad (2.7)$$

Since the set of active constraints at the solution is not known in advance, it is possible to choose initially just the equality constraints as active constraints. A set of active inequality constraints is then identified in the QP subproblem. Based on this idea, the 'feasibility step' at the k^{th} iteration p_{s_k} may be defined as

$$\begin{aligned}
 QP \text{ C.I} \quad & \min_{p_{s_k}} \quad \frac{1}{2} p_{s_k}^T B_k p_{s_k} \\
 & s.t. \quad h(x_k) + \nabla_x h(x_k)^T p_{s_k} = 0 \\
 & \quad \quad g(x_k) + \nabla_x g(x_k)^T p_{s_k} \geq 0
 \end{aligned} \tag{3.1}$$

The 'optimality step' at the k^{th} iteration p_{t_k} may be defined as,

$$\begin{aligned}
 QP \text{ C.II} \quad & \min_{p_{t_k}} \quad \nabla_x f(x_k)^T p_{t_k} + \frac{1}{2} p_{t_k}^T B_k p_{t_k} \\
 & s.t. \quad \nabla_x h(x_k)^T p_{t_k} = 0 \\
 & \quad \quad g(x_k) + \nabla_x g(x_k)^T (p_{s_k} + p_{t_k}) \geq 0
 \end{aligned} \tag{3.2}$$

It is assumed that the Jacobian matrix $\nabla_x h(x)$ has full rank and there exist matrices Y and Z such that the search directions p_s and p_t may be formulated as

$$p_s = Y (-h(x)) + Z p_z$$

and

$$p_t = Z p_n. \tag{3.3}$$

where

p_z is an $(n - meq)$ -dimensional vector.

p_n is an $(n - meq)$ -dimensional vector.

The matrices Y and Z are constructed such that they have the following properties :

$$\nabla_x h(x)^T Y = I$$

and

$$\nabla_x h(x)^T Z = \hat{O} \tag{3.4}$$

where

- I is an identity matrix.
- Y is an $(n \times meq)$ matrix.
- Z is an $(n \times (n - meq))$ matrix.
- $\hat{O}$ is a null matrix.

In general, there are two ways to construct Y and Z , namely, the direct variable elimination method and the projection method.

With the direct variable elimination method, it is assumed that the variables x can be decomposed into dependent variables v and independent variables u , i.e.

$$x^T = (v^T, u^T). \quad (3.5)$$

The Jacobian matrix $\nabla_x h(x)$ can hence be decomposed as follows, where the matrix $\nabla_v h(x)$ is assumed to be non-singular.

$$\nabla_x h(x)^T = (\nabla_v h(x)^T \quad \nabla_u h(x)^T)$$

The matrices Z and Y can be chosen as follows

$$\begin{aligned} Z &= \begin{pmatrix} -\nabla_v h(x)^{-T} & \nabla_u h(x)^T \\ I & \end{pmatrix} \\ Y &= \begin{pmatrix} \nabla_v h(x)^{-T} \\ \hat{O} \end{pmatrix} \end{aligned} \quad (3.6)$$

Gabay [1982] used this technique to solve optimisation problem with equality constraints.

With the projection method, the Jacobian matrix $\nabla_x h(x)$ is decomposed as follows via an orthogonal decomposition (e.g. QR decomposition).

$$\begin{aligned} \nabla_x h(x) &= Q \begin{pmatrix} R \\ \hat{O} \end{pmatrix} \\ &= (Q_1 \quad Q_2) \begin{pmatrix} R \\ \hat{O} \end{pmatrix} \end{aligned} \quad (3.7)$$

where

$$Q^T Q = I$$

R is an $(meq \times meq)$ upper triangular matrix.

- Q is an $(n \times n)$ nonsingular and orthogonal matrix.
 Q_1 is an $(n \times meq)$ matrix.
 Q_2 is an $(n \times (n - meq))$ matrix.

The columns of Q_1 are an orthonormal basis of the column space of $\nabla_x h(x)$ and the columns of Q_2 are an orthonormal basis of the null space of $\nabla_x h(x)$. The matrices Y and Z can then be chosen as

$$\begin{aligned}
 Y &= Q_1 R^{-T}, \\
 Z &= Q_2.
 \end{aligned} \tag{3.8}$$

Since the columns of Y and Z are orthonormal, these matrices have the following properties,

$$\begin{aligned}
 Y^T Z &= \hat{O}, \\
 Z^T Z &= I
 \end{aligned}$$

This technique was used by Coleman and Conn [1982], Nocedal and Overton [1985], and Gill *et al.* [1985 (b)].

For large scale systems, the QR factorisation is more expensive in terms of both storage and computational time, and the matrices Q_2 and R may become dense matrices even though the Jacobian matrix $\nabla_x h(x)$ is sparse [Gill, Murray, Saunders and Wright, 1984 (b)]. The direct elimination method is therefore favoured. In this chapter, only the direct variable elimination method is considered.

The substitution of the definition of p_s and p_t (equation (3.3)) into the above two subproblems QP C.I and QP C.II respectively yields

$$\begin{aligned}
 \min_{p_{z_k}} & - p_{z_k}^T Z_k^T B_k Y_k h(x_k) + \frac{1}{2} p_{z_k}^T Z_k^T B_k Z_k p_{z_k} \\
 \text{s.t.} & (g(x_k) - \nabla_x g(x_k)^T Y_k h(x_k)) + \nabla_x g(x_k)^T Z_k p_{z_k} \geq 0 \quad (3.9) \\
 \text{and then} & \\
 p_{s_k} &= -Y_k h(x_k) + Z_k p_{z_k}
 \end{aligned}$$

where $Z_k^T B_k Z_k$ is referred to as a reduced (projected) Hessian.

and

$$\begin{aligned}
& \min_{p_{n_k}} \nabla_x f(x_k)^T Z_k p_{n_k} + \frac{1}{2} p_{n_k}^T Z_k^T B_k Z_k p_{n_k} \\
RQP\ C.II \quad & s.t. (g(x_k) + \nabla_x g(x_k)^T p_{s_k}) + \nabla_x g(x_k)^T Z_k p_{n_k} \geq 0 \quad (3.10) \\
& \text{and then} \\
& p_{t_k} = Z_k p_{n_k}
\end{aligned}$$

where $\nabla_x f(x_k)^T Z_k$ is referred to as a reduced gradient.

The resulting sub-problems *RQP C.I* and *RQP C.II* are referred to as reduced QP sub-problems, and define a class of decomposition methods. It is assumed that the reduced (projected) Hessian $Z^T B Z$ is positive definite. When it is updated directly, the matrix $Z^T B Y$ is not available and hence the 'feasibility step' p_s cannot be calculated directly. Here different models are proposed to define this step, p_s .

3.2.1.1. Reduced Model I

Ignoring the matrix $Z^T B Y$ and the inequality constraints in the *RQP C.I* subproblem, p_{s_k} becomes zero. The resulting subproblem is simply reduced to

$$p_{s_k} = -Y_k^T h(x_k) \quad (3.11)$$

and p_{n_k} is solved from equation (3.10).

The search direction d_x at the k^{th} iteration is therefore

$$\begin{aligned}
d_{x_k} &= p_{s_k} + p_{t_k} \\
&= -Y_k^T h(x_k) + Z_k p_{n_k}
\end{aligned} \quad (3.12)$$

The reduced QP subproblem *RQP C.II* also predicts the active constraints and estimates the Lagrangian multipliers for the active inequality constraints, ν_k . The Lagrangian multipliers for the equality constraints λ_k can be estimated from the following equation :

$$\lambda_k = Y_k^T (\nabla_x f(x_k) - \nabla_x g(x_k)^T \nu_k) \quad (3.13)$$

This is equivalent to solving the following least squares problem when the constraints are consistent and linearly independent.

$$\min_{\lambda} \quad \|\nabla_x f(x) - \nabla_x h(x) \lambda - \nabla_x g(x) \nu\|_2 \quad (3.14)$$

It can be observed that this model is the same as that proposed by Locke *et al.* [1983 (b)].

In fact, this reduced model I is equivalent to solving the following system :

$$\begin{aligned} \min_{d_{x_k}} \quad & \nabla_x f(x_k)^T d_{x_k} + \frac{1}{2} d_{x_k}^T \hat{B} d_{x_k} \\ \text{s.t.} \quad & h(x_k) + \nabla_x h(x_k)^T d_{x_k} = 0 \\ & g(x_k) + \nabla_x g(x_k)^T d_{x_k} \geq 0 \end{aligned} \quad (3.15)$$

where

$$\hat{B} = \begin{pmatrix} \hat{O} & \hat{O} \\ \hat{O} & B_{uu} \end{pmatrix}.$$

u is an $(n - meq)$ -dimensional vector of design variables.

B_{uu} is an approximation to the second order derivatives of the Lagrangian function with respect to the design variables u .

$$B_{uu} = \nabla_{uu}^2 f(x_k) - \sum_{i=1}^{meq} \nabla_{uu}^2 h_i(x_k) \lambda_i - \sum_{j=1}^{mi} \nabla_{uu}^2 g_j(x_k) \nu_j$$

Also, from equation (3.6) and equation (3.13),

$$\begin{aligned} Z^T (\nabla_x f(x) - \nabla_x g(x) \nu) &= \nabla_u f(x) - \nabla_u g(x) \nu \\ &\quad - \nabla_u h(x) \nabla_v h(x)^{-1} (\nabla_v f(x) - \nabla_v g(x) \nu) \\ &= \nabla_u f(x) - \nabla_u h(x) \lambda - \nabla_u g(x) \nu \end{aligned} \quad (3.16)$$

The matrix B_{uu} can therefore be updated based on equation (3.16).

3.2.1.2. Reduced Model II

Replacing the Hessian matrix B in RQP C.I by an identity matrix, and ignoring the inequality constraints yields

$$p_{s_k} = -Y_k h(x_k) + Z_k (Z_k^T Z_k)^{-1} Z_k^T Y_k h(x_k) \quad (3.17)$$

and then p_{n_k} is solved from RQP C.II .

The search direction d_{x_k} thus becomes

$$\begin{aligned} d_{x_k} &= p_{s_k} + p_{t_k} \\ &= -Y_k h(x_k) + Z_k \{ (Z_k^T Z_k)^{-1} Z_k^T Y_k h(x_k) + p_{n_k} \} \\ &= - \{ I - Z_k (Z_k^T Z_k)^{-1} Z_k^T \} Y_k h(x_k) + Z_k p_{n_k} \end{aligned} \quad (3.18)$$

The Lagrangian multipliers for the equality constraints λ at the k^{th} iteration may be estimated as

$$\lambda_k = Y_k^T (I - Z_k (Z_k^T Z_k)^{-1} Z_k^T) (\nabla_x f(x_k) - \nabla_x g(x_k) \nu_k) \quad (3.19)$$

It can be shown that a search direction generated from this model is the same as that of the resulting method proposed by Vasantharajan and Biegler [1987].

3.2.1.3 Reduced Model III

Replacing the Hessian matrix B in RQP C.I by an identity matrix, the resulting problem can be formulated as

$$\begin{aligned} \min_{p_{z_k}} & - p_{z_k}^T Z_k^T Y_k h(x_k) + \frac{1}{2} p_{z_k}^T Z_k^T Z_k p_{z_k} \\ RQP \text{ C.III} \quad & s.t. (g(x_k) - \nabla_x g(x_k)^T Y_k h(x_k)) + \nabla_x g(x_k)^T Z_k p_{z_k} \geq 0 \\ & \text{and then} \\ & p_{s_k} = -Y_k h(x_k) + Z_k p_{z_k} \end{aligned} \quad (3.20)$$

The search direction d_{x_k} with p_{t_k} from equation (3.10) can be obtained as

$$\begin{aligned} d_{x_k} &= p_{s_k} + p_{t_k} \\ &= -Y_k h(x_k) + Z_k p_{z_k} + Z_k p_{n_k} \end{aligned} \quad (3.21)$$

The Lagrangian multipliers for the equality constraints λ at the k^{th} iteration may also be estimated using equation (3.19). One of the advantages of this formulation is that step bounds on the 'feasibility step' and the 'optimality step' can easily be imposed.

It is noted that these three formulations differ in the way of calculating the 'feasibility' step p_s . In the first formulation, neglecting the matrix $Z^T B Y$ can be valid only when the dependent variables u in the equality constraints $h(x)$ appear linearly, or when the residuals $h(x)$ are small. This step depends on the choice of the dependent variables. In the second formulation, replacing the Hessian matrix by an identity matrix is equivalent to ignoring the curvature of the objective function and constraints (the Lagrangian function). This step does not depend on the choice of design variables. However, it depends on the scale of the variables because the step will change when the identity matrix is replaced by a positive definite diagonal matrix, i.e., the matrix $Z^T Y$ can be replaced by the matrix $Z^T \bar{D} Y$ where $\bar{D}$ is a diagonal scaling matrix with positive elements. In the third formulation, this feasibility step is restricted by the inequality constraints.

The 'optimality' step p_t also depends on the choice of the dependent variables since the approximate reduced Hessian changes with the choice of the dependent variables. Therefore all these reduced QP formulations depend on the choice of the design variables.

These decomposition methods reduce to the reduced gradient method when the current estimate satisfies the equality constraints sufficiently accurately.

Other reduced models are still possible.

3.2.2 Termination Criteria

It is observed that when the step p_s generated from any of the three reduced models *RQP C.I*, *RQP C.II* and *RQP C.III* approaches zero, the current estimate approaches a point in the feasible region. Also the optimality conditions of these reduced quadratic models approach those of the original optimisation problem when the step length d_x generated from any of these models approaches zero. Therefore, it is reasonable to terminate the process when the following condition is satisfied.

$$Tol \leq \epsilon \quad (3.22)$$

where

$$Tol = \max \{ p_{n_k}^T Z_k^T B_k Z_k p_{n_k}, \sum_{i=1}^{meq} h_i(x_k) | + \sum_{j=1}^{mi} \max(0, -g_j(x_k)) \},$$

$$\nabla_x f(x_k)^T d_k + \sum_{i=1}^{meq} |\lambda_{i_k} h_i(x_k)| + \sum_{j=1}^{mi} |\nu_{j_k} g_j(x_k)| \quad (3.23)$$

and ϵ is a small positive number.

The second term in the right hand side of equation (3.23) checks whether the iterates satisfy all the constraints and the third term checks whether the iterates satisfy the optimality conditions.

3.2.3 Global Convergence

The search direction calculated from the reduced QP subproblem may not yield a direction towards a solution of the original problem when the starting point is far away from it. In general, as discussed in Chapter 2, there are two ways to force the current point converge to the solution from the remote points : line search methods and trust region methods. Only the line search method is considered here.

3.2.3.1 Line Search Methods

Two merit functions used in this study are the absolute value penalty function of the form,

$$P_e(x) = f(x) + \sum_{i=1}^{meq} r_i |h_i(x)| + \sum_{j=1}^{mi} r'_j \max(0, -g_j(x)) \quad (2.20)$$

and the augmented Lagrangian function of the form,

$$P_a(x) = f(x) - \sum_{i=1}^{meq} \lambda_i h_i(x) - \sum_{j=1}^{mi} \nu_j g_j(x) +$$

$$\frac{1}{2} \left\{ \sum_{i=1}^{meq} r_i h_i(x)^2 + \sum_{j=1}^{mi} r'_j \left[\max(0, -g_j(x) - \frac{\nu_j}{r'_j})^2 - \nu_j^2 \right] \right\} \quad (2.29)$$

where r and r' are penalty parameters. It can be proved that the search directions generated from the above three reduced models are descent directions for both merit functions under certain conditions.

With the absolute value penalty function, as mentioned previously, the main difficulty is that the acceptable step length $\|d_x\|_2$ may become very small when the penalty parameters r and r' are too large. This inefficiency might occur even when the current point is near the solution (Maratos effect) [Maratos, 1978]. Many approaches have been proposed to deal with this situation, as discussed in Chapter 2. Here, the following strategy is used. The full step length is accepted when the following conditions are met :

$$\nabla_x f(x)^T d_{x_k} + \frac{1}{2} p_{n_k}^T Z_k^T B_k Z_k p_{n_k} \leq 0$$

$$f(x_{k+1}) \leq f(x_k)$$

and

$$(p_{n_k} + p_{z_k})^T Z_k^T B_k Z_k (p_{n_k} + p_{z_k}) \leq \delta \quad (3.24)$$

where δ is a small number.

The first condition ensures that the search direction is a descent direction for the absolute value penalty function and the objective function, even when the penalty parameters equal zero. The second condition states that the objective function actually decreases and the third condition suggests that the step length for the design variables be small. This strategy attempts to balance the reduction of the value of the objective function and the violation of the constraints. However, these conditions are awkward when the number of the active constraints is equal to the number of variables. In this case, a line search is performed but the objective function term in equation (2.20) is neglected. That is, the line search objective function is switched to the following form :

$$P_m(x) = \sum_{i=1}^{meq} r_i |h_i(x)| + \sum_{j=1}^{mi} r'_j \max(0, -g_j(x)) \quad (3.25)$$

In this case, there are no degrees of freedom and the strategy is simply to limit the violation of the constraints.

The penalty parameters are updated as follows

$$\begin{aligned} r_{i_{k+1}} &= \max \left(\lambda_{i_k}, \frac{1}{2} (|\lambda_{i_k}| + r_{i_k}) \right), \quad i = 1, \dots, meq \\ r'_{j_{k+1}} &= \max \left(|\nu_{j_k}|, \frac{1}{2} (|\nu_{j_k}| + r_{j_k}) \right), \quad j = 1, \dots, mi \end{aligned} \quad (2.27)$$

This update scheme ensures that the penalty parameters are larger than the estimated Lagrangian multipliers. It is noted that the penalty parameters are allowed to decrease in order to reduce the inefficiency.

When the augmented Lagrangian function is used as a merit function, the Lagrangian multipliers may be regarded as independent variables, as suggested by Yamashita [1982] and Schittkowski [1983] or, alternatively, they can be viewed as functions of the optimisation variables, as proposed by Powell and Yuan [1986]. In this study, the Lagrangian multipliers are regarded as constants during a line search. An argument for the use of this type of line search function is that the following augmented Lagrangian function with sufficiently large penalty parameters r and r' has the same optimal point as the original problem :

$$P_l(x) = f(x) - \sum_{i=1}^{meq} \lambda_i^* h_i(x) - \sum_{j=1}^{mi} \nu_j^* g_j(x) + \frac{1}{2} \left\{ \sum_{i=1}^{meq} r_i h(x)_i^2 + \sum_{j=1}^{mi} r'_j \left[\max(0, -g_j(x) - \frac{\nu_j^*}{r'_j})^2 - \nu_j^{*2} \right] \right\} \quad (3.26)$$

where the multipliers λ^* and ν^* are the Lagrangian multipliers at the optimal point. In general, since the actual Lagrangian multipliers are not known in advance, an optimal point of this merit function with estimated Lagrangian multipliers will not be an optimal point of the original problem. However, when the sequence of Lagrangian multipliers converges to the optimal one and the penalty parameters are sufficiently large, this merit function may provide a point which is an improved estimate of the optimal solution. The choice of the penalty parameters for this augmented Lagrangian function needs information on the curvature of the objective function and constraints, which is in general difficult to obtain. Here, the penalty parameters are updated in the following way :

$$\begin{aligned} r_{i_{k+1}} &= \max \left(2 r_{i_k}, 2 | \lambda_{i_k} |, \frac{1}{p_{n_k}^T Z_k^T B_k Z_k p_{n_k}} \right), \quad i = 1, \dots, meq \\ r'_{j_{k+1}} &= \max \left(2 r'_{j_k}, 2 | \nu_{j_k} |, \frac{1}{p_{n_k}^T Z_k^T B_k Z_k p_{n_k}} \right), \quad j = 1, \dots, mi \end{aligned} \quad (3.27)$$

It is expected that the penalty parameters will be sufficiently large when the current estimates are near the solution.

3.2.3.2 Feasibility Improvements

It is noted that this class of decomposition methods either ignores the matrix $Y^T B Z$ and hence the vector $Y^T B Z h(x)$ (reduced model I) or approximates the matrix $Y^T B Z$ by $Y^T Z$ (reduced model II and III). This feature distinguishes this decomposition method from the full Hessian SQP method. In fact, this matrix can be safely ignored when all the constraints are linear or when all the dependent variables are linear. For a nonlinearly constrained system, this suggests that the violation of the constraints should not be too large. This can be achieved by line searches with sufficiently large penalty parameters as described above, since large penalty parameters tend to force the point to satisfy the equality constraints more closely. Alternatively, a feasibility improvement step can also be introduced to achieve this. However, there is no intention to force the current point to become strictly feasible, since inefficiencies may then occur. Good convergence properties can be expected when the following condition is met,

$$\lim_{k \rightarrow \infty} \frac{\|p_{s_k}\|}{\|p_{t_k}\|} \rightarrow 0 \quad (3.28)$$

where p_s is the ' feasibility step ' and p_t is the ' optimality step '. In this study, the following formulation QP C.III is used to calculate the feasibility step s_l at the l^{th} step of a feasibility improvement iteration,

$$\begin{aligned} \min_{s_l} \quad & s_l^T \underline{D} s_l \\ \text{QP C.III} \quad & \text{s.t. } h(x_{k+1+l}) + \nabla_x h(x^+)^T s_l = 0 \\ & g(x_{k+1+l}) + \nabla_x g(x^+)^T s_l \geq 0 \end{aligned} \quad (3.29)$$

where

$\underline{D}$ is a diagonal scaling matrix with positive elements.

$s_0 = 0$.

$x_{k+1+l} = x_{k+1} + s_{l-1}$, for $l \geq 1$.

x^+ can be selected as either x_{k+1} or x_{k+1+l} .

The same decomposition technique previously used is applied to solve QP C.III iteratively. The iteration is terminated when

$$\sum_{i=1}^{meq} |h_i(x_{k+1+l})| + \sum_{j=1}^{mi} \max(0, -g_j(x_{k+1+l})) \leq \gamma Tol \quad (3.30)$$

or alternatively,

$$p_{s_k} \leq \gamma p_{t_k} \quad (3.31)$$

where Tol is the convergence criterion, equation (3.23), and $0 \leq \gamma \leq 1$.

This iterative process of improving feasibility proceeds only when

$$\|s_{l+1}\|_2 < \|s_l\|_2$$

and is performed after the line search.

A basic algorithm for a line search with feasibility improvements is given as follows :

Algorithm C.II — Line Search Followed by Feasibility Steps

1. Define a merit function $P_m(x_k, \cdot)$ as either equation (2.20) or equation (2.29).
2. Update parameters for the merit function, using equation (2.27) or (3.27) if necessary.
3. Find the step length parameter α_l such that
$$P_m(x_k + \alpha_l d_{x_k}, \cdot) - P_m(x_k, \cdot) \leq \alpha_l \beta \nabla_x P_m(x_k, \cdot)^T d_{x_k}$$
4. Solve equation (3.29) for the feasibility step s_l until the termination conditions, equation (3.30) or equation (3.31) are met.
5. Set $x_{k+1} = x_k + \alpha_l d_{x_k} + \sum_{l=1}^{nl} s_l$
where nl is number of feasibility steps.

3.2.4 Local Convergence and Reduced Hessian Update

Powell [1978 (a)] showed that the methods that use the reduced (projected) Hessian are two-step superlinearly convergent under certain conditions. Similar results were given by Nocedal and Overton [1985], and Coleman and Conn [1984], however, Yuan [1985 (b)] and Byrd [1985] gave examples which showed irregular convergence. Gabay [1982] showed that the method with the feasibility improvements is one-step superlinearly convergent. The same result was also presented by Bertsekas [1982]. These results imply that the convergence rate is slower when the matrix $Z^T B Y$ is ignored. However, the convergence rate can be improved by the introduction of the feasibility improvement. Byrd and Schnabel [1986] pointed out that the matrix Z may not be a continuous function of variables x when the matrix Z is obtained from QR factorization. The continuity property of the associated Z is crucial in proving local convergence for this decomposition method. Coleman and Sorensen [1984] and Gill *et al.* [1985 (a)] proposed some techniques to deal with discontinuities. Discontinuities can also occur when the the matrix Z is generated from the LU decompositions

because an inverse of the matrix $\nabla_x h(x)$ is involved. A technique for dealing with the singularity of $\nabla_x h(x)$ will be discussed in the section (3.3.4).

Under the assumption that the term $Z_k^T B_k Y_k h(x_k)$ can be ignored, the reduced Hessian of the Lagrangian function $Z_k^T B_k Z_k$ can be updated by choosing a matrix $Z_{k+1}^T B_{k+1} Z_{k+1}$ to satisfy the least change secant relation

$$\begin{aligned} \min_{Z^T B Z} \quad & \| Z_{k+1}^T B_{k+1} Z_{k+1} - Z_k^T B_k Z_k \|_2 \\ \text{s.t.} \quad & Z_{k+1}^T B_{k+1} Z_{k+1} (p_{n_h} + p_{x_h}) = y_k \\ & (Z_{k+1}^T B_{k+1} Z_{k+1})^T = (Z_k^T B_k Z_k) \end{aligned} \quad (3.32)$$

The change in the reduced gradients of the Lagrangian function y_k may be chosen as

$$\begin{aligned} y_k = Z(x_{k+1})^T \{ & (\nabla_x f(x_{k+1}) - \nabla_x g(x_{k+1}) \nu_k) \\ & - (\nabla_x f(x_k) - \nabla_x h(x_k) \lambda_k - \nabla_x g(x_k) \nu_k) \} \end{aligned} \quad (3.33)$$

However, the following expression for y_k is chosen in this work.

$$\begin{aligned} y_k = & Z(x_{k+1})^T (\nabla_x f(x_{k+1}) - \nabla_x h(x_{k+1}) \lambda_k - \nabla_x g(x_{k+1}) \nu_k) \\ & - Z(x_k)^T (\nabla_x f(x_k) - \nabla_x h(x_k) \lambda_k - \nabla_x g(x_k) \nu_k) \\ = & Z(x_{k+1})^T (\nabla_x f(x_{k+1}) - \nabla_x g(x_{k+1}) \nu_k) \\ & - Z(x_k)^T (\nabla_x f(x_k) - \nabla_x g(x_k) \nu_k) \end{aligned} \quad (3.34)$$

The reason for using equation (3.34) instead of equation (3.33) is that the latter expression requires less storage and does not depend on the Lagrangian multipliers for the equality constraints. This can alleviate difficulties when the Lagrangian multipliers are not uniquely defined. However, equation (3.34) implies the condition [Powell, 1987]

$$d_{x_h} = p_s + o(|d_{x_h}|). \quad (3.35)$$

Therefore, the residuals of the active constraints should not be too large.

Among least change secant update formulae, the superiority of the BFGS update, equation (2.56), is well accepted and this update is therefore used for the present work.

In order to guarantee a unique solution to the reduced QP subproblem and the global convergence, it is desirable to keep the reduced Hessian of the Lagrangian

function positive definite. However, the reduced Hessian may not be positive definite during iterations since some inequality constraints may become active. There are many ways to tackle such difficulties by taking advantages of the hereditary positiveness property of the *BFGS* update, as described in Chapter 2. Since in practice Powell's modification of the *BFGS* update performs well, this modification is used. However, Powell [1985 (b)] showed that his modification may result in ill-conditioning of the Hessian matrix. Therefore, some further modifications are made for the present study.

1. The *BFGS* formula is discontinuous when the following condition occurs.

$$p_n^T Z^T B Z p_n = 0. \quad (3.36)$$

This can occur when all the decision variables are on their bounds. Unlike with the full *SQP* method, this is not the solution of the original problem unless the current point satisfies all the constraints. No update for the reduced Hessian is performed when this condition occurs.

2. In some cases the adjustment for keeping the Hessian approximation positive definite may cause severe ill-conditioning and cause the Hessian to become unbounded. In this case, the reduced Hessian is restarted by an identity matrix. The ill-conditioning is judged from the ratio between the largest diagonal element and the smallest diagonal element. That is, the Hessian is reset to an identity matrix when

$$\frac{\max\{ |(B)_{ii}|, \quad i = 1, \dots, n \}}{\min\{ |(B)_{ii}|, \quad i = 1, \dots, n \}} \geq \rho \quad (3.37)$$

where

$(B)_{ii}$ is a diagonal element of matrix B .

ρ is a large positive number, say 10^{10}

The use of equation (3.37) is based on the assumption that the reduced Hessian $Z^T B Z$ is diagonally dominated and avoids using the inverse of this matrix. The main purpose is to detect the extreme ill-conditioning matrix. Other ways to estimate the conditioning of a matrix can also be used, e.g. based on the estimation of the largest and smallest eigenvalues.

3. Following the suggestions made by Nocedal and Overton [1985] the reduced Hessian will not be updated when the ratio $\frac{\|h(x)\|}{|d|}$ is unacceptably large.

This is because equation (3.34) is of practical value only if the norm of the step p_t is small [Powell, 1987].

3.3 Implementation Considerations

In addition, the efficiency and robustness of an algorithm strongly depend on its implementation and the following factors were considered.

1. Formulation of the optimisation problem.
2. Choice of the initial reduced Hessian.
3. Choice of the design variables.
4. Calculation of the projection basis.
5. Strategy for dealing with inconsistent linearised constraints.
6. Strategy for predicting the likely active inequality constraints..
7. Solution of the reduced QP subproblems.

3.3.1 Formulation of the Optimisation Problem

The general optimisation problem of a form given in equation (2.1) is converted to a problem with equality constraints and simple bounds by the introduction of slack variables. The original problem thus becomes :

$$\begin{aligned}
 & \min_w \quad f(w) \\
 P \ C.I \quad & s.t. \quad c(w) = 0 \\
 & w_\ell \leq w \leq w_u
 \end{aligned} \tag{3.38}$$

where

- w is an $(n + m_i)$ -dimensional vector of variables and is defined as $w^T \stackrel{def}{=} (x^T, e^T)$.
- x is an n -dimensional vector of variables.
- e is an (m_s) -dimensional vector of extra variables.
- w_u is an $(n + m_i)$ -dimensional vector of upper bounds for variables w , and it is defined as $w_u^T \stackrel{def}{=} (x_u^T, \infty)$.
- x_u is an n -dimensional vector of upper bounds for variables x .
- w_ℓ is an $(n + m_i)$ -dimensional vector of lower bounds for variables w , and is defined as $w_\ell^T \stackrel{def}{=} (x_\ell^T, 0)$.
- x_ℓ is an n -dimensional vector of lower bounds for variables x .
- $f(w)$ is a scalar objective function.
- $c(w)$ is an $(m_{eq} + m_i)$ -dimensional vector of transformed equality constraints and is defined as $c(w)^T \stackrel{def}{=} (h(x)^T, (g(x) - e)^T)$.

$h(x)$ is an meq -dimensional vector of equality constraints.
 $g(x)$ is an mi -dimensional vector of inequality constraints.
 meq is the number of equality constraints.
 mi is the number of inequality constraints.

This formulation increases the number of the total variables by (mi) but not that of the design variables. Hence this formulation does not affect the size of the reduced Hessian. This representation avoids manipulation of two sparse matrices. It is noted that a square variable transformation, i.e.

$$g(x) - \epsilon^2 = 0$$

which attempts to eliminate the inequality constraints completely is not appropriate for this method because divergence of the optimisation often occurs unless all estimates of w are within the feasible region of the inequality constraints, as explained in chapter 2.

3.3.2 Choice of the Initial Reduced Hessian

The initial reduced Hessian in RQP C.II $(Z^T B Z)_0$ may be chosen as the identity matrix I a diagonal matrix or the matrix $(Z^T Z)_0$. The choice of an identity matrix I is equivalent to choosing the initial Hessian matrix B_0 as

$$B_0 = \begin{pmatrix} O & O \\ O & I \end{pmatrix} \quad (3.39)$$

Alternatively, the reduced Hessian matrix B_0^+ can also be initialised as

$$B_0^+ = \begin{pmatrix} O & O \\ O & D^+ \end{pmatrix} \quad (3.40)$$

where D^+ is a diagonal matrix based on the order of the magnitude of the design variables as used in equation (2.65). The choice of the matrix $(Z^T Z)_0$ is equivalent to choosing an identity matrix for the initial Hessian matrix, B_0 . Vasantharajan and Biegler [1987] showed the effectiveness of the second strategy only on few simple mathematical problems. They failed to produce the same results for chemical process problems. Our numerical experience show that the performance of the decomposition algorithm is disappointing when the initial Hessian is chosen as the matrix $(Z^T Z)_0$. This choice for the initial reduced Hessian may result in an ill-conditioned reduced Hessian matrix and a small step p_s may occur and hence it will not be considered further. Therefore an identity matrix is used as the initial reduced Hessian in this work.

3.3.3 Choice of the Design Variables

In principle, $(n - meq)$ design variables may be assigned by users. An automatic selection of design variables can also be obtained by employing an $L U$ decomposition of the rectangular Jacobian matrix $\nabla_x h(x)$. This is achieved by using the sparse linear equation solver *MA28A* [Duff, 1977]. This also allows the detection of a possible structural singularity of the matrix $\nabla_x h(x)$ which may be an indication of an ill-posed problem. This option is automatically invoked when a set of design variables assigned by users results in structural singularities of the system. When the rank of this rectangular matrix is less than the number of equality constraints, the system is assumed to be ill-posed and the algorithm terminates.

For a system with a large number of linear variables, particularly when the number of nonlinear variables is less than the number of degrees of freedom of the system, it is desirable to factorise the rectangle Jacobian matrix based on the linearity, i.e., to factorise the Jacobian matrix into a lower triangular form such that the elements in the diagonal blocks are constant whenever it is possible. The variables corresponding to the diagonal elements can hence be chosen as dependent variables.

For example, let us consider a system of equality constraints of the form,

$$h(x) = A \bar{x} + \hat{h}(\hat{x}) = 0$$

where

$$x = (\bar{x}, \hat{x}).$$

$\hat{x}$ is an $\hat{n}$ -dimensional vector of nonlinear variables.

$\bar{x}$ is an $\bar{n}$ -dimensional vector of linear variables.

A is an $(meq \times \bar{n})$ coefficient matrix.

$\hat{h}$ is an $\hat{n}$ -dimensional vector of nonlinear functions.

The number of nonlinear variables is assumed to be no greater than degree of the freedom, i.e.

$$\hat{n} \leq (n - meq).$$

It is preferable to choose all these nonlinear variables as design variables since this merely means

$$\bar{x}' = -(A')^{-1} (A'' \bar{x}'' + \hat{h}(\hat{x}))$$

where

$$\begin{aligned}
A &= (A' \ A'') \\
\bar{x} &= (\bar{x}' \ \bar{x}'') \\
A' &\text{ is assumed to be nonsingular.}
\end{aligned}$$

In practice, this can be achieved by the decomposition algorithms suggested by Pantelides [1988 (b)].

It is not appropriate to choose slack variables as design variables since the inequality constraints corresponding to these slack variables may not be active at the solution.

3.3.4 Construction of the Projection Matrix

In practice, the matrices Y and Z can be obtained by solving the following augmented system via an $L U$ decomposition.

$$\begin{pmatrix} \nabla_x h(x)^T \\ \Lambda^T \end{pmatrix} (Y \ Z) = I \quad (3.41)$$

The matrix Λ is an $(n \times (n - meq))$ permutation matrix with exactly one 1 in each row and column corresponding to the position of a design variable, and zero elsewhere. The advantages of this representation are that no fixed input structure is required; the product $Z p_n$ and the product $Y h(x_k)$ are in the 'right' form and the choice of design variables can easily be changed. Equations (3.41) are again solved by the sparse linear solver *MA28A* at the very first iteration. The sparse routine *MA28B* [Duff, 1977] is used for the subsequent iterations to avoid refactorising the matrix

$$\begin{pmatrix} \nabla_x h(x)^T \\ \Lambda^T \end{pmatrix}.$$

This matrix may be singular during the iterations due to numerical reasons. If a numerical singularity is detected during the factorisation, the resulting factorised matrix has a null sub-matrix. In this case, an identity matrix replaces the null sub-matrix after the factorisation. The matrix Y is implicitly stored to save storage space.

3.3.5 Strategy for Dealing With Inconsistent Constraints

When constraints are inconsistent, due to the local linearisation, the underlying reduced QP will fail. Many suggestions for overcoming this difficulty have been discussed in the previous chapter. To avoid unnecessary ill-conditioned problems, no modifications are made when the constraints in the reduced QP subproblems $RQP\ C.I$ and $RQP\ C.II$ are consistent. Powell's modification [1978 (a)] is used if the constraints are inconsistent, however, as discussed in Chapter 2, this may fail even when the original constraints are consistent.

The following modification is preferred when Powell's modification fails.

$$\begin{aligned} \min_{p_{n_k} \ r} & \nabla_x f(x_k)^T Z_k p_{n_k} + \frac{1}{2} p_{n_k}^T Z_k^T B_k Z_k p_{n_k} + R(r) \\ \text{s.t.} & (g(x_k) + \nabla_{x_k} g(x_k)^T p_{s_k}) + \nabla_x g(x_k)^T Z_k p_{n_k} \geq -r \\ & r \geq 0 \end{aligned} \quad (3.42)$$

where

$R(r)$ can be selected either as $\frac{\tau}{2} r^2$ or $\tau' r$.
 τ, τ' can be set as a positive number, say, 10^3 .

This modification is equivalent to solving the following system

$$\begin{aligned} \min_{p_{n_k}} & \nabla_x f(x_k)^T Z_k p_{n_k} + \frac{1}{2} p_{n_k}^T Z_k^T B_k Z_k p_{n_k} + \\ & \frac{\tau}{2} \max(0, -(g(x_k) + \nabla_{x_k} g(x_k)^T p_{s_k} + \nabla_x g(x_k)^T p_{n_k}))^2 \end{aligned} \quad (3.43)$$

or

$$\begin{aligned} \min_{p_{n_k}} & \nabla_x f(x_k)^T Z_k p_{n_k} + \frac{1}{2} p_{n_k}^T Z_k^T B_k Z_k p_{n_k} + \\ & \tau' \max(0, -(g(x_k) + \nabla_{x_k} g(x_k)^T p_{s_k} + \nabla_x g(x_k)^T p_{n_k})) \end{aligned} \quad (3.44)$$

The original constraints are considered to be inconsistent when the penalty parameter r is unacceptably large, say 10^5 . This approach is found to be effective especially when those constraints are ' soft ', i.e. constraints violation may be tolerated.

3.3.6 Strategy for Predicting the Likely Active Inequality Constraints

The simple bounds for the dependent variables become general linear constraints when the decomposition technique is used. Since all the dependent variables are assumed to be bounded, it is required to store a $2 \times (n - meq) \times meq$ Jacobian matrix which may become very large. It is observed, however, that at most an equal number of constraints as that of decision variables will become active. Therefore only the 'most likely active' constraints are included within the reduced QP subproblems in order to increase efficiency and to reduce the storage requirements. It is also desirable to include as many constraints as possible when the trial point is far away from the solution. However, only the active constraints need to be considered when the trial points are near the solution. The 'most likely active' constraint set used by Mine *et al.* [1984] is defined as

$$K_{\epsilon}(x) = \{ i \in \{1, \dots, mi\} \mid g_i(x) + \max(0, -g_i(x)) \leq \epsilon \}. \quad (3.45)$$

where

$\epsilon = \max(\zeta \text{ Tol}, \eta)$ with ζ a large number, say, 10^4 .

η is a small number, say, 10.

Tol is the defined termination criterion, equation (3.22).

3.3.7 Solution of the Reduced QP Subproblem

The reduced QP subproblem is solved by the ZQPCVX subroutine [Powell, 1983] described in Chapter 2. This dual method is in particular effective to solve the reduced QP since few constraints will become active.

3.4 Complete SRQP Algorithm

This decomposition method is called successive reduced quadratic programming method (SRQP) since it requires the iterative solution of reduced programming subproblems RQP C.II and RQP C.III. A complete algorithm for SRQP method can finally be stated as follows.

Algorithm C.III — Complete SRQP Algorithm

Begin < Initialisation Section >

 Given an initial estimate x_0 .

 Given a tolerance ϵ .

 Given maximum number of iterations allowed max_f .

 Given maximum feasibility steps allowed max_s .

 Choose design variables manually, i.e. assign Λ in equation (3.41),
 or automatically.

 Choose a reduced model as reduced model I, II, or III.

 Choose a merit function as either equation (2.20) or (2.29).

 Choose an initial Hessian as an identity matrix or a diagonal matrix.

 Set $k = 0$.

End

Repeat

 Set $k = k + 1$.

Begin < Construct and solve reduced QP >

 Solve equation (3.41) for matrices Y and Z .

 Predict the likely active set K_ϵ (equation (3.45)).

If reduced model I **Then**

 solve equation (3.10), (3.11), (3.12), and (3.13).

Else_If reduced model II **Then**

 solve equation(3.10), (3.17), (3.18), and (3.19).

Else_If reduced model III **Then**

 solve equation(3.10), (3.19), (3.20) and (3.21).

End If

If Failed **Then** solve equation (3.42).

End

Begin < Check Convergence Criteria >

If inequality (3.22) is satisfied **Then** *Exit. (Solution Found)*.

End

Begin < Perform Line Search >

 Perform Algorithm C.II.

End

Begin < Update the reduced Hessian >

 Equation (3.34), (2.56), (2.58), (2.60).

End

Until $k \geq max_f$.

3.5 Comparison With Other Optimisation Methods

In this section, the *SRQP* method is compared with its related methods including feasible direction methods and the projected Lagrangian method, based on a method due to Robinson [1972] and implemented in the *MINOS* optimisation code.

3.5.1 Comparison with Feasible Direction Methods

The *SRQP* method can be viewed as a relaxed version of the feasible direction methods including *GRG* methods and projection methods. There is no intention for the *SRQP* method to satisfy the equality constraints to a tight tolerance. This avoids a possible failure in attempting to solve the equality constraints for certain values of the decision variables.

The *SRQP* method uses the solution of a reduced *QP* subproblem to predict the active inequality constraints. This avoids 'jamming' and 'zigzagging' problems which may otherwise occur.

In short, the *SRQP* method avoids the difficulties which may occur in the feasible direction methods and improves the efficiency and robustness of these methods.

3.5.2 Comparison with an Optimisation Code - *MINOS*

MINOS attempts to solve large scale optimisation problems of the form :

$$\begin{aligned}
 \min_{\hat{x}} \quad & f(\hat{x}) + a_1^T \hat{x} + a_2^T \bar{x} \\
 \text{s.t.} \quad & \hat{h}(\hat{x}) + A_1 \bar{x} = b_1 \\
 & A_2 (\hat{x}) + A_3 \bar{x} = b_2 \\
 & \hat{x}_\ell \leq \hat{x} \leq \hat{x}_u \\
 & \bar{x}_\ell \leq \bar{x} \leq \bar{x}_u
 \end{aligned} \tag{3.46}$$

where

- $\hat{x}$ is an $\hat{n}$ -dimensional vector of nonlinear variables.
- $\hat{x}_\ell$ is an $\hat{n}$ -dimensional vector of lower bounds for nonlinear variables $\hat{x}$.
- $\hat{x}_u$ is an $\hat{n}$ -dimensional vector of upper bounds for nonlinear variables $\hat{x}$.
- $\bar{x}$ is an $\bar{n}$ -dimensional vector of linear variables.
- $\bar{x}_\ell$ is an $\bar{n}$ -dimensional vector of lower bounds for linear variables $\bar{x}$.
- $\bar{x}_u$ is an $\bar{n}$ -dimensional vector of upper bounds for linear variables $\bar{x}$.
- f is a scalar objective function.

D is an m -dimensional vector of nonlinear constraints.
 a_1 and a_2 are vectors of coefficients.
 b_1 and b_2 are vectors of residuals.
 A_1, A_2 , and A_3 are coefficient matrices.

The subproblem to be solved at the k^{th} iteration takes the form

$$\begin{aligned}
 \min_{\hat{x}, \bar{x}} \quad & f(\hat{x}) + a_1^T \hat{x} + a_2^T \bar{x} - \hat{\lambda}_k^T (\hat{h} - \hat{h}') + \\
 & \frac{1}{2} \rho (\hat{h} - \hat{h}')^T (\hat{h} - \hat{h}') \\
 \text{s.t.} \quad & \hat{h}(\hat{x}) + A_1 \bar{x} = e_1 \\
 & A_2 (\hat{x}) + A_2 = e_2 \\
 & \hat{x}_\ell \leq \hat{x} \leq \hat{x}_u \\
 & \bar{x}_\ell \leq \bar{x} \leq \bar{x}_u
 \end{aligned} \tag{3.47}$$

where

ρ is a penalty parameter.
 $\hat{\lambda}$ is an m -dimensional vector of Lagrangian multipliers estimates for the nonlinear constraints.
 $\hat{h}' = \hat{h}(\hat{x}_k) + \nabla_{\hat{x}} \hat{h}(\hat{x}_k) (\hat{x} - \hat{x}_k)$

Each subproblem is solved by a reduced gradient method [Murtagh and Saunders, 1982]. This method, in some sense, attempts to approach a minimum point of the augmented Lagrangian function before restoring the feasibility. It is effective when the constraints are all linear or mildly nonlinear. For optimisation problems with nonlinear constraints, a subproblem may be unbounded unless the penalty parameter ρ is sufficiently large. This may be the case when there are large numbers of nonlinear equality constraints. A large penalty parameter can impede the progress of this method since the penalty term $(\hat{x} - \hat{x}_k)^T \nabla_{\hat{x}} \hat{h}(\hat{x}_k)^T \nabla_{\hat{x}} \hat{h}(\hat{x}_k) (\hat{x} - \hat{x}_k)$ becomes dominant. The efficiency of this method therefore depends on the choice of the penalty parameter. In contrast, feasible path direction methods restore feasibility before approaching the minimum point of the objective function. The idea of the *SRQP* method is to restore the feasibility with respect to the constraints and to approach a local solution of the optimisation problem at the same time.

3.6 Numerical Experiences

The proposed decomposition algorithm is implemented in an optimisation code which is structured and has the same features as the test program described in Chapter 2.

The objectives of the tests presented in this section are to demonstrate principles underlying the decomposition method, to show the differences among three reduced models and the effect of the choice of the decision variables and the initial Hessian. More comprehensive tests on large scale problems are presented in Chapter 4.

Three test problems are employed. The first problem used to show the difference among the reduced models is a modified version of *M116*

$$\begin{aligned} \min_{x_1, x_2} \quad & -x_1 \\ \text{s.t.} \quad & x_1^2 + x_2^2 = 1 \\ & 0 \leq x_1 \leq 1 \\ & 0 \leq x_2 \leq 1 \\ & \text{starting point } (0.2, 0.5) \end{aligned}$$

The second test problem is an alkylation process, which was suggested by Sauer *et al.* [1960]. The flowsheet of this process is shown in Figure 3.1 and is described in appendix C.1. This well known test problem has a large number of linear variables and hence the structure of this problem can be exploited.

The third test problem is the *Williams-Otto process* [Williams and Otto, 1964]. Figure 3.2 shows the flowsheet of the process which is described in appendix C.2. This is also a well known test problem and is a highly nonlinear problem. Thus it can be used to test the efficiency of the decomposition method.

The default options of the algorithm used in this study are

1. The reduced Hessian of the Lagrangian function is initialised to an identity matrix.
2. The reduced model I is used.
3. An absolute tolerance of 10^{-7} is used.
4. An absolute value penalty merit function, equation (2.20), with the relaxation strategy (equation (3.24)) is employed.

5. No feasibility improvements are performed.

3.6.1 Comparison of Reduced Models and Choice of Design Variables

Figure 3.3 illustrates the search directions at the first iteration generated from the three reduced models.

It can be observed that the 'feasibility step' p_s of the *reduced model I* defines the least distance to linearised equality constraints with search direction of design variables set to zero. The 'feasibility step' p_s of the *reduced model II* defines the least distance to linearised equality constraints and that of the *reduced model III* is the step to the nearest point on linearised nonlinear constraints. The 'optimality step' p_t is a step from $x + p_s$ to the minimum of the quadratic model of the Lagrangian function in the subspace orthogonal to the constraint normals. In fact, in this example, all these models produce the same search direction since the number of active constraints equals to that of variables.

Figure 3.4 shows the structure of the Jacobian matrix for the second test problem and Figure 3.5 for the third test problem. Table 3.1 shows the performance of these three models with a different choice of design variables for the second test problem and Table 3.2 for the third test problem.

These numerical results show that the performance of the *SRQP* method with all the reduced models depends on the choice of design variables and that the choice of the linear variables as dependent variables improves the performance of the algorithm.

To test the proposed strategy for dealing with a structural singularity due to an improper choice of design variables, the variables X_8 , X_9 , and X_{10} are initially chosen as design variables for the alkylation process. The routine detects a structural singularity and re-assigns the variables X_1 , X_3 and X_4 as design variables. The code locates an optimum after 12 iterations with *reduced model III*. When the variables FRC , FRE , FRP , and FP are initially chosen as design variables, the routine re-assigns FG , FR , FA , and FB as design variables and reaches an optimal solution after 94 iterations with *reduced model III*.

Figure 3.1 Alkylation Process Flowsheet

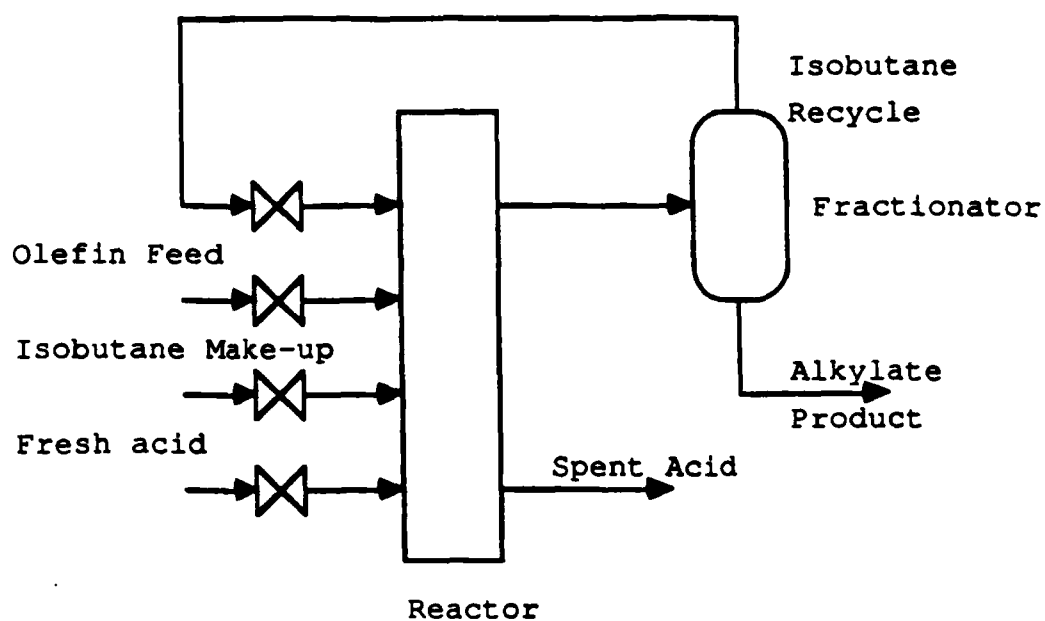


Figure 3.2 Williams-Otto Process Flowsheet

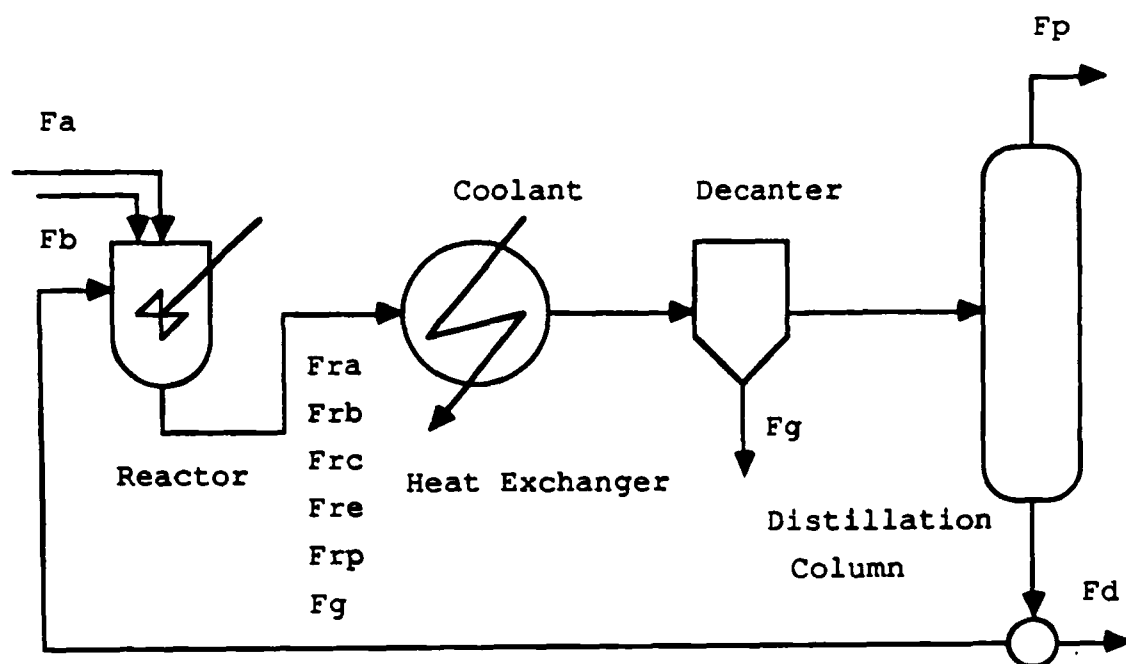
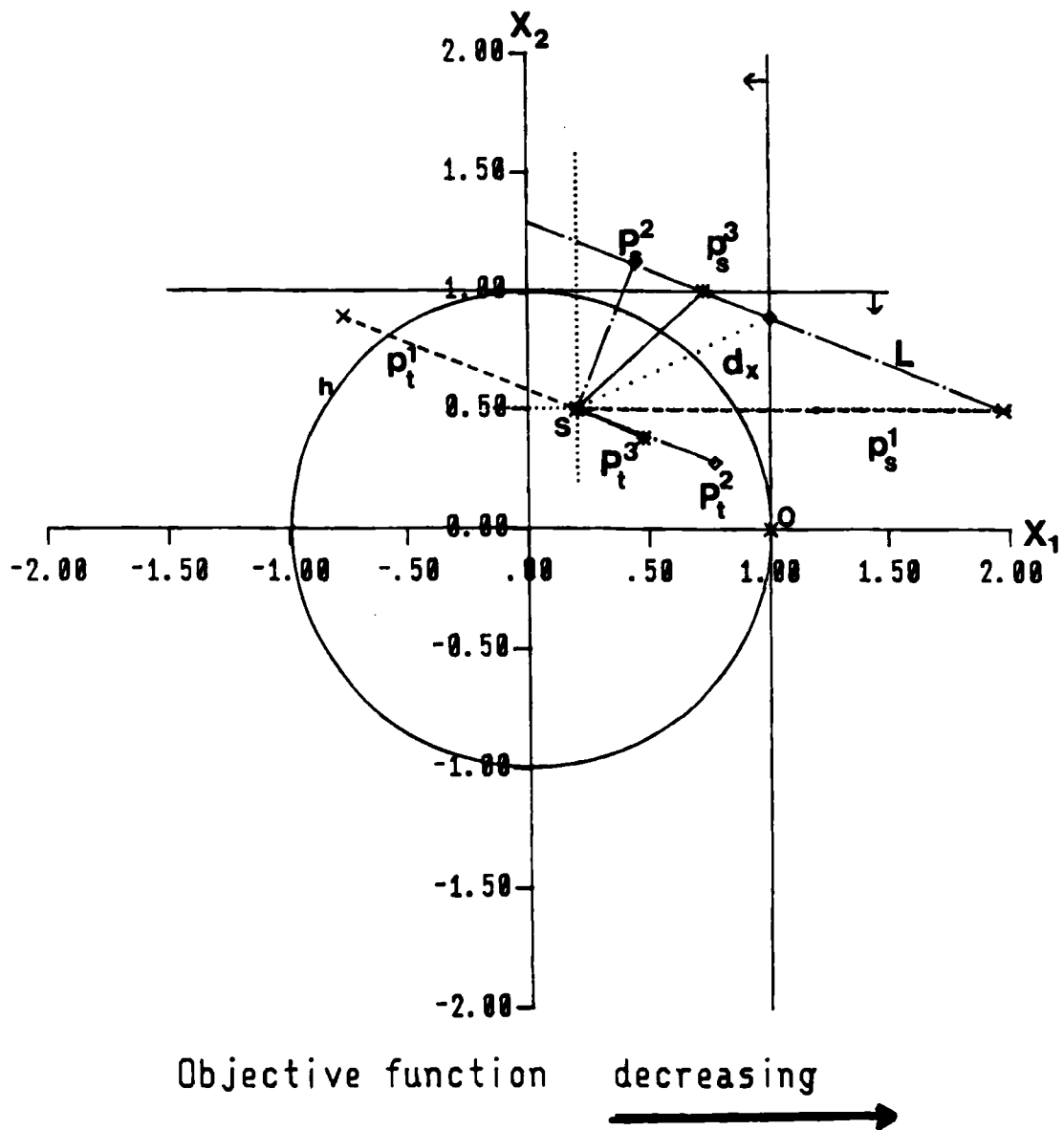


Figure 3.3 Comparison of Three Reduced Models



Key (Figure 3.3)

Problem		
	— objective function $(\min_{x_1, x_2})$	$-x_1$
circle	h — constraints (s.t.)	$x_1^2 + x_2^2 - 1 = 0$
	— simple bound	$x_1 \leq 1$
	— simple bound	$0 \leq x_1$
	— simple bound	$x_2 \leq 1$
	— simple bound	$0 \leq x_2$
point	s — starting point	$(x_1^0, x_2^0) = (0.2, 0.5)$
point	o — solution	$(x_1^*, x_2^*) = (1, 0)$
Approximate model		
circle	f — objective function	$-(p_t)_{x_1} + \frac{1}{2}((p_t)_{x_1}^2 + (p_t)_{x_2}^2)$
line	L — linearised constraints	$0.4(p_s)_{x_1} + (p_s)_{x_2} - 0.71 = 0$
	— simple bound	$-0.2 \leq (p_s)_{x_1} \leq 0.8$
	— simple bound	$-0.5 \leq (p_s)_{x_2} \leq 0.5$
Reduced model I		
line	p_s^1 — feasibility step	
line	p_t^1 — Optimality step	
Reduced model II		
line	p_s^2 — feasibility step	
line	p_t^2 — Optimality step	
Reduced model III		
line	p_s^3 — feasibility step	
line	p_t^3 — Optimality step	
line	d_x — SRQP step	

Figure 3.4 Jacobian Structure of an Alkylation Process

	<i>V</i>	<i>X4</i>	<i>X5</i>	<i>X2</i>	<i>X10</i>	<i>X7</i>	<i>X6</i>	<i>X3</i>		<i>X1</i>	<i>X8</i>	<i>X9</i>
<i>E</i>												
3		<i>L</i>								<i>N</i>	<i>N</i>	
4		<i>L</i>	<i>L</i>							<i>L</i>		
1			<i>L</i>	<i>L</i>						<i>N</i>	<i>N</i>	
6					<i>L</i>							<i>L</i>
7					<i>L</i>	<i>L</i>						
5						<i>L</i>	<i>L</i>				<i>N</i>	
2		<i>N</i>					<i>N</i>	<i>N</i>				<i>N</i>

where

- L* A constant linear element.
- N* A non-constant nonlinear element.
- V* Variable.
- E* Equation.

Figure 3.5 Jacobian Structure of the Williams—Otto Process

	<i>V</i>	<i>FG</i>	<i>FRE</i>	<i>FRC</i>	<i>FRA</i>	<i>FRP</i>	<i>T</i>	<i>FR</i>	<i>FRB</i>		<i>FD</i>	<i>FB</i>	<i>FA</i>	<i>FP</i>
<i>E</i>														
8		<i>L</i>									<i>L</i>	<i>L</i>	<i>L</i>	<i>L</i>
7			<i>L</i>			<i>L</i>								<i>L</i>
4		<i>N</i>	<i>N</i>	<i>N</i>			<i>N</i>	<i>N</i>	<i>N</i>		<i>N</i>			<i>N</i>
2		<i>N</i>			<i>N</i>		<i>N</i>	<i>N</i>	<i>N</i>		<i>N</i>		<i>L</i>	<i>N</i>
5		<i>N</i>		<i>N</i>		<i>N</i>	<i>N</i>	<i>N</i>	<i>N</i>		<i>N</i>			<i>N</i>
6		<i>L</i>		<i>N</i>		<i>N</i>	<i>N</i>	<i>N</i>						
3		<i>N</i>		<i>N</i>	<i>N</i>		<i>N</i>	<i>N</i>	<i>N</i>		<i>N</i>	<i>L</i>		<i>N</i>
1		<i>L</i>	<i>L</i>	<i>L</i>	<i>L</i>	<i>L</i>		<i>L</i>	<i>L</i>					

where

- L* A constant linear element.
- N* A non-constant nonlinear element.
- V* Variable.
- E* Equation.

Table 3.1 Effect of the Choice of Design Variables on the Alkylation Process

Design Variables	RM I NG (NF)	RM II NG (NF)	RM III NG (NF)
X1, X8, X9	9 (9)	13 (14)	13 (14)
X4, X6, X9	11 (12)	16 (17)	17 (20)
X5, X6, X7	14 (15)	16 (21)	14 (19)
X3, X4, X5	20 (20)	18 (18)	18 (18)
X1, X2, X3	26 (28)	25 (27)	25 (27)

Key

RM I The Reduced Model I.
RM II The Reduced Model II.
RM III The Reduced Model III.
NG (NF) Number of gradient (function) evaluations.
Machine IBM 4341 .

Table 3.2 Effect of the Choice of Design Variables On the Williams-Otto Process

Design Variables	RM I NG (NF)	RM II NG (NF)	RM III NG (NF)
FA, FB, FD, FP	54 (61)	59 (87)	59 (83)
T, FA, FB, FD	70 (90)	66 (119)	47 (61)
T, FA, FB, FG	90 (153)	63 (101)	46 (65)
FRB, FRC, FRE, FP	144 (228)	165 (339)	132 (245)
FA, FB, FD, FRE	134 (217)	139 (325)	92 (138)

Key

RM I The Reduced Model I.
RM II The Reduced Model II.
RM III The Reduced Model III.
NG (NF) Number of gradient (function) evaluations.
Machine IBM 4341 .

For the alkylation process, it is intuitive to choose X_1 , X_2 , and X_3 as design variables as suggested by Sauer *et al.* [1964]. However, this may not be the best choice as shown in Table 3.3. Exploiting the structure of the problem may result in a better performance. The automatic choice of the design variables achieves this purpose.

3.6.2 Effect of the Choice of the Initial Hessian

The reduced Hessian is initialised to a diagonal matrix based on the order of magnitude of the variables as in equation (2.65) in this study. Tables 3.3 and 3.4 show the effect of the choice of the initial Hessian. It appears that scaling the initial reduced Hessian is effective in decreasing the number of gradient evaluation for badly scaled problems.

Table 3.3 Effect of the Choice of the Initial Reduced Hessian
On the Alkylation Process

Design Variables	HESS I NG (NF)	HESS II NG (NF)
X_1, X_8, X_9	9 (9)	10 (10)
X_4, X_6, X_9	11 (12)	9 (10)
X_5, X_6, X_7	14 (15)	10 (11)
X_3, X_4, X_5	20 (20)	13 (15)
X_1, X_2, X_3	26 (28)	17 (20)

Key

HESS I	An identity matrix is chosen as initial reduced Hessian.
HESS II	A diagonal matrix is chosen as initial reduced Hessian.
NG (NF)	Number of gradient (function) evaluations.
Machine	IBM 4341 .

Table 3.4 Effect of the Choice of the initial Reduced Hessian
on the Williams-Otto Process

Design Variables	HESS I NG (NF)	HESS II NG (NF)
FA, FB, FD, FP	54 (61)	49 (67)
T, FA, FB, FD	70 (90)	53 (73)
T, FA, FB, FG	90 (153)	76 (133)
FRB, FRC, FRE, FP	144 (228)	107 (171)
FA, FB, FD, FRE	134 (217)	114 (183)

Key

HESS I	An identity matrix is chosen as initial reduced Hessian.
HESS II	A diagonal matrix is chosen as initial reduced Hessian.
NG (NF)	Number of gradient (function) evaluations.
Machine	IBM 4341 .

3.7 Discussion and Conclusions

In this chapter, a class of optimisation methods for large scale optimisation problems is presented and its implementation is discussed. The optimisation step is divided into two components : a step towards the minimum point of the objective function in a reduced space and a step towards the feasible region. This representation is shown to be useful in devising a class of optimisation methods.

This present work shows that when there are many equality constraints and consequently relatively few degrees of freedom, it is possible to update the reduced (projected) Hessian of the Lagrangian function directly and to use a reduced quadratic programming subproblem to predict the active inequality constraints.

It is our experience that a good performance of *SRQP* algorithm depends on careful attention to many algorithmic details. Accordingly, all of these factors are also examined in this chapter, resulting in a particularly robust overall algorithm.

All the projection methods investigated in this chapter are dependent on the choice of the design variables but this choice can be effectively performed by the automatic scheme.

The performance of the decomposition method is improved when a diagonal matrix is used as an initial reduced Hessian. However, this strategy strongly depends on a sensible initial guess of the order of magnitude of the variables, and a partial simulation is recommended to obtain a sensible initial point which reflects the order of magnitude of the variables.

The *reduced model I* and *reduced model II* can be viewed as special cases of the *reduced model III*. Conceptually, they differ in the definition of the 'feasibility step'. The efficiency of the algorithm will be tested in more depth in the next chapter.

Chapter 4

Flowsheet Optimisation

4.1 Introduction

This chapter is concerned with the application of optimisation techniques to chemical processes within a process simulator.

A chemical process consists of process units (e.g. a reactor), and streams which include the inter-connections between units, feeds and products. In general, there are two limiting ways to construct a model for chemical processing units, namely,

1. Modular oriented (*MO*) approach, and
2. Equation oriented (*EO*) approaches.

In the modular oriented approach, a process model contains both a set of model equations (mass, momentum, and energy balance equations etc.) which describe its behaviour, and a solution method for this set of equations. Given values of input streams (e.g. reactant flowrate, temperature, and pressure), values of equipment parameters (e.g. reactor volume), and/or those of unit parameters (eg. reaction rate constant), the output streams (e.g. product flowrate, temperature, and pressure) are then calculated. Thus the input streams should be given or guessed before the unit calculation is performed.

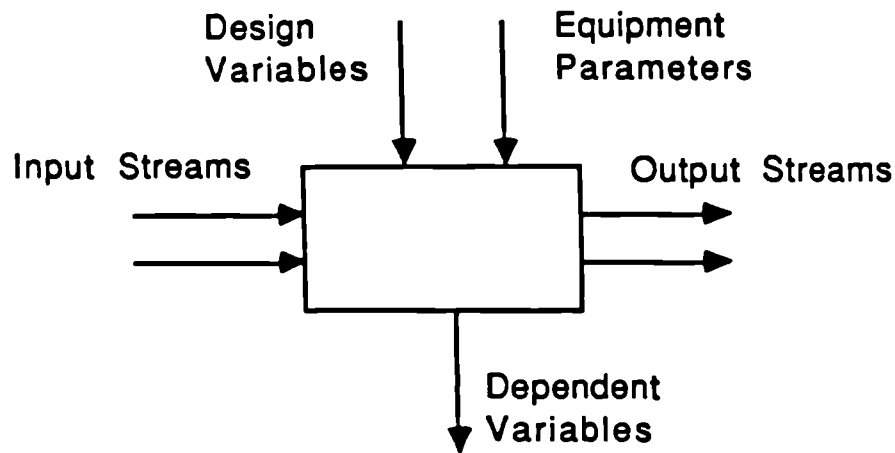
For a simulator based on this approach, partitioning with precedence ordering techniques and the tear criteria are used to select the group of units to be solved together, to decide a proper calculation sequence and to determine a set of streams (tear streams) and/or variables (tear variables) within these streams, which must be guessed within the groups according to the flowsheet topology. It is noted that the calculation sequence becomes unimportant when all the streams and/or unknowns are torn (guessed). At the flowsheet level, a system is solved which contains the tear equations, physical specifications and/or limitations external to the units, and design equations which may be posed by process engineers. The size of the system depends on the tear criteria. Nested level calculations are generated with this approach (e.g. the physical property calculation level, unit calculation level, and the flowsheet calculation level), which are in general expensive to be evaluated. A simulator whose calculation sequence is followed module-by-module

is termed a sequential modular simulator. A simultaneous modular simulator, as its name implies, is one in which all stream are torn and whose modules are evaluated simultaneously at the flowsheet level. Recent surveys of modular oriented simulators can be found in the literature [Evans, 1982], [Biegler, 1983].

In the equation oriented approach, a process unit is only represented by a set of equations which describes the system. For a simulator based on this approach, the model equations for all units, physical specifications and limitations, and design equations are assembled together according to the flowsheet topology and solved simultaneously using a general purpose numerical technique in one level. This approach eliminates the expensive unit calculation level and gives a great flexibility in the range of allowable specifications. The resulting system is in general large. Equation oriented approaches were reviewed by Perkins [1983], Shacham *et al.* [1982], and Westerberg and Benjamin [1985].

A typical module information flow is shown in Figure 4.1.

Figure 4.1 Flowsheet Module



In practice, it is desirable to use some unit modules (procedures) in an equation oriented approach and to solve equations in the modular oriented approach, resulting in simulators which can solve mixed system of equations and procedures [Pantelides, 1988 (a)]. In this case, the system equations at the flowsheet level are made of tear equations, model equations, design equations, and design and/or physical constraints.

Since rigorous models may be expensive to evaluate, approximate (reduced) models may be employed. This approach can be applied to both the modular oriented approach [e.g. Evans 1982] and the equation oriented approach. In this case, the system at the flowsheet level is split in two levels, one which constructs approximate models based on rigorous models, and the other which consists of design equations and/or constraints, tear equations corresponding to the approximate models (for the modular oriented approaches) or approximate model equations (for equation oriented approaches) to perform flowsheet calculations. Criteria are applied to update the approximate models from rigorous models. This approach is often referred to as a two-tier (level) approach [Johns and Vadhwana, 1985], [Evans *et al.*, 1985].

In fact, the modular oriented approach, the equation oriented approach, and the two-level approach can be viewed as different decomposition levels for solving chemical processes [Pantelides, 1988 (b)]. In this thesis, we only consider flowsheet optimisation within the modular and equation oriented approaches. The extension to other approaches (e.g. two-level approach), if required, should be straightforward.

Westerberg [1981] indicated that for flowsheet simulation most of the CPU time is spent in calculating physical properties (approximately 70 %). Since these are usually expensive, unavoidable and must be done at each ' function evaluation ', numerical methods which minimise flowsheet calculations are preferred.

4.2 Flowsheet Optimisation Strategies

A process optimisation problem may be formulated as a nonlinear constrained optimisation problem according to the structure of a simulator:

$$\begin{array}{ll}
\min_{\hat{u}, \hat{v}} & f(\hat{v}, \hat{u}) \\
s.t. & h_p(\hat{v}, \hat{u}) = 0 \\
& h_d(\hat{v}, \hat{u}) = 0 \\
& g_p(\hat{v}, \hat{u}) \geq 0 \\
& g_d(\hat{v}, \hat{u}) \geq 0
\end{array} \tag{4.1}$$

P D.I

where

- $\hat{v}$ is a set of variables representing
 - tear variables for a *MO* approach; or
 - process variables for an *EO* approach.
- $\hat{u}$ is a set of variables representing design variables.
- f is an objective function (typically of economic form).
- h_p is set of equations representing
 - tear equations for a *MO* approaches; or
 - process equations for an *EO* approaches.
- h_d is a set of design equations.
- g_p is a set of physical limitations.
- g_d is a set of design limitations.

There are three possible optimisation strategies to deal with this formulation, depending on the architecture of simulator within which they are implemented [Biegler, 1983],

- With decomposition.
 1. Two-level approach.
 2. Feasible path approach.
- Without decomposition.
 3. Infeasible path approach.

The first two methods attempt to reduce the size of the original optimisation problem by the Lagrangian decomposition techniques (two-level approach) or by the elimination of the equality constraints, eg. tear equations, model equations etc. (feasible path approach), while the third method solves the optimisation problem *P D.I* directly by a suitable optimisation method. In fact, implementations of these approaches can be achieved by the choice of the corresponding optimisation algorithms, e.g. the choice of a feasible direction method is equivalent to a feasible

path approach. On the other hand, the existing facilities of the simulators can also be explored to implement these approaches.

In this thesis, we are only concerned with feasible and infeasible path approaches since the two-level approaches have been proved inefficient for process optimisation applications [Biegler and Grossmann, 1985].

4.2.1 Feasible Path Approaches

The feasible path approach, as its name implies, generates trial points which satisfy both equality constraints (design equations, process equations and/or tear equations) and inequality constraints (physical limitations). In this approach, the process variables are partitioned into dependent variables and independent variables (design variables), and for each choice of the design variables, the simulator is used to converge the tear equations (for modular oriented simulators), or the process equations (for equation oriented simulators), and/or the design specifications, while also satisfying the design or physical limitations. Therefore, the method includes two levels, the first level performs the simulation to converge all the equality constraints (h_p and h_d) and to satisfy the inequality constraints, (g_p and g_d) and the second level performs optimisation. The resulting optimisation problem is thus an unconstrained nonlinear optimisation problem or a constrained optimisation problem with simple bounds for the associated design variables. This approach is also termed a black box method and is illustrated in Figure 4.2.

This approach played an important role in the early development of flowsheet optimisation [Friedman and Pinder, 1972], [Gaines and Gaddy, 1976], [Ballman and Gaddy, 1977], [Doering and Gaddy, 1980], [Parker and Hughes, 1981].

The advantage of this approach is that the resulting optimisation problems are in general small in terms of the number of variables. Since each search point is feasible, if the process is terminated at the optimisation level before reaching a solution, the terminating point is still feasible and may be acceptable as a practical, although sub-optimal, solution of the problem.

Difficulties can however arise since it is not easy to solve a system of equations with inequality constraints especially when the number of inequalities, g_p , is large. A feasible region may not exist for certain values of the design variables or a solution may lie on the boundary of the physical limitations (g_p). In the second case, the system becomes overdetermined since the number of equations

exceeds the number of variables. Current solution methods have difficulties in locating such points. The solution method of a system of equations with inequality constraints is an active research area [Shacham, 1986], [Pantelides, 1988 (b)] In addition, it may be difficult to choose an independent set of design variables if any design specifications are present. These phenomena are not uncommon especially for the equation oriented approach, where there is a large number of inequality constraints. To alleviate these difficulties, the inequality constraints may be classified into ' hard ' constraints and ' soft ' constraints. The hard constraints, e.g. non-negative flows, define the region where the system is mathematically well defined and must be satisfied at all time. On the other hand, the soft constraints are allowed to be violated. Thus the feasible path variant approaches include the equations (tear equations or process equations) and the associated physical limitations (hard constraints) in the simulation level, and hence the optimisation level contains design equations, associated physical limitations and/or design limitations, as illustrated in Figure 4.3.

Two different implementations of this approach has been proposed [Biegler, 1982], namely,

1. reduced feasible path variant method, and
2. converged feasible path variant method.

In the reduced feasible path variant method, only the design variables are included in the optimisation level and hence the size of the optimisation problem depends on the number of design variables and is in general small.

In the converged feasible path variant method, the optimisation level includes all the design variables, tear variables (for modular oriented approaches) or process variables (for equation oriented approaches), despite the fact that process equations or tear equations are already converged within the simulation level at every step of the optimisation. The size of the resulting optimisation problem is still large. The advantages of this approach are that the information about the gradient of design variables with respect to the process variables is not formed explicitly, the optimisation routine may provide a better initial point for the simulation, and the method can easily be switched to the infeasible path approach by the adjustment of the tolerance at the simulation level. For both the equation oriented simulators and modular oriented simulators with all the streams torn, this approach requires both a large scale equation solver and a large scale optimisation routine.

Figure 4.2 Feasible Path Approach

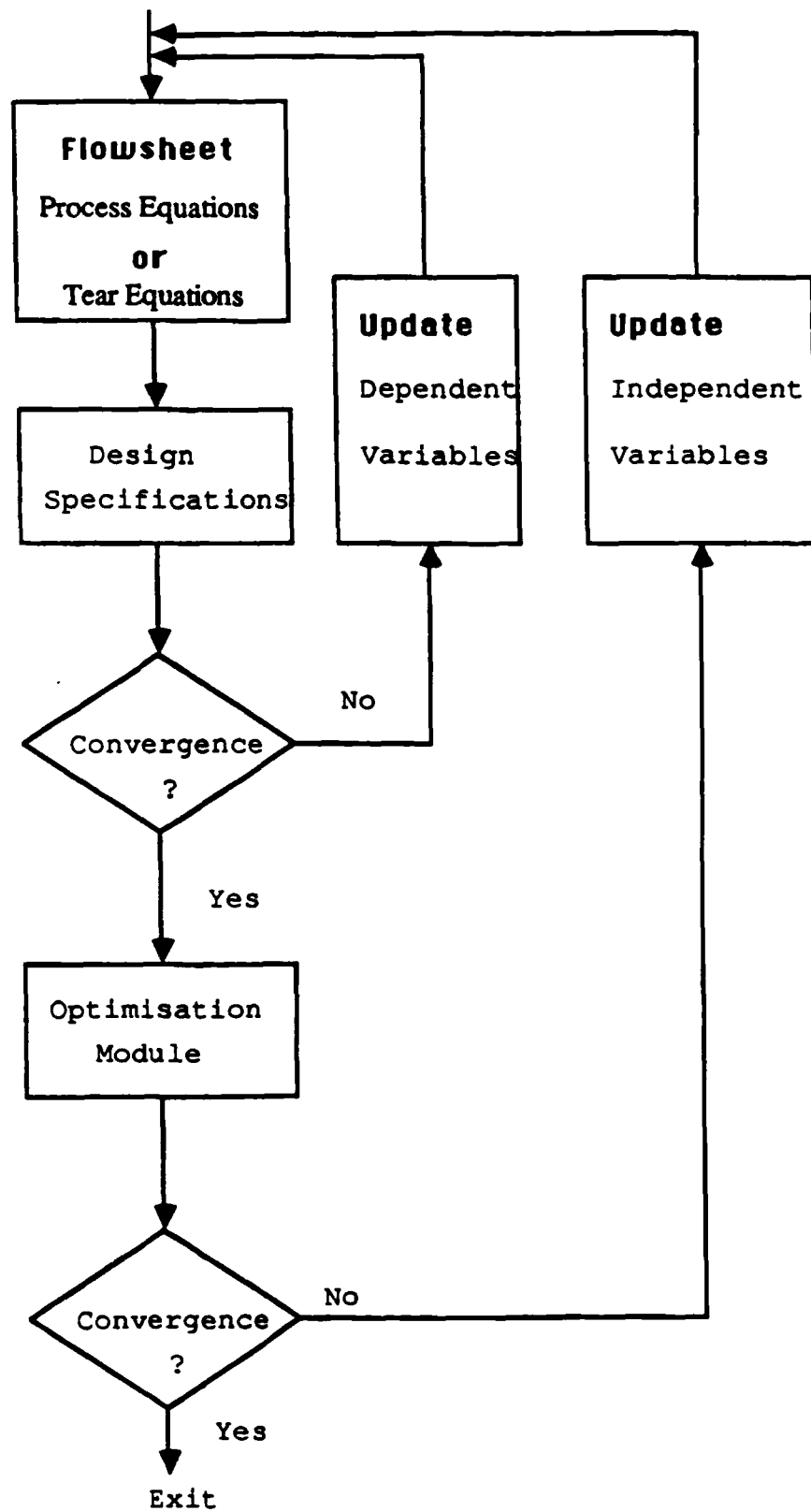
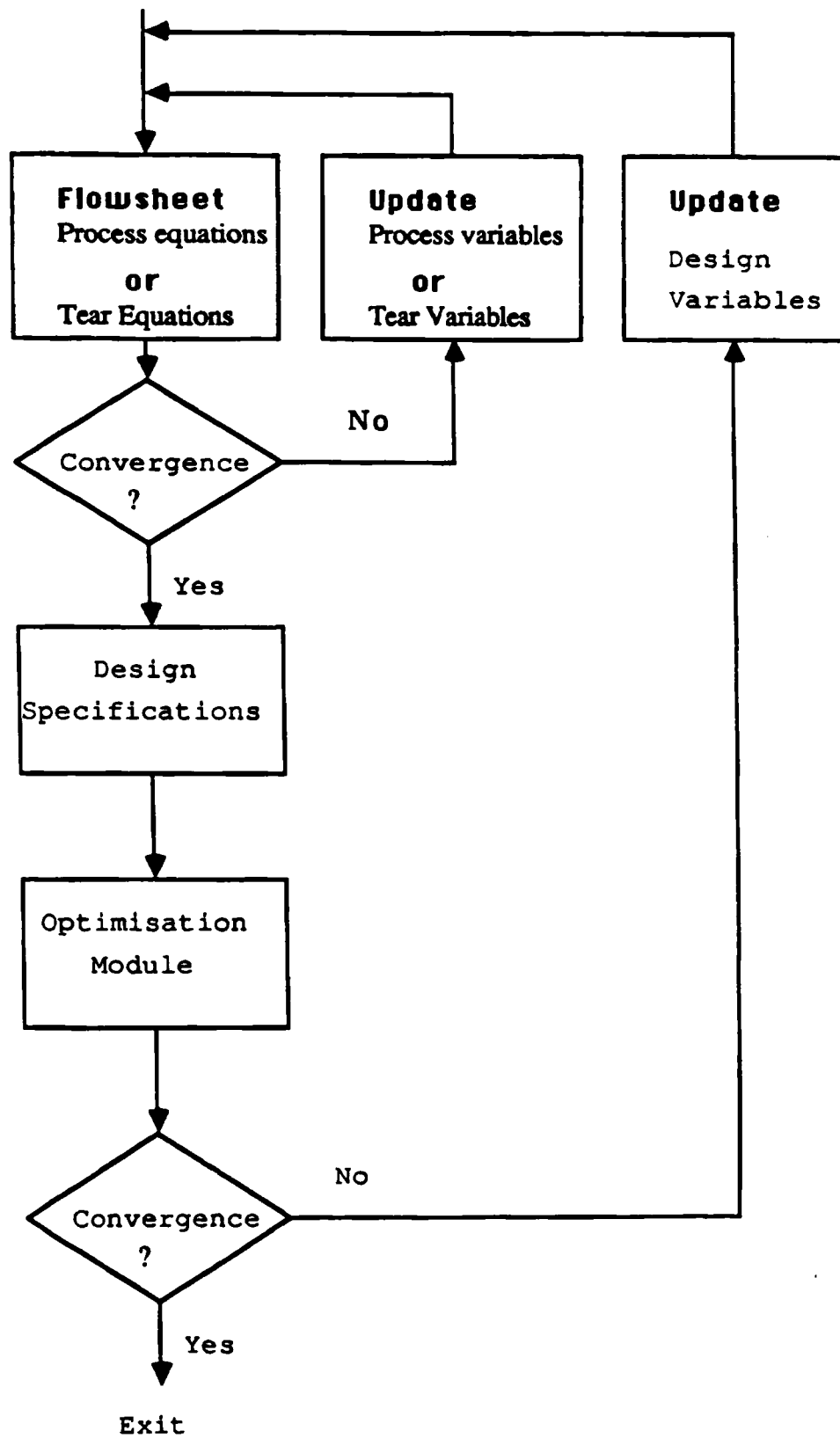


Figure 4.3 Feasible Path Variant Approach



Feasible path (variant) approaches require a complete simulation to converge the tear equations (for *MO* approaches) or process equations (for *EO* approaches) for each trial value of the design variables and this may be expensive. The simulation level may fail to provide a feasible solution especially when the system is highly constrained since a feasible region may not even exist for certain value of design variables.

Also, when the trial points are far away from an optimal solution, inefficiency may result from following the nonlinear constraints. Furthermore, flowsheet simulation may create noise in the function evaluations.

4.2.2 Infeasible Path Approaches

In this approach, the optimisation problem, *P D.I*, is solved as illustrated in Figure 4.4. The size of the resulting optimisation problem for the modular oriented approach depends on the number of both tear and design variables, while for the equation oriented approach the result is a large sparse optimisation problem. This method avoids repeated convergence simulations for each optimisation iteration and hence eliminates the possibility of premature termination due to a convergence failure of the simulation.

The effectiveness of this approach on a modular oriented simulator was shown by Biegler [1982], Chen and Stadtherr [1985] and Kaijaluto [1983]. Using this approach, Trevino-Lozano *et al.* [1985] solved flowsheet optimisation problems on a two-level approach simulator, and Hutchison *et al.* [1986] optimised several chemical processes on an equation oriented simulator.

Kisala [1985], [1987] and Biegler [1987], however, showed that this infeasible path approach may have difficulties in converging tear equations for a modular oriented approach. Therefore, they developed a method termed infeasible path hybrid method by converging the tear equations (or process equations for *EO* approaches) to some degree during the course of the optimisation, as shown in Figure 4.5. At first glance, this approach may be implemented simply by adjusting the tolerance in the simulator level in a converged feasible variant approach. The tolerance in the simulation level can be controlled by specifying the number of iterations [Kisala, 1987] or the magnitude of the residuals of the tear (or process) equations. However, the tear equations cannot be converged in an arbitrary way without reference to the optimisation method since different information are

required from the simulator for different tasks (e.g. line searches, or Hessian update). A more sophisticated control of the underlying simulator is needed.

In this study this strategy is incorporated directly into the optimisation algorithm by introducing feasibility steps, as discussed in the previous chapters. The advantage of this implementation is that the inequality constraints can be treated efficiently by the optimisation algorithm. In addition, this approach can be easily used to converge the design equations (h_d). This may be helpful when the design equations are highly nonlinear.

The infeasible path method typically requires fewer flowsheet calculations than the feasible path method at the cost of increasing the size of the optimisation problems in terms of variables. However, in practice, a chemical process may not be physically well defined over a wide range of constraints. Therefore, some precautions must be taken to identify and to satisfy the ' hard ' constraints when an infeasible path method is used.

This approach is less familiar to design engineers since each point may not be feasible and acceptable, and a large number of variables may require to be initialised. In addition, it is difficult to identify the effect of the design variables on the objective function in the course of optimisation and thus providing diagnostic information becomes a more serious issue for the infeasible path approach.

The infeasible path hybrid method provides more flexibility in the definition of which constraints must be satisfied and to what degrees. In practice, this may avoid the inefficiency of the feasible path methods and reduce the risk of the infeasible path method reaching physically unattainable regions of process models. This improvement in the robustness of the process optimisation, however, is achieved at the expense of increasing the number of function evaluations.

Figure 4.4 Infeasible Path Approach

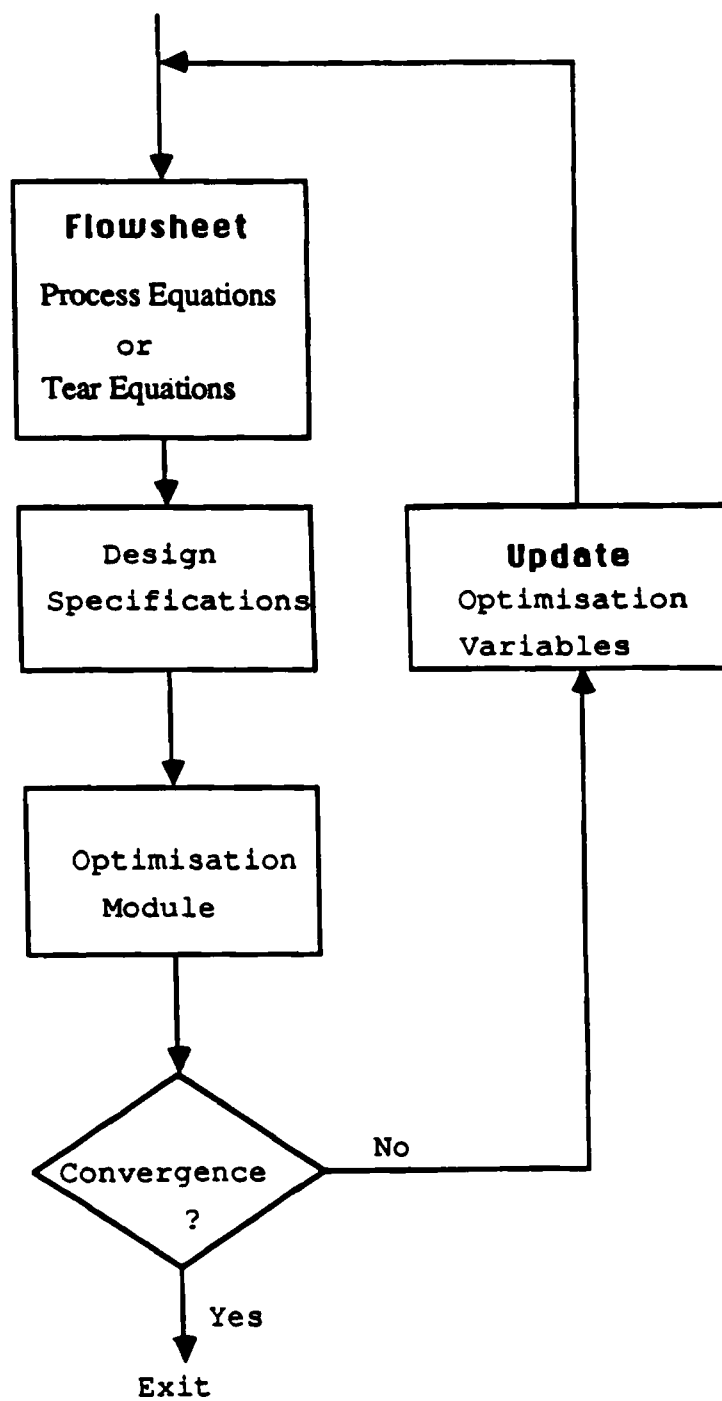
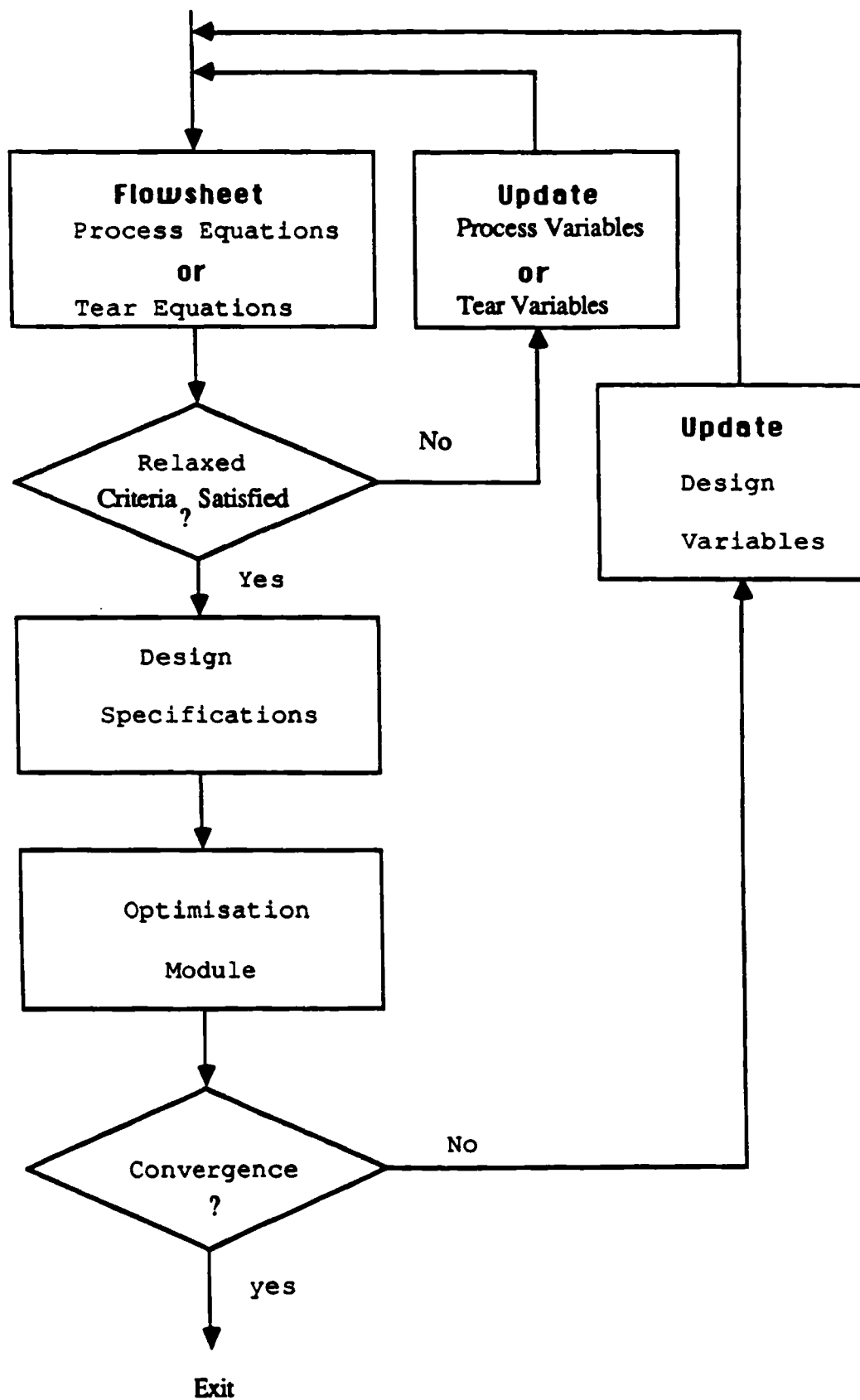


Figure 4.5 Infeasible Path Hybrid Approach



4.3 Implementation of Flowsheet Optimisation

As observed in the previous chapters, gradients of the objective function and constraints with respect to the optimisation variables are required and can be generated in various ways. To increase the robustness and efficiency of the flowsheet optimisation, a reasonable initial point is necessary. Moreover, for modular oriented approaches with an infeasible path approach, the size of the flowsheet optimisation is dependent on the tear set and thus its selection becomes an important issue. These issues are discussed below.

4.3.1 Gradient Evaluation Strategies

In general, the gradients of a function can be generated in analytical form, by approximation methods, or by update formulae from previous information. Analytical derivatives can be obtained manually or by symbolic differentiation [e.g. Pantelides, 1988 (b)]. The approximated gradients can be generated from two-point finite differences, three-point finite differences (central difference), or from an approximate model and so on. Many update formulae, for example the Broyden update formula, have been proposed to update the gradients. The choice of a suitable method may be done based on accuracy, ease of implementation, and the cost of generating the gradients. For optimisation, accuracy is the most important factor since the gradient information is used not only to generate search directions but also to judge the solutions and to obtain the curvature information, as mentioned in the previous chapters. Inaccurate gradients may result in failure to obtain a correct search direction, to predict a correct active set and to converge to an optimal point. Himmelblau and Lindsay [1980] reported that the two-point finite difference formula is the most efficient method among the finite difference formulae in terms of CPU time and function evaluations for unconstrained optimisation when the function can be evaluated accurately and a proper perturbation size is selected. The three-point finite difference (central difference) formula is more accurate than the two-point finite difference formula [Vandergraft, 1983]. However, it is too expensive to be employed in flowsheet optimisation. Update formulae, on the other hand, do not in general generate accurate gradients. Therefore the use of central difference and update schemes to obtain the approximate derivatives will not be considered further.

In the equation oriented approach, it is straightforward to implement the above gradient evaluation strategies because the process equations are individually available. Analytical gradients obtained by symbolic differentiation are usually the

best in terms of both accuracy and computing effort [Pantelides, 1988 (b)]. In addition, the algorithm of Curtis, Powell and Reid [1974], which perturbs more than one variables simultaneously, can be used to evaluate numerical gradients efficiently when a finite difference method is used. Therefore this section is concentrated on generating the gradients in the modular oriented approach.

In general, there are two techniques for gradient evaluation in the modular oriented approaches :

1. flowsheet (direct loop) perturbation approach,
2. modular gradient (procedure derivatives) approach.

In the direct loop perturbation approach, design and/or tear variables are perturbed by a small amount in turn while keeping the rest of variables constant. It is noted that only the modules (blocks) downstream in the calculation sequence from a perturbed variable have to be evaluated for the converged feasible path variant approach and the infeasible path method. A flowsheet simulation is required when each design variable is perturbed for the reduced feasible path variant methods. This method is easy to implement especially when a modular oriented simulator has a convergence block to perform flowsheet simulation by converging the tear equations and the design equations simultaneously. It is merely required to substitute the convergence block by an optimisation block in which the objective function and design equations are introduced.

The cost of this approach for both the infeasible path approach and converged feasible path method in terms of a cost factor (e.g. CPU time) is given as

$$TCF = \sum_{i=1}^{NCS} \sum_{j=1}^{TNV} \sum_{k=1}^{NB_{i,j}} CF_{ijk} \quad (4.2)$$

where

TCF	total cost factor.
NCS	total number of calculation sequence termini.
TNV	total number of variables. $\sum_{l=1}^{NTS} NTV_l + NDV$
NTS	total number of tear streams.
NTV_l	the number of variables in the l^{th} tear stream, e.g. the number of components plus temperature, and pressure.

NDV	total number of design variables.
NB_{ij}	the number of modules between the j^{th} design or tear variable and the i^{th} terminus in the calculation sequence.
CF_{ijk}	the cost factor for the evaluation of a k^{th} module associated with the j^{th} tear stream or design variable in the i^{th} terminus in the calculation sequence

It can be observed that the cost of the flowsheet perturbation approach for infeasible path approach mainly depends on the number of tear streams and tear variables, and the position of the design variables in the calculation sequence.

The cost of this approach for the feasible path approach is expressed as

$$TCF = NDV \times CFS \quad (4.3)$$

where CFS is the cost factor for a flowsheet simulation.

In practice it is relatively easy to perform a flowsheet simulation when each design variable is perturbed by a small amount. Only a few iterations are required to converge a perturbed simulation in most cases.

The second approach employs a chain ruling technique to arrange the gradients in the right calculation sequence, after the gradients of output variables with respect to input variables in each module are evaluated. These gradients can be evaluated in an analytical form. This approach is particularly efficient when the unit calculations are made by the Newton method because very few calculations are required to generate the gradients. Macchietto *et al.* [1988] reported a substantial reduction in computational time when this method is used. The gradients of output variables with respect to input variables can also be obtained from perturbing the input variables of each module. An extra effort is required to evaluate the reduced gradients of the tear variables with respect to the design variables when this method is applied to the reduced feasible path (variant) approaches. This method is also referred to as a block perturbation method and its effectiveness on flowsheet optimisation was shown by Shivaram and Biegler [1983] and Chen and Stadtherr [1985]. However, extra storage for all the modular (procedure) gradients is required for this method. A technique to implement chain ruling was proposed by Chan and Prince [1985]. The cost of implementation depends on the structure of the simulator and in particular on the availability of the information regarding the calculation sequences from the design variables to the tear variables.

The complexity of implementing this gradient evaluation method also depends on the choice of a tear set.

The cost factor of the gradient evaluation for the block perturbation can be expressed as follows,

$$TCF = \sum_{i=1}^{NCS} \sum_{j=1}^{NB_i} NBV_j TC_{i,j} + \sum_{k=1}^{TND} TCF_k \quad (4.4)$$

where

- $TC_{i,j}$ the cost factor of evaluating the j^{th} module in the i^{th} calculation sequence.
- TCF_k the cost factor of perturbing the k^{th} design variable.
- NB_i the number of modules in the i^{th} calculation sequence.
- NBV_j the number of input variables in the j^{th} module.
- NCS the number of calculation sequences.

The analytic procedure derivative with the chain ruling strategy is the most accurate and efficient strategy among these gradient evaluation strategies. However, this gradient strategy is harder to implement. The relative efficiency of the flowsheet perturbation over block perturbation with chain ruling is dependent on the structure of the flowsheet, and on the optimisation problems. The overheads for gradient evaluations for the infeasible path may be significant when the number of tear variables is far more than that of design variables.

The best choice among gradient evaluation strategies depends on the structure of the simulator, types of problems and available storage. It is recommended to estimate the cost of various strategies from equation (4.2), (4.3) and (4.4) before choosing a strategy when the simulator allows to implement any of alternatives. In this study, the flowsheet perturbation approach is used due to the structure of the available simulator.

It is worthy of noting that the accuracy of the gradients strongly depends on the perturbation size when perturbation techniques are used. The most important errors involved in the perturbation techniques are truncation error, rounding (condition or cancellation) error and the procedure error. The rounding error is introduced by the limited machine precision, the truncation error is caused by the approximation to functions, and procedure error comes from convergence of iterative procedures to a finite accuracy. The rounding error and the procedure

error will increase with a reduction of perturbation size while the truncation error increases with an increase of the perturbation size.

These errors make it difficult to choose a proper perturbation size in the modular oriented approach. There are at least two nested levels of calculations (physical property calculations and unit calculations) for the infeasible path approach and at least three nested level calculations (physical property calculations, unit calculations, and flowsheet simulation) for the feasible path approaches. Also at least two nested calculations (physical property calculations and unit calculations) are involved in the block perturbation method for both the feasible path variant and the infeasible path approach.

In some cases, it may not be possible to reduce the procedure error to the desired level by tightening its convergence tolerance because of the limited machine precision. It is noted that the procedure errors are bounded by the convergence tolerance within the unit calculation.

Gill *et al.* [1983 (b)] proposed an algorithm to estimate the perturbation size based on gradients evaluated from forward finite difference, backward finite difference, and second order difference. The perturbation size is selected as the smallest value that results in acceptable values of relative errors for all the above different approximation schemes. The relative error is derived from a ratio between the rounding error and the corresponding truncation error in the difference scheme. These techniques are expensive since extra function evaluations are required.

Kaijaluoto and Morton [1987] proposed a procedure for gradient evaluations. The algorithm starts with the estimation of the noise level of the system and the evaluation of gradients from a central difference scheme. Then a second correction term that approximates the Hessian of the function is calculated from a second order difference. The perturbation size for the evaluation of the second order term is based on the balance of the function noise and truncation error. Subsequent gradients are evaluated from a modified finite difference scheme in which the second order term is introduced. The second order correction term is re-evaluated periodically. The authors showed the effectiveness of approach by applying it to the optimisation of Williams-Otto process. The effectiveness of this algorithm depends on the accuracy of the second order correction term. The period used to re-evaluate this term depends on the nonlinearity of the optimisation problems. In fact the method becomes a costly central difference scheme when the second order term is re-evaluated at every optimisation iteration.

In addition, Biegler [1985] showed that the chain-ruling strategy may lead to gradient errors larger than the flowsheet perturbations when the procedure error is significant compared to the perturbation size. This is due to the accumulation of the procedure errors.

This study emphasises the consideration of the procedure error in order to choose the perturbation size. A perturbation size based on the balance between truncation error, procedure error, and rounding error is derived in appendix D.1. It is also required that the (relative) tolerance in the inner loop should be tighter than the (relative) tolerance in the outer loop.

4.3.2 Initialisation of Process Variables

In practice, a sensible initial point is often required to solve optimisation problems since the initial point should not make the system unattainable either physically or mathematically.

In the feasible path approach, the design variables at the optimisation level and tear variables or process variables at simulation level need to be initialised. The variables at the simulation level need to be initialised at every optimisation iteration. The design variables, the tear variables or process variables at the very first simulation may be initialised based on corresponding physical properties and/or previous knowledge about the problem. The process variables in subsequent simulations may be initialised in the following ways :

1. direct substitution of the values corresponding to the solution of the previous iteration.
2. using the relationship between the design variables and the tear (or process) variables.
3. by the inclusion of process variables in the optimisation routine [Biegler, 1982].

The first method simply stores the values of variables at each iteration, and is easy to implement. The tear or process variables in the second method can be initialised as [Biegler, 1982]

$$\hat{v}_{k+1} = \hat{v}_k - \nabla_{\hat{u}} h_p(\hat{v}_k, \hat{u}_k)^{-T} \nabla_{\hat{v}} h_p(\hat{v}_k, \hat{u}_k)^T \Delta \hat{u}_k \quad (4.5)$$

Alternatively, the following relation can be used,

$$\hat{v}_{k+1} = \hat{v}_k - \nabla_{\hat{u}} h_p(\hat{v}_k, \hat{u}_k)^{-T} (h_p(\hat{v}_k, \hat{u}_k) + \nabla_{\hat{v}} h_p(\hat{v}_k, \hat{u}_k)^T \Delta \hat{u}_k) \quad (4.6)$$

where

- $\hat{u}_k$ is a set of design variables.
- $\hat{v}_k$ is a set of variables representing
 - tear variables for modular oriented approach, or
 - process variables for equation oriented approach.
- h_p is a set of equations representing
 - tear equations for modular oriented approach, or
 - process equations for equation oriented approach.
- $\Delta \hat{u}$ is a perturbed value for variables $\hat{u}$.
- $\nabla_{\hat{v}} h_p$ is the gradients of equations h_p with respect to variables $\hat{v}$.

This method can be efficiently implemented either with modular oriented approaches when procedure derivatives are used for gradient evaluation or with equation oriented approaches when the analytical derivatives are available.

The third method is a direct by-product of the converged feasible path approaches. The choice among these methods thus depends on how gradients are calculated.

In the infeasible path method, all the variables need to be initialised at the optimisation level. An initial value of tear or process variables may be obtained from a complete or partial simulation controlled by the number of iterations or the simulation tolerance with guessed values of design variables. Jirapongphan [1980] and Biegler and Hughes [1982] showed that the use of a converged or nearly converged simulation as a starting point may result in small optimisation steps and lead to premature termination in the modular oriented approach. However, with the same simulator architecture, Chen and Stadtherr [1985] showed that good results may be obtained by performing several direct substitution iterations. Trevino-Lozano observed that the decomposition method proposed by Locke *et al.* [1983] may fail when the initial values of the variables make the quadratic programming subproblem infeasible [Kisala *et al.*, 1987] and recommended taking several Newton-Raphson iterations of the simulation when this is likely to occur.

In general, the use of either complete or partial simulation for the initialisation of the process optimisation may provide a sensible starting point and hence improve the robustness of any optimisation method by decreasing the chance of involving numerical difficulties and ill-conditioning. As pointed out in Chapter 2, however, optimisation routines balance the reduction of the objective function and the satisfaction of constraints and when a penalty function is used with the SQP methods to force convergence, a small initial violation of the constraints at a converged initial point may force the methods to take small steps. This situation is likely to occur when the objective function is insensitive to the optimisation variables, constraints are highly nonlinear and large values of the penalty parameters are used. Small optimisation steps may also cause premature termination or make the algorithm identify a false local minimum in the modular oriented simulators when functions are noisy due to the nested calculations. In fact, these difficulties can also occur when the iterates are near a feasible point. These problems may be avoided by scaling the objective function, reducing the function noise and choosing suitable initial penalty parameters.

As shown in Chapter 2, an initial Hessian based on the magnitude of the variables can dramatically improve the performance of the optimisation method. Therefore, in this work, a few direct substitutions to obtain a reasonable starting point are employed when using a modular oriented simulator. The initial penalty parameter is set to a small number, say 10^{-6} , to reduce inefficiency and step bounds for each variable are introduced directly into the QP subproblem, as described in Chapter 2, to avoid excessively large steps. A partial flowsheet simulation is also recommended for optimisation in an equation oriented simulator to provide an initial order of magnitude of all variables and to avoid physical and arithmetical difficulties.

4.3.3 Selection of Tear Streams

The size of the flowsheet optimisation problem and the efficiency of optimisation methods depend on the number of tear variables for the modular oriented approach. Selection of a tear set is equivalent to solving the following linear integer programming problem :

$$\begin{aligned}
 PD.II \quad & \min_z \sum_{i=1}^n \gamma_i z_i \\
 & s.t. \sum_{i=1}^n \psi_{ij} z_i \leq 1.
 \end{aligned} \tag{4.7}$$

where

γ_i a weight for the i^{th} stream.

z_i a 0-1 integer variable corresponds to the i^{th} stream.

ψ_{ij} denotes the presence or absence of the stream i in loop j , i.e.

$$\psi_{ij} = \begin{cases} 0 & \text{if stream } i \text{ is not in loop } j \\ 1 & \text{otherwise.} \end{cases}$$

The tear variables should have some effect on the constraints and/or the objective function. It is also desirable to minimise the number of optimisation variables to reduce the cost of the gradient evaluations and the overhead of the optimisation methods. Therefore, the following heuristic rules may be used,

1. Choose a tear set that minimizes the number of tear variables.
2. Choose a tear set that will result in a good convergence behaviour. This issue is dependent on the types of the optimisation problems. In general, a tear set whose tear variables have a stronger effect on the constraints or the objective function is preferred.

These heuristic rules may be incorporated by choosing suitable weights, γ_i , in the integer program *P D.II*. Thus the weight γ_i can be chosen as the number of variables in the i^{th} stream multiplied by a weight corresponding to the effect on the constraints and/or the objective function.

In practice, it is not necessary to tear all the variables in a tear stream. Those variables, eg. inert components, which have little effect on the system (objective function, design equations and the other tear variables) or have known fixed values, e.g. pressure, can be removed from the tear set. These variables can be dealt with separately outside the optimisation routines. Biegler [1985] also pointed out that pressure is often a weak function of flow rate or enthalpy and thus this variable can be left out from the tear set when the outlet pressure of a unit is fixed.

In this study, it is assumed that the selection of the tear streams is performed by the users, based on these or similar heuristic rules.

4.4 Numerical Experiences

The characteristics of both the *SQP* algorithm for the small scale optimisation and *SRQP* algorithm for large scale optimisation with few design variables were analysed in the previous chapters, and their algorithmic performance was also evaluated on both mathematical and simple chemical process problems.

This section investigates the practical performance of these methods on more complex and realistic processes, many of which are of industrial significance. The first part deals with process optimisation on a commercial *MO* simulator, and the second part with a commercial *EO* simulator.

The objectives of this study are to examine the algorithmic performance of the proposed optimisation methods, and the relative efficiency of various optimisation strategies.

The measures of merit considered in this numerical study in order of importance are based on the the number of gradient evaluations and function evaluations, and central processing unit (*CPU*) time. The number of function evaluations excludes those evaluations required for evaluating the gradients in both type simulators when the analytical derivatives are not available. The *CPU* time, however, includes all function evaluations for the feasible path (variant) approaches and infeasible path hybrid approaches. In the modular oriented approach it is helpful to include a measure of merit termed the simulation time equivalent (*STE*). The simulation time equivalent is defined as the ratio of *CPU* time spent on the flowsheet optimisation over the *CPU* time spent on the initial flowsheet simulation at the starting point. The validity of using the number of function and gradient evaluations is based on the assumption that the flowsheet evaluations are far more expensive than the optimisation algorithm itself in terms of *CPU* time. This assumption is in general true especially for simulations involving physical properties calculations and for the modular oriented simulators [Westerberg, 1981]. Richard and Pike [1987] argued that it is necessary to use the *CPU* time to compare the relative efficiency of feasible path methods over infeasible path methods since the feasible path methods may take fewer iterations but each iteration is more expensive. The advantages of using *STE* as a measure of merit are that it is independent of the type of the machine, and it, with the aid of the number of gradient evaluations, can also be used to verify both the efficiency of the gradient evaluation strategy and the overall efficiency of the flowsheet optimisation. However,

it depends on the choice of the starting point and the numerical method used for flowsheet simulation.

Test problems used in this study fall into two categories, well known test problems from literature and simplified industrial processes. The well known test problems are well documented in the literature so that some comparisons can be made as to the efficiency of this implementation. The use of simplified industrial processes is to establish the capabilities of the current algorithms with more realistic problems.

4.4.1 Process Optimisation with a MO Approach

The optimisation algorithm described in Chapter 2 was interfaced with the *GENESIS* (*GEN*eralised *SI*mulation *S*ystem) simulator [Banks, 1984] developed within the British Petroleum (*BP*) company. *GENESIS*, a traditional sequential simulator, handles systems encountered in the petroleum and petrochemical industries, and is representative of current industrial simulators. It has comprehensive libraries of thermodynamic and physical property data and a library of crude oil properties. The simulator deals with simulation and design problems by using a quasi-Newton method to converge the design and tear equations simultaneously [Perkins, 1982].

Seven process optimisation problems, the characteristics of which are summarised in Table 4.1 are employed for this study.

Test problem 1 is a simple flash system with a recycle, based on a flowsheet proposed by Biegler and Hughes [1981] as shown in Figure 4.6. A light hydrocarbon feed mixed with a recycle is fed to an adiabatic flash. The overhead is removed as product and a fraction of the bottoms is recycled through an adiabatic flash at its dew point. The objective of this problem is to maximise the molar flow of propane in the overhead flash vapour by manipulating flash pressure and recycle fraction within certain ranges. The specification of this problem is listed in Table 4.2. Stream 4 is selected as a tear stream.

Test problem 2, studied by Lang and Biegler [1987], seeks to maximise the degree of separation of a mixture of hydrocarbons among the overheads, bottoms and two sidestreams in a distillation column with 25 theoretical trays and two sidestreams by controlling the reflux ratio and the fractions of feed to the two sidestreams and the distillate. The flowsheet of this process is illustrated in Figure 4.7 and its specification is listed in Table 4.3.

Table 4.1 Characteristics of Test Problems for a *MO* Approach

Problem No.	Processes	N(NTV)	MEQ	M	NB	OPT	ϵ
P 1	Simple flash	9(7)	7	7	4	9.5	10^{-4}
P 2	Simple distillation	4(0)	0	1	8	-8.804	10^{-4}
P 3	Refrigeration	7(4)	4	7	7	2.748×10^6 *	10^{-2}
P 4	Cryogenic exchanger	9(3)	3	15	15	1.644×10^4	10^{-2}
P 5	Crude oil distillation	6(0)	0	17	12	-4.467×10^3	10^{-3}
P 6	Catalytic reforming	15(10)	11	15	20	1.789×10^5	10^{-3}
P 7	Cavett's process	38(34)	35	36	42	28.68	10^{-3}

Key

- N Number of all the variables.
 NTV Number of tear variables.
 MEQ Number of equality constraints.
 M Number of constraints except simple bounds.
 NB Number of simple bounds.
 OPT Optimal value of the objective function.
 ϵ Tolerance of the optimisation calculation.
 * The value cited in the literature is 1.57×10^6 .

Test problem 3, suggested by Himmelblau [1979] is a light hydrocarbon refrigeration process with three adiabatic flash drums and overhead recycles through isentropic compressors, as shown in Figure 4.8. The objective is to minimise the work in the compressors. The use of a different compressor model results in a disagreement between the solution obtained from this study and that from the literature [Himmelblau, 1981]. Table 4.4 lists the specification of this process. The tear stream is stream 2.

Test problem 4 contains a complex cryogenic exchanger process developed by BP, illustrated in Figure 4.9. This process cools the overhead product from a distillation column by using three shell and tube heat exchangers with a mixed composition refrigerant (MCR). The objective is to minimise the compressor work by adjusting pressure in flash 1, MCR component flow rate and difference of pinch temperatures between stream 13 and stream 16. Streams 14, 16 and 17 are chosen as tear streams.

Test problem 5 is a crude oil distillation column problem developed by BP. The process is illustrated in Figure 4.10. The objective is to achieve four main product qualities as closely as possible by varying the crude unit side offtake rates.

Test Problem 6, illustrated in Figure 4.11, is a catalytic reforming process developed by BP. This process contains three reactors to convert naphtha to stabilised reformat. The objective is to maximise the profit and to meet a fixed research octane number by controlling the inlet temperature in three reactors, the ratio of hydrogen and feed hydrocarbon and total feed flow rate. The tear stream is stream 6.

Test problem 7 is the well-known Cavett's problem with four adiabatic flash drums and recycles [Cavett, 1963]. The objective here is to minimise the flow rate of hydrogen sulphide in the product stream by adjusting the pressures in the four adiabatic flash drums. A flowsheet of this process is described in Figure 4.12 and its specification is listed in Table 4.5. Streams 2 and 8 are chosen as tear streams. This problem can also be used to test the relative efficiency of the feasible path method over the infeasible path method since it contains nested recycles.

The physical properties for this study are based on the *Redlich-Kwong-Soave* equation of state for the test problems one, three, and five, while the rest of the calculations are calculated from a modified *Redlich-Kwong* equation of state [Zudekevitch and Joffe, 1970], all of which are available through the *GENESIS* simulator.

Three strategies for process optimisation were implemented, namely a feasible path variant, an infeasible path and an infeasible path hybrid approach. Gradients are generated by direct flowsheet perturbations in all cases.

A starting point is generated by one direct substitution iteration except for test problem 3 and 4 where two direct substitution iterations are used.

A relative convergence criterion, equation (2.23), is employed for the optimisation. The relative tolerance for all the unit calculations is 10^{-5} and the relative tolerance for flowsheet simulation in the feasible path variant approach is 10^{-5} for test problem 1 and 10^{-3} for test problems 3, 4, 6 and 7. There are no recycles in test problem 2 and 5.

The machine used is a VAX 8530 (machine precision is 2.2×10^{-6}) and single precision is used for all the calculations.

The following default options are used for this study unless specified.

1. An infeasible path strategy is employed.
2. The variables are initialised by one direct substitution except for test problem 4 where two substitutions are used.
3. The initial Hessian is chosen as a diagonal matrix based on the order of magnitude of variables, equation (2.65).
4. The relative step bounds are 1, i.e.,
$$-\max(1, |x|) \leq d_x \leq \max(1, |x|).$$
5. The line search method and an absolute value penalty merit function, equation (2.20), with a simple watchdog technique (algorithm B.II) is used.
6. The relative convergence criterion, equation (2.24), is used.

This section starts with studying the effect of initialising the Hessian and tear variables. Different line search and trust region algorithms are then examined. The relative efficiency of various optimisation strategies including a feasible path variant, an infeasible path and an infeasible path hybrid approach are also examined.

Figure 4.6 Simple Flash Process Flowsheet

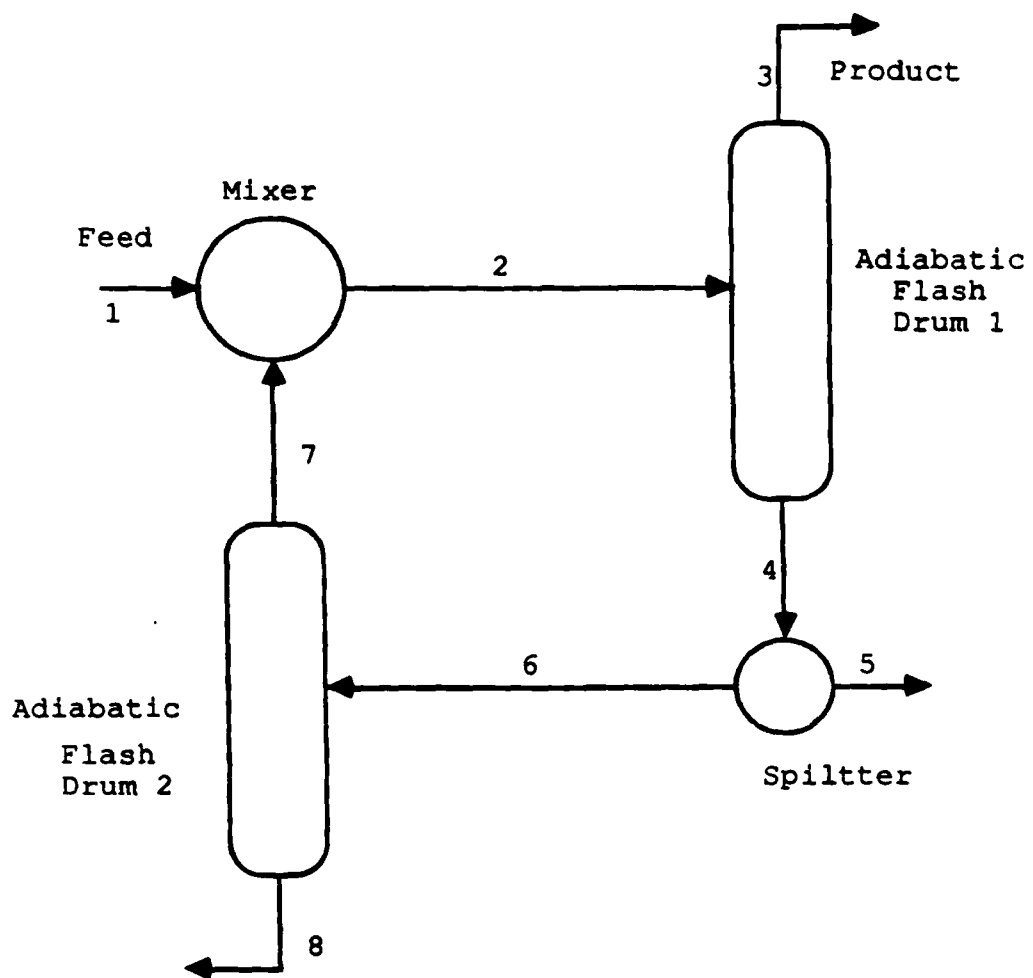


Table 4.2 Specifications for a Simple Flash Process

Feed

Temperature 100 Degrees F

Pressure 150 psig

Component flowrate (lb mol / hr)

Propane 10

N-Butane 20

1-Butene 15

Trans-2-Butene 20

Cis-2-Butene 20

n-Pentane 10

Design variables

	Initial	Optimum
Sf - Split fraction in Splitter (Stream 5 / stream 4)	0.7	0.2
Pf - Adiabatic flash pressure (Psig)	45	10

Objective function

Maximise Propane in Stream 3

Initial 4.16

Optimum 9.50

Constraints

$0.2 \leq Sf \leq 0.8$

$10 \leq Pf \leq 45$

Figure 4.7 Simple Distillation Optimisation

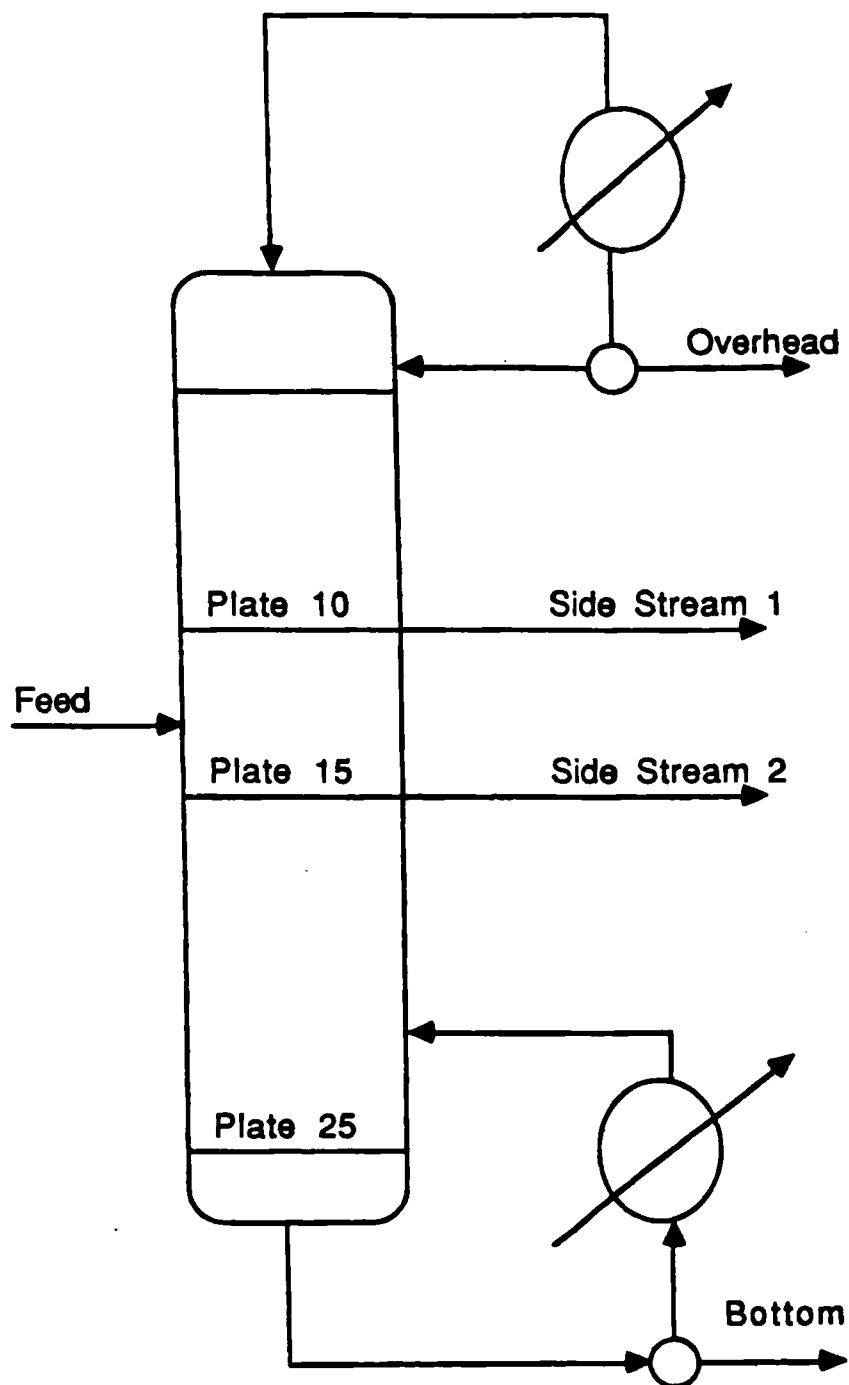


Table 4.3 Specifications for a Simple Distillation Column

Feed			
Temperature	120 Degrees F		
Pressure	26 psig		
Component flowrate	(lb mole / hr)		
n-Pentane	(C5)	3.049	
n-Hexane	(C6)	1.276	
n-Heptane	(C7)	1.597	
n-Octane	(C8)	0.438	
n-Nonane	(C9)	3.119	
Design Variables			
		Initial	Optimum
Fr1	fraction of feed to side stream 1	0.2	0.235
Fr2	fraction of feed to side stream 2	0.2	0.127
Fr3	fraction of feed to the overhead stream	0.3	0.323
Ref	Reflux ratio	1.0	20.
Objective function			
Maximise			
C5 in the overhead stream + C6 in side stream 1 + C7 and C8 in side stream 2 + C9 in the bottom stream			
Initial	6.643		
Optimum	8.804		
Constraints			
0.1 ≤ Fr1 ≤ 0.7			
0.1 ≤ Fr2 ≤ 0.7			
0.1 ≤ Fr3 ≤ 0.7			
Fr1 + Fr2 + Fr3 ≤ 0.95			
0.5 ≤ Ref ≤ 20			
P2 = 20		Pressure in the overhead stream	

Figure 4.8 Refrigeration Process Flowsheet

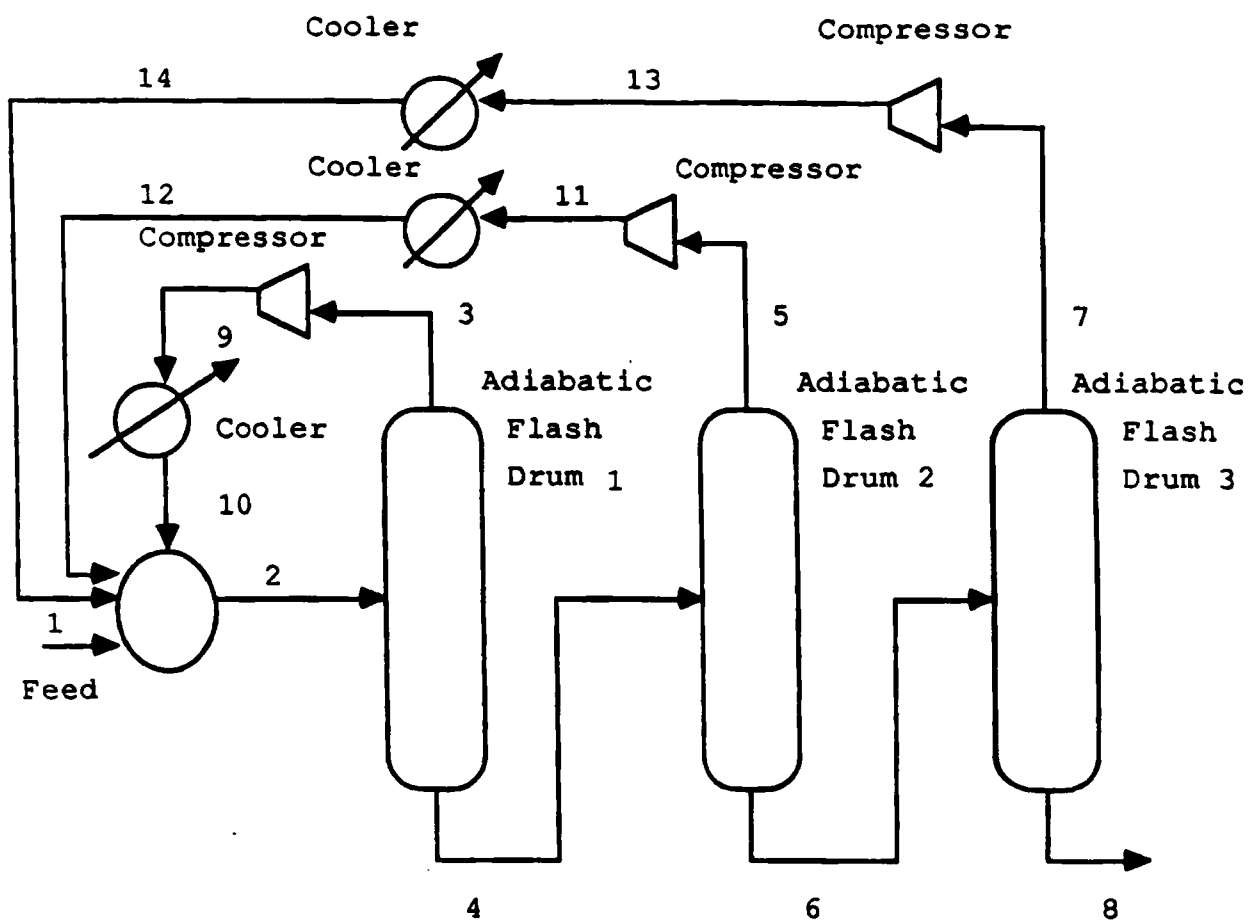


Table 4.4 Specifications for a Refrigeration Process

Feed

Temperature	200	Degrees F
Pressure	500	Psig
Component	Flow rate	(lb mole / hr)
Ethane	250	
Propane	600	
N-Butane	150	

Design Variables

		Initial	Optimum
Pf1	Pressure in Flash 1	300	378.21
Pf2	Pressure in Flash 2	175	201.16
Pf3	Pressure in Flash 3	15	52.80

Objective function

Minimise

$$H(9) - H(3) + H(11) - H(5) + H(13) - H(7)$$

Initial 5.269E6

Optimum 2.748E6

H(i) Enthalpy in Stream i (Btu / hr)

Constraints

$$P(5) - P(3) \leq 0$$

$$P(7) - P(5) \leq 0$$

$$T(7) + 20 \leq 0$$

P(i) Pressure in Stream i (psig)

T(i) Temperature in Stream i (degrees F)

Figure 4.9 Complex Cryogenic process Flowsheet

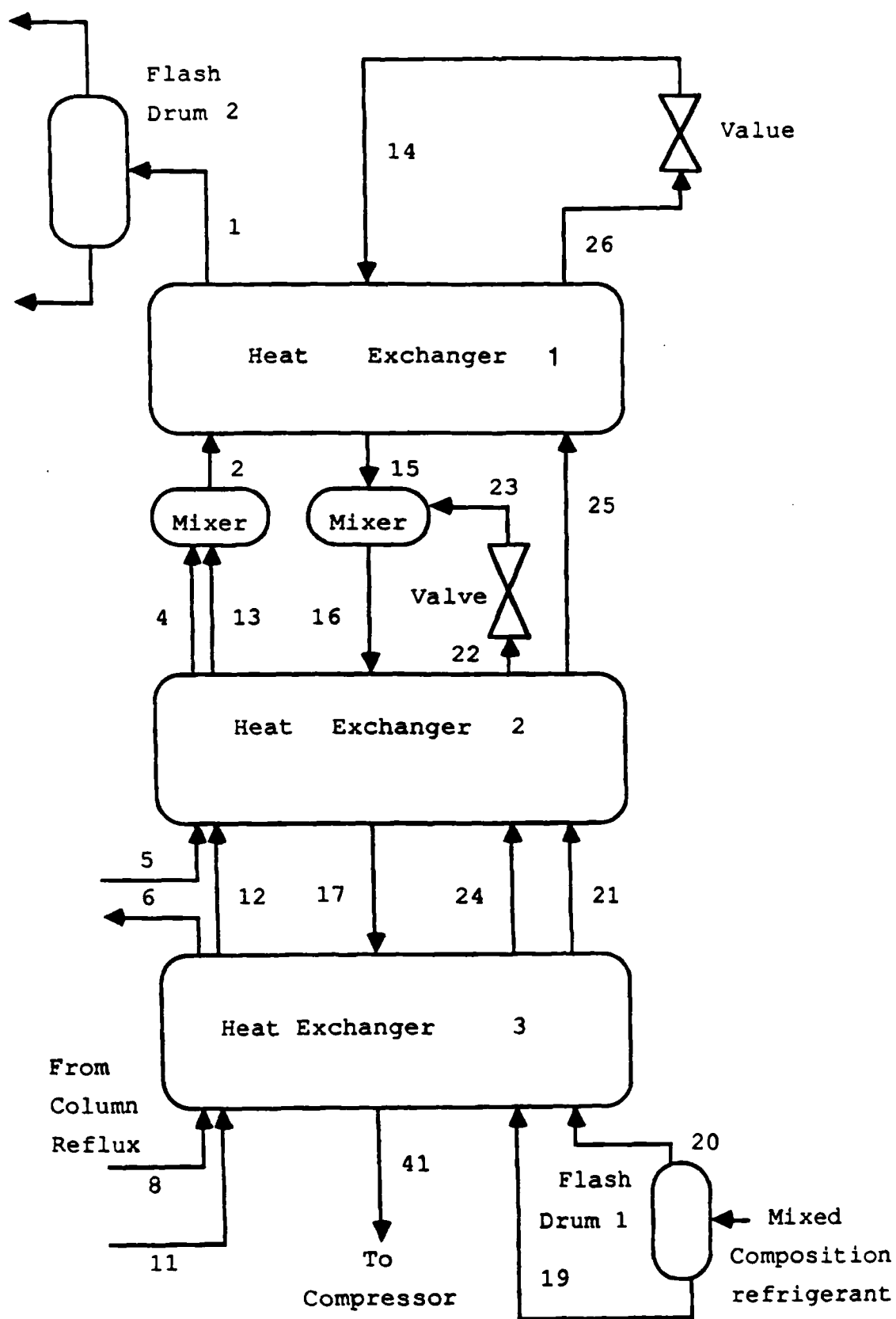


Figure 4.10 Crude oil Distillation Optimisation

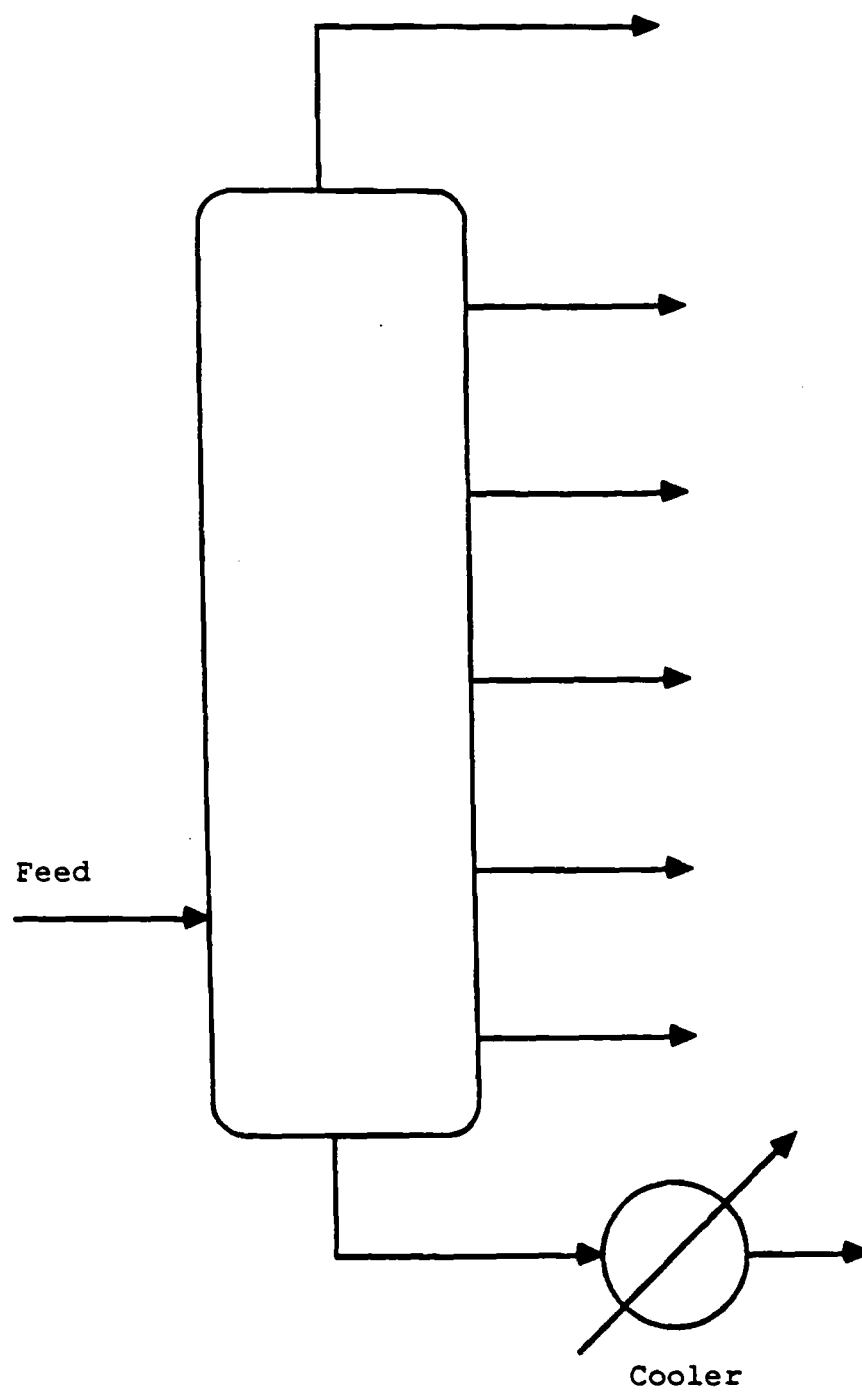


Figure 4.11 Catalytic Reformer Process Flowsheet

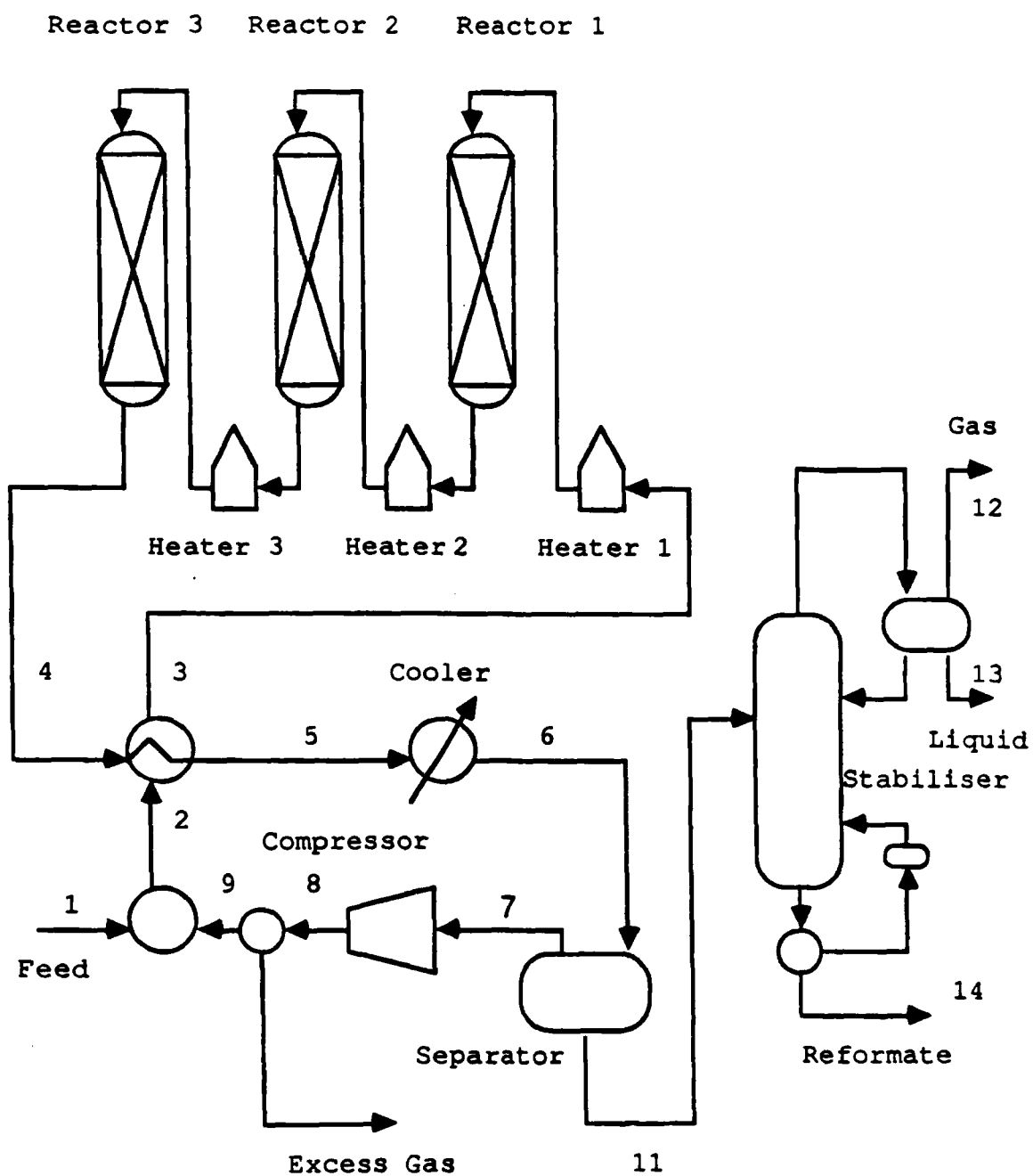


Figure 4.12 Cavett's Process Flowsheet

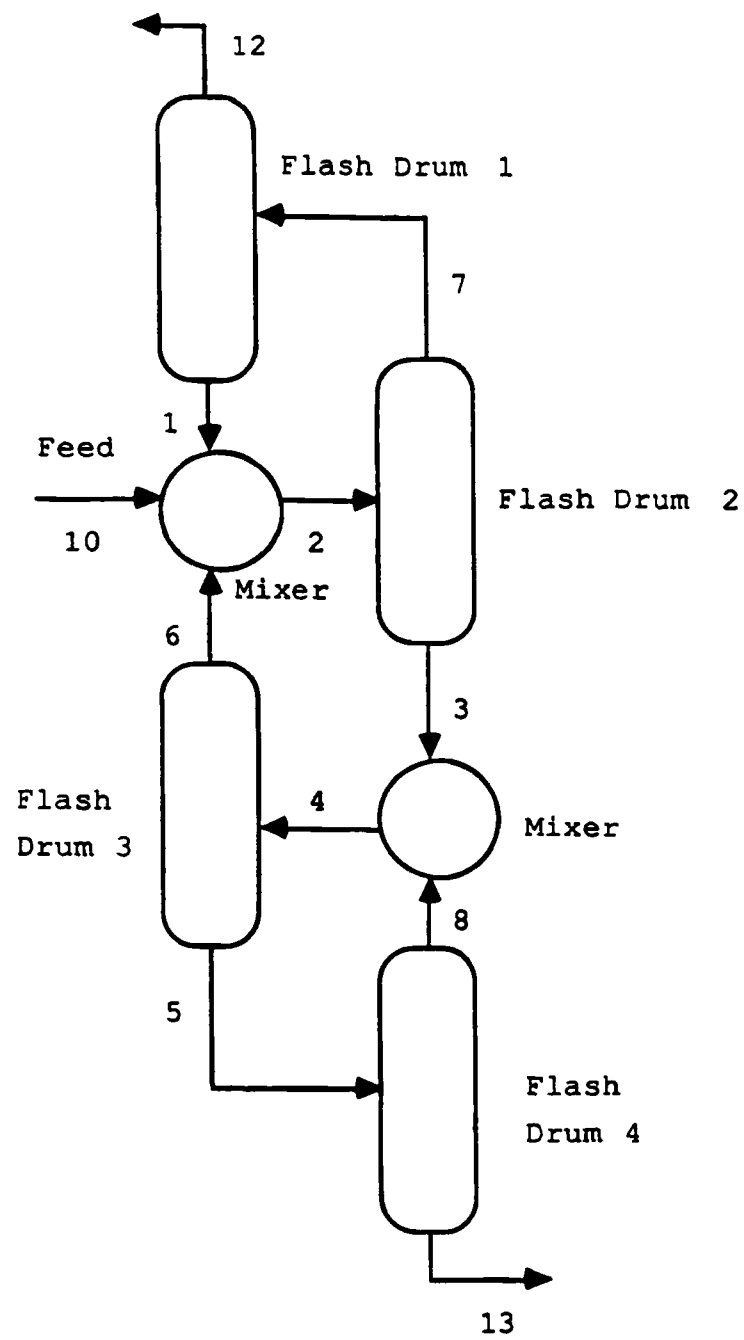


Table 4.5 Specifications for the Cavett's Process

Feed

Temperature	120 F
Pressure	370. psig
Component	(lb mole / hr)
Nitrogen	358.20
Carbon Monoxide	4965.60
Hydrogen Sulphide	339.00
Methane	2995.50
Ethane	2395.50
Propane	2291.00
i-Butane	604.00
n-Butane	1539.90
i-Pentane	790.40
n-Pentane	1129.90
n-Hexane	1764.70
n-Heptane	2606.70
n-Octane	1844.50
n-Nonane	1669.00
n-Decane	831.70
n-Undecane	1214.50

Design variables

		Initial	Optimum
Pf1	Pressure in Flash 1	800.00	832.48
Pf2	Pressure in Flash 2	270.00	370.00
Pf3	Pressure in Flash 3	49.00	46.41
Pf4	Pressure in Flash 4	13.00	5.00

Objective function

Minimise

Hydrogen Sulphide in Stream 3

Initial	57.02
Optimum	28.81

Table 4.5 Specifications for the Cavett's Process (continued)

Constraints

$$\text{IC4 (12)} = 115$$

$$\text{N2 (1)} \leq 0.01$$

$$600 \leq \text{Pf1} \leq 1000$$

$$170 \leq \text{Pf2} \leq 370$$

$$25 \leq \text{Pf3} \leq 75$$

$$5 \leq \text{Pf4} \leq 20$$

IC4(12) i-Butane in Stream 12

N2(1) Nitrogen in Stream 1

4.4.1.1 Effect of the Initial Hessian

Three strategies for the choice of the Hessian proposed in Chapter 2 are compared. An identity matrix serves as an initial Hessian in the first strategy, equation (2.64). The second strategy, equation (2.65) is based on the magnitude of variables at the initial point. The gradient information at the initial point is also included in the third strategy, equation (2.66).

The results in Table 4.6 show that scaling the Hessian is effective especially when the variables cover a wide range of magnitudes, e.g. test problem 1. The strategy based on the equation (2.64) performs satisfactorily for these test problems.

Table 4.6 Effect of choice of the Initial Hessian

Problem No.	HESS I		HESS II		HESS III	
	NG (NF)	CPU (STE)	NG (NF)	CPU (STE)	NG (NF)	CPU (STE)
P 1	11 (12)	0.9 (6.6)	7 (7)	0.6 (4.0)	5 (5)	0.5 (3.3)
P 2	15 (29)	11.5 (30.7)	15 (31)	11.7 (31.3)	15 (28)	11.4 (30.5)
P 3	11 (11)	4.1 (14.4)	9 (11)	3.4 (11.8)	6 (12)	2.2 (7.8)
P 4	11 (11)	7.5 (22.0)	10 (16)	7.7 (22.6)	12 (12)	8.1 (24.1)
P 5	6 (6)	4.4 (15.4)	6 (6)	4.4 (15.4)	6 (6)	4.3 (15.2)
P 6	11 (11)	8.7 (21.5)	9 (12)	7.2 (17.9)	11 (11)	8.6 (21.4)
P 7	10 (10)	9.2 (26.9)	10 (10)	9.1 (26.6)	10 (10)	9.1 (26.6)

Key

HESS I	The initial Hessian is an identity matrix, equation (2.64).
HESS II	The initial Hessian is a diagonal matrix based on the order of the magnitude of the variables, equation (2.65).
HESS III	The initial Hessian is a diagonal matrix based on the order of magnitude of variables and gradients, equation (2.66).
NG	Number of gradient evaluations.
NF	Number of function evaluations.
Machine	VAX 8530 .
CPU	Central processing unit time (min).
STE	Simulation time equivalent.

4.4.1.2 Initialisation of Tear Variables

Two initialisation schemes were investigated. The first scheme uses one direct substitution for these test problems, except for test problem 4 where two such iterations are used. The second scheme uses five direct substitutions for all the test problems. This study is performed only on those problems with recycles, i.e. all problems except problem 2 and 5.

Table 4.7 shows that it is helpful to initialise the optimisation calculation by a few direct substitutions, e.g. problem 3, 4, and 6, in agreement with the results of Chen and Stadtherr [1985 (b)]. The purpose of performing direct substitutions is to provide better estimates since tear variables are in general initialised by the users.

Table 4.7 Effect of Initialisation of Tear Variables

Problem No.	TV I		TV II	
	NG (NF)	CPU (STE)	NG (NF)	CPU (STE)
P 1	7 (7)	0.6 (4.0)	7 (7)	0.6 (4.1)
P 2	-	-	-	-
P 3	9 (11)	3.4 (11.8)	7 (12)	2.9 (10.3)
P 4	10 (16)	7.7 (22.6)	3 (5)	2.7 (8.0)
P 5	-	-	-	-
P 6	9 (12)	7.2 (17.9)	6 (10)	5.1 (12.7)
P 7	10 (10)	9.1 (26.6)	10 (10)	9.1 (26.6)

Key

TV I	Tear variables are initialised by two direct substitutions for problem P 4 and one direct substitution for all other problems.
TV II	Tear variables are initialised by five direct substitutions.
NG (NF)	Number of gradient (function) evaluations.
Machine	VAX 8530 .
CPU	Central processing unit time (min).
STE	Simulation time equivalent.

4.4.1.3 Line Search and Trust Region Methods

Two line search strategies and a trust region method are compared. The first strategy implements the line search strategy with a modified augmented Lagrangian function, equation (2.29), as a merit function. The second strategy uses an absolute exact penalty function, equation (2.20), as a merit function with a watchdog technique, algorithm B.II. The absolute exact penalty function is also used as a merit function for the trust region method in the third strategy.

Table 4.8 show that these strategies perform equally well. The use of an augmented Lagrangian function may result in accepting an improper step length which causes a large number of function evaluations in test problem 6.

Table 4.8 Comparison of Line Search and Trust Region Methods

Problem No.	LSH I		LSH II		TR I	
	NG (NF)	CPU (STE)	NG (NF)	CPU (STE)	NG (NF)	CPU (STE)
P 1	9 (10)	0.7 (5.0)	7 (7)	0.6 (4.0)	7 (7)	0.6 (4.2)
P 2	16 (32)	12.5 (33.3)	15 (31)	11.7 (31.3)	16 (29)	12.0 (32.)
P 3	11 (11)	4.0 (14.3)	9 (11)	3.4 (11.8)	8 (9)	3.0 (10.7)
P 4	9 (11)	6.5 (19.1)	10 (16)	7.7 (22.6)	8 (10)	5.9 (17.3)
P 5	6 (6)	4.4 (15.4)	6 (6)	4.4 (15.4)	6 (8)	4.8 (16.8)
P 6	11(16)	8.9 (22.2)	9 (12)	7.2 (17.9)	10(12)	7.7 (19.2)
P 7	15 (20)	14.0 (39.5)	10 (10)	9.1 (26.6)	12 (13)	11.0 (32.2)

Key

- LSH I A Line search method with a modified Lagrangian as a merit function.
- LSH II A Line search method with an absolute exact penalty function as a merit function.
- TR I A trust region method with an absolute exact penalty function as a merit function.
- NG Number of gradient evaluations.
- NF Number of function evaluations.
- Machine VAX 8530 .
- CPU Central processing unit time (min).
- STE Simulation time equivalent.

4.4.1.4 Comparison of Optimisation Strategies

This section compares the various optimisation strategies including a feasible path variant, an infeasible path, and an infeasible path hybrid approach. This comparison is made only for those problems with recycles.

Table 4.9 shows that the feasible path method requires fewer gradient evaluations than the infeasible path methods for these test problems, but each iteration is more expensive since a complete flowsheet simulation is required for every function evaluation.

Table 4.9 Comparison of Optimisation Strategies for a *MO* Approach

Problem No.	FP		IP		IPH	
	NG (NF)	CPU (STE)	NG (NF)	CPU (STE)	NG (NF)	CPU (STE)
P 1	5 (5)	0.8 (5.8)	7 (7)	0.6 (4.0)	7 (22)	0.9 (6.4)
P 2	-	-	15 (31)	11.7 (31.3)	-	-
P 3	6 (12)	4.1 (14.5)	9 (11)	3.4 (11.8)	5 (16)	2.4 (8.6)
P 4	5 (10)	7.2 (20.5)	10 (16)	7.7 (22.6)	3 (8)	2.9 (8.7)
P 5	-	-	6 (6)	4.4 (15.4)	-	-
P 6	4 (15)	4.5 (11.8)	9 (12)	7.2 (17.9)	6 (21)	6.0 (15.0)
P 7	4 (10)	6.3 (18.6)	10 (10)	9.1 (26.6)	9 (30)	8.7 (25.5)

Key

FP	The feasible path approach.
IP	The infeasible path approach.
IPH	The infeasible path hybrid approach.
NG (NF)	Number of gradient (function) evaluations.
Machine	VAX 8530 .
CPU	Central processing unit time (min).
STE	Simulation time equivalent.

The increased number of gradient evaluations in the infeasible path method may be explained by the nonlinearity of the tear equations. However, as shown in the results on Cavett's problem, when the number of tear variables increases, the effort required for the gradient evaluations, the storage requirement for the Jacobian matrix of the tear equations and the computational efforts in solving a

QP subproblem become increasingly significant. In this case, the feasible path method may become more efficient as shown in test problem 6.

The feasibility improvements reduce the number of gradient evaluations at the cost of increasing function evaluations. This strategy may become more efficient when the tear equations are highly curved, e.g problem 4.

4.4.2 Process Optimisation with an *EO* Approach

The *SPEEDUP* package, [Pantelides, 1988 (a)] an equation oriented simulator developed at the Imperial College, is employed to study the performance of the proposed decomposition algorithm.

Eight process optimisation problems are used and the characteristics of these problems are summarised in Table 4.10.

Test problem 8, an alkylation process, which was suggested by Sauer *et al.* [1964], is shown in Figure 3.1. Olefin feed and isobutane make-up are introduced into a reactor. Fresh acid is added to catalyse the reaction and spent acid is withdrawn. The hydrocarbon product from the reactor is fed to a fractionator, from the top of which isobutane is removed as a recycle to the reactor. Alkylate product is withdrawn from the bottom of the fractionator. The maximum profit serves an objective for this process. This process is described in appendix C.1 in detail.

Test problem 9 is the *Williams-Otto process* [Williams and Otto, 1964]. The process, shown in the Figure 3.2, consists of a stirred-tank reactor, a heat exchanger to cool the reactor outflow, a decanter to separate a heavy oily by-product and a distillation column to separate the product from other components as column top product. A portion of the bottom product of the distillation column is recycled to the reactor, while the remainder is used as plant fuel. The process is to produce a certain amount of a pure product with maximum investment return. The detailed description of this process is list in Appendix C.2.

Test problem 10 is a double-effect evaporator process as shown in Figure 4.13. The process is designed to concentrate sugar solution through two evaporators. Both sugar solution and steam are introduced into the first effect evaporator while the concentrated product is withdrawn from the second effect recycle stream. Maximum profit from the process is described.

Test problem 11, a methanol process, is illustrated in Figure 4.14. This problem was proposed by the organisers of the conference of the *European Federation of Chemical Engineering in Antwerp (May, 1982)* [De Leeuw Den Bouter *et al.* 1983]. A mixture of carbon mono-oxide, hydrogen, and a recycle is fed to a reactor to produce methanol. A fraction of the product is recycled. Maximum profit is the objective of this process.

Test problem 12 seeks maximum profit in a flotation process with seven flotation cells, as described by Chan and Prince [1986]. Figure 4.15 shows this process.

Test problem 13, similar to problem 12, utilises a flotation process with eight flotation cells illustrated in Figure 4.16 [Chan and Prince, 1986].

Test problem 14 is a simplified model for an industrial power generation process. Maximum profit is the objective function for this process.

Test problem 15 is a simplified model for an industrial ethylene process. The objective function of this problem is to maximise the profit.

Table 4.10 Characteristics of Test Problems for an EO Approach

Problem No.	Process	N	MEQ	NZ	OPT
P 8	Alkylation	31	28	68	-1161.34
P 9	Williams-Otto	16	12	70	-98.678
P 10	Methanol Process	111	107	194	-11693.1
P 11	Double Effect Evaporator	74	71	333	-1509.63
P 12	Flotation Process (a)	63	56	211	-439.66
P 13	Flotation Process (b)	83	75	267	-1040.55
P 14	Power Generation Process	174	157	533	1132.618
P 15	Ethylene Process	401	392	1413	23863.8

Key :

- N Number of variables.
- MEQ Number of equality constraints.
- NZ Number of non-zero elements in the Jacobian matrix.
- OPT Value of the objective function.

Figure 4.13 Double Effect Evaporator Process Flowsheet

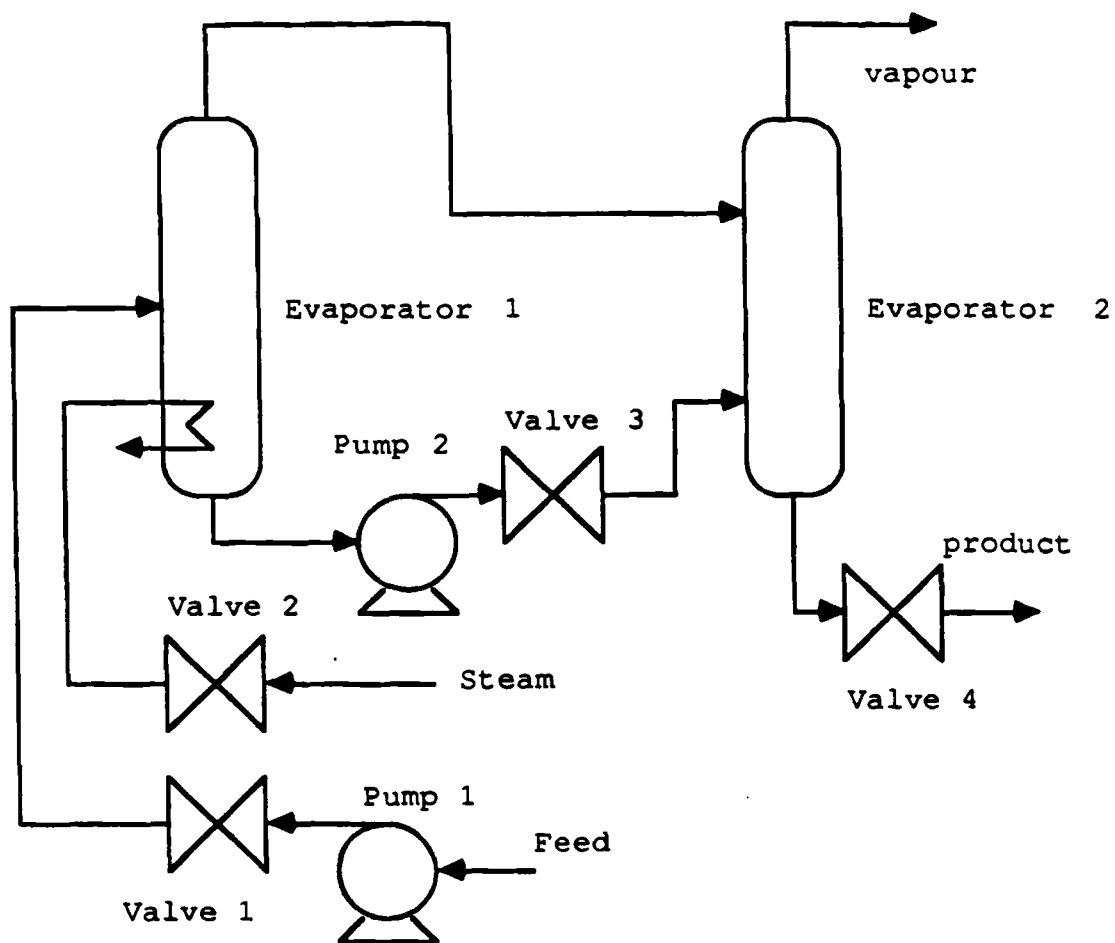


Figure 4.14 Methanol Process Flowsheet

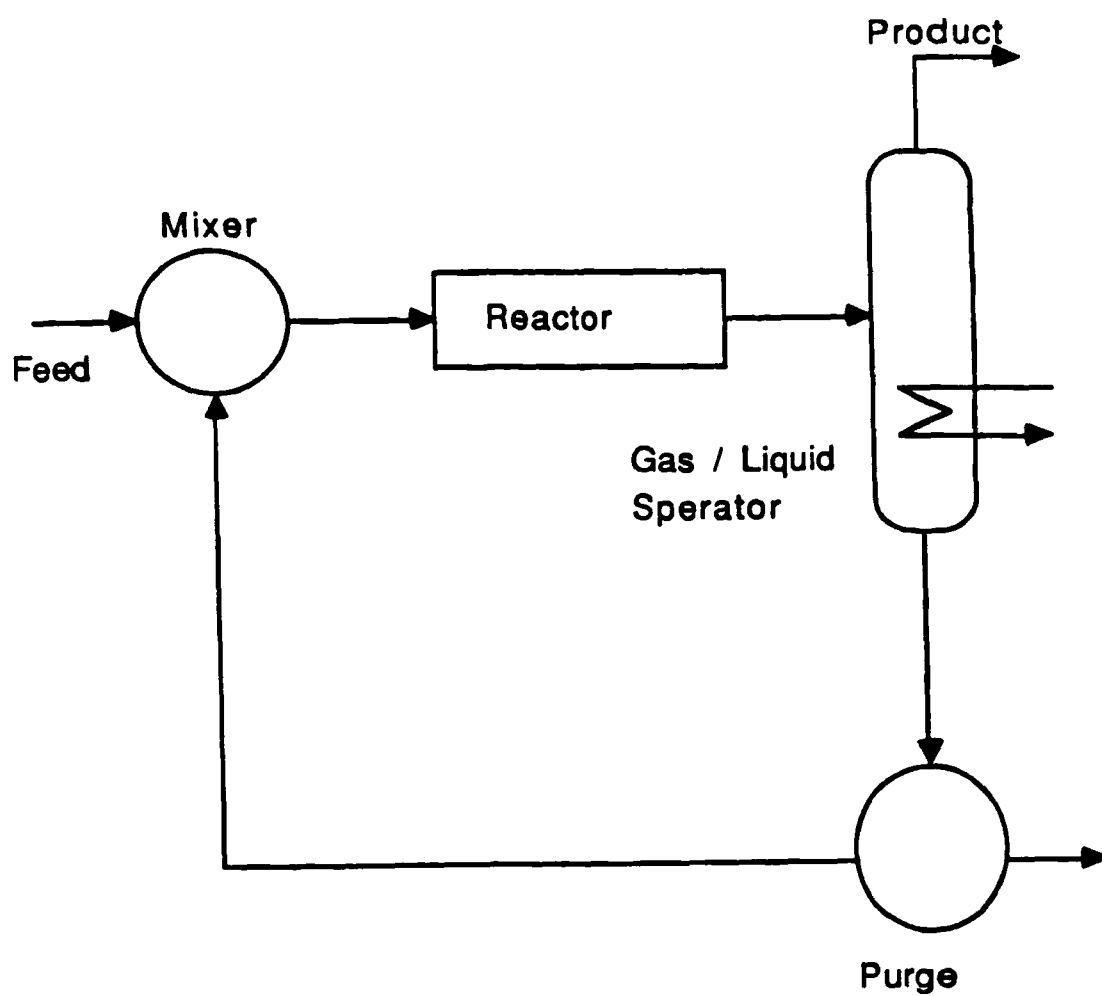


Figure 4.15 Flotation Process Flowsheet (a)

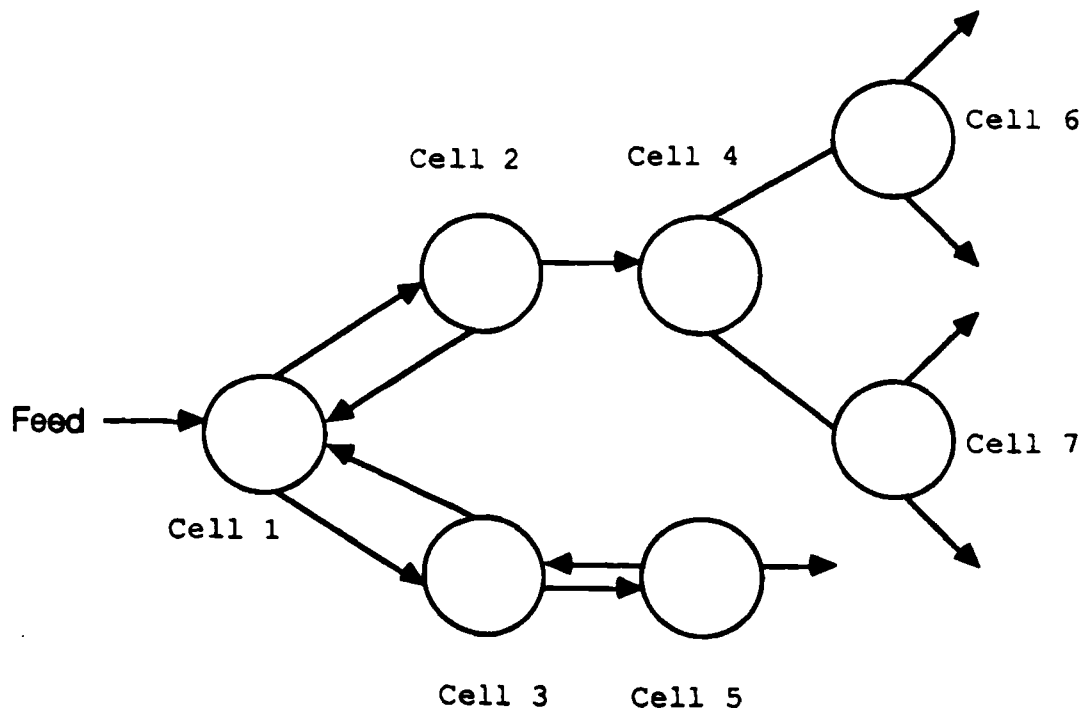
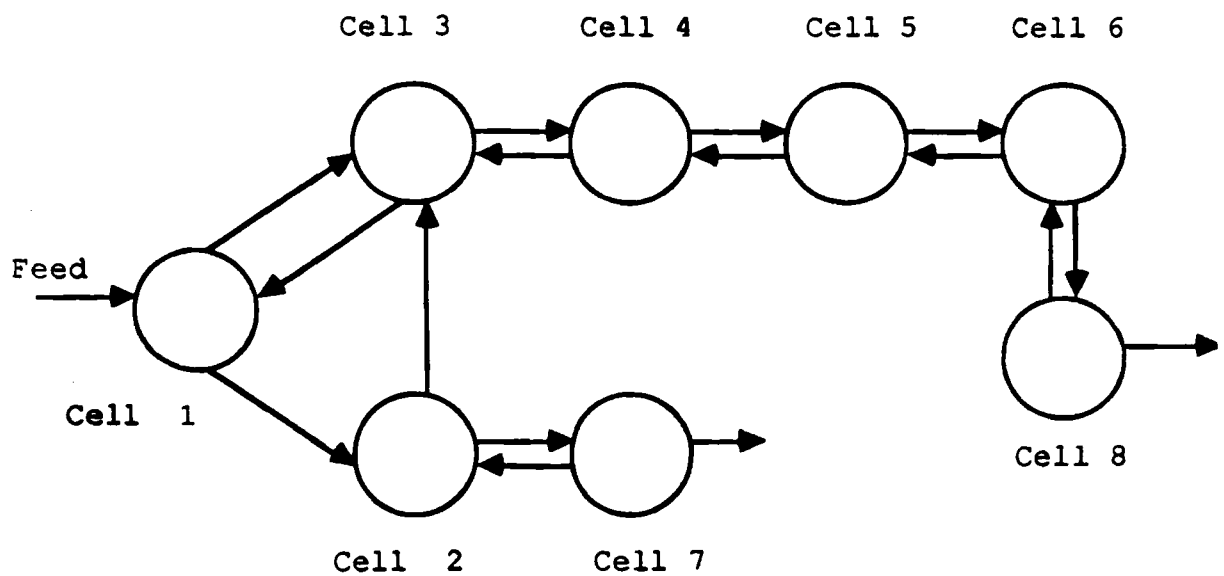


Figure 4.16 Flotation Process Flowsheet (b)



The infeasible base points are generated based on the simulator initialisation scheme, i.e. process variables are initialised based on the variables type which categorises each variable. Variables of the same type will be initialised to the same value unless specified by the users. Feasible starting points are generated by performing a flowsheet simulation with tolerance 10^{-5} . Infeasible starting points for problem 14 and 15 are generated by performing partial flowsheet simulations to an absolute tolerance of 10^5 from the infeasible base points.

Analytical gradients are generated from symbolic manipulations in this study [Pantelides, 1988 (b)] whenever possible. The procedure derivatives are evaluated by finite difference in the test problems 14 and 15. An artificial variable is introduced to define the objective function, i.e.

$$\begin{aligned}
 & \min_{x, e_1, e} e_1 \\
 & \text{s.t. } h(x) = 0 \\
 & e_1 - f(x) = 0 \\
 & g(x) - e = 0 \\
 & x_\ell \leq x \leq x_u \\
 & 0 \leq e \leq \infty
 \end{aligned}
 \tag{4.8}$$

where

e_1 is a scalar artificial variable.

e is an (m_i) -dimensional vector of variables.

x_ℓ is an n -dimensional vector of lower bounds for variables x

x_u is an n -dimensional vector of upper bounds for variables x

All the variables are bounded in the *SPEEDUP* simulator.

The default options used in this study are

1. The Hessian of the Lagrangian function is initialised as an identity matrix.
2. The reduced model I is used.
3. An absolute value penalty merit function, equation (2.20), with relaxation, equation (3.24), is employed.
4. No feasibility improvements are performed.

All the variables are bounded by their corresponding upper and lower bounds. An absolute tolerance of 10^{-7} is used for the first six problem while 10^{-5} is used for the last two problems which involve procedures for calculating physical properties.

The computer employed for this study is *IBM 4341* (machine precision is 2.2×10^{-6}) and double precision is used.

4.4.2.1 Comparison of Reduced Models

The three models for large scale optimisation proposed in Chapter 3 are compared. An identity matrix is used for the initial Hessian since the starting points are arbitrary except for test problems 8 and 9.

Table 4.11 shows the performance of the different models. The *reduced model II* and *reduced model III* fail to solve the methanol process when the algorithm starts with an infeasible point. The reason is that the constraints in this process are linearly dependent at this starting point and hence a large step is produced. This causes a failure in the line search. However, a good solution is obtained when starting from a feasible steady state point.

The *reduced model III* is more expensive in terms of *CPU* time since it requires the solution of two reduced *QP* subproblem. It may, however, require less gradient evaluations, e.g. test problems 9, 10 and 12.

The increase in gradient evaluations for *reduced model II* in test problems 9, 12, and 15, and for the *reduced model III* in problems 1, 11, 15 indicates that replacing the Hessian of the Lagrangian function in calculating the ' feasibility step ' by an identity matrix does not always result in improved performance since it is assumed that the variables have the same order of the magnitude. It was shown in Table 4.11 that scaling the Hessian was very beneficial for test problems 8, 9, 14, 15. This also indicates that these problems are badly scaled and scaling the reduced Hessian results in a reduced number of gradient evaluations.

Since both process and design variables are involved in the optimisation level, initialisation of process variables plays an important role. To avoid physically unattainable starting points a partial or complete simulation for given design variables is recommended to initialise the process variables, as employed in test problem 14 and 15.

Table 4.11 Comparison of Reduced Models

Problem No.	RM I NG (NF) CPU	RM II NG (NF) CPU	RM III NG (NF) CPU
P 8	26 (28) 1.9	27 (29) 2.0	31 (32) 2.3
P 8 (Y)	11 (11) 0.878	17 (24) 1.4	11 (13) 1.1
P 9	65 (74) 3.2	72 (108) 4.5	51 (64) 3.6
P 9 (Y)	49 (56) 2.7	67 (95) 4.1	41 (61) 2.4
P 10	26 (32) 6.2	Failed	Failed
(ss)	18 (24) 5.6	22 (32) 6.2	17 (21) 5.9
P 11	18 (20) 3.2	15 (17) 2.6	19 (26) 3.8
P 12	57 (75) 12.8	72 (140) 15.8	43 (70) 13.9
P 13	49 (60) 16.4	51 (78) 18.4	49 (75) 19.0
P 14	73 (76) 163.6	37 (37) ^d 78.4	86 (93) 249.3
(ss)	81 (89) 187.4	42 (42) ^d 88.1	64 (65) 171.9
(Y)(ss)	35 (38) 86.0	30 (36) ^d 76.6	47 (59) 139.1
P 15	40 (40) 136.4	48 (48) 163.9	48 (48) 176.4
(ss)	41 (41) 141.2	50 (50) 170.3	47 (47) 170.9
(Y)(ss)	17 (23) 61.1	18 (25) 63.3	18 (25) 70.8

key :

RM I	Reduced model I
RM II	Reduced model II
RM III	Reduced model III
NG (NF)	Number of gradient (function) evaluations.
ss	Starting with a feasible point.
<i>d</i>	Terminate at a different optimum.
Y	Scaling option is used.
Machine	IBM 4341
CPU	Central Processing Unit time (sec).

4.4.2.2 Comparison of Merit Functions

The two line search strategies proposed in Chapter 3 are examined here.

Table 4.12 shows that both the exact penalty function with relaxation and the augmented Lagrangian function can be efficiently employed as merit functions in the line search.

Table 4.12 Comparison of Merit Functions

Problem No.	EP		AP	
	NG (NF)	CPU(sec)	NG (NF)	CPU(sec)
P 8	26 (28)	1.9	29 (30)	2.2
P 9	65 (74)	3.2	45 (47)	2.4
P 10	26 (32)	6.2	26 (27)	5.9
P 11	18 (20)	3.2	18 (19)	3.1
P 12	57 (75)	12.8	85 (105)	17.6
P 13	49 (60)	16.4	61 (66)	23.5
P 14 (ss)	81 (89)	187.4	71 (72)	159.9
P 15 (ss)	41 (41)	141.2	43 (66)	150.0

Key

EP	Absolute value penalty merit function, equation (2.20).
AP	Augmented Lagrangian merit function, equation (2.29).
NG	Number of gradient evaluations.
NF	Number of function evaluations.
Machine	IBM 4341
CPU	CPU time (sec).

4.4.2.3 Comparison of Optimisation Strategies

A feasible path approach is offered as a standard option in *SPEEDUP*. This options utilises the modified *VMCWD* code described in Chapter 2.

Compared with Table 4.11, Table 4.13 show that the infeasible path strategy with feasibility improvements requires fewer gradient evaluations. However, it may be difficult to solve a system of equations with inequality constraints to a prescribed tolerance, as observed in the test problem 12 and 13. The infeasible path strategies with feasibility improvement reduces the number of gradient evaluations in the optimisation level due to the use of the reduced Hessian. When the gradients are expensive to evaluate, the feasibility improvement without analytical derivatives is favoured. The feasible path method fails in three cases due to the inability of solving a simulation problem for some value of design variables.

The simulation time equivalent *STE* is not used here as a measure of merit. The computational time spent in the optimisation algorithm is dominant for all test problems since no expensive procedures are involved in these test problems. Thus the *CPU* time reflects well the efficiency of the optimisation algorithm. The *CPU* time distribution for test problem 12, shown in Table 4.14, indicates that most of the *CPU* time is spent on the factorisation of the Jacobian matrix and the solution of the reduced *QP* subproblem. The *CPU* time therefore increases with the size of the problems and the number of design variables.

Table 4.13 Comparison of Feasibility Strategies for an *EO* Approach

Problem No.	FP			IPH 1		
	NG(NF)	NTG (NTF)	CPU	NG(NF)	NTG(NTF)	CPU
P 8	21(21)	110 (110)	6.3	25(30)	44(49)	2.4
P 9	Failed			46(53)	88(95)	5.7
P 10	6(10)	71(75)	19.5	26(32)	26(32)	6.3
P 11	12(12)	75(75)	12.4	14(15)	21(22)	2.9
P 12	29(29)	195(195)	32.7	32(40)	63(71)	10.7
P 13 (ss)	Failed			50(58)	93(101)	20.9
P 14	Failed			63(66)	142(144)	201.9
P 15 (ss)	38(38)	190(2030)	184.4	40(40)	77(77)	189.9

Table 4.13 Comparison of Feasibility Strategies (Cont'd)

Problem No.	IPH 2		CPU
	NG (NF)	NTG (NTF)	
P 8	24 (29)	24 (54)	2.0
P 9	46 (49)	49 (136)	4.9
P 10	26 (32)	26 (32)	6.1
P 11	14 (14)	14 (26)	3.1
P 12	37 (53)	37 (92)	10.4
P 13	49 (56)	49 (98)	18.9
P 14 (ss)	40 (42)	40 (109)	122.4
P 15 (ss)	64 (67)	64 (198)	232.0

Key

FP	Feasible path approach.
IPH 1	Infeasible path hybrid with gradients evaluated during feasibility improvement (Algorithm C.II).
IPH 2	Infeasible path hybrid with gradients remaining constants during feasibility improvement (Algorithm C.II).
NG	Number of gradient evaluations in the optimisation level.
NTG	Total number of gradient evaluations.
NF	Number of function evaluations in the optimisation level.
NTF	Total number of function evaluations.
Machine	IBM 4341
CPU	CPU time (sec).

Table 4.14

The CPU Time Distribution on Test Problem 12

Step No.	CPU (sec)	Percentage
1	0.237	2.28
2	3.609	34.67
3	1.427	13.71
4	0.147	1.41
5	2.924	28.09
6	1.137	10.92
7	0.092	0.88
G	0.644	6.19
F	0.157	1.51
other	0.035	0.34
Total	10.409	100.

Key

Step 1	Initialisation Section.
Step 2	Factorisation of the Jacobian of the equality constraints and formulation of quadratic program.
Step 3	Solution of quadratic program for search direction, estimation of Lagrangian multipliers.
Step 4	Line search section.
Step 5	Formulation of the quadratic program for feasibility step
Step 6	Solution of the quadratic program for feasibility step
Step 7	Update of the Hessian.
Step F	Function evaluations.
Step G	Gradient evaluations.
Other	Other parts of program, e.g. bound checking.
Machine	IBM 4341
CPU	CPU time (sec).

4.4.2.4 Comparison with an Optimisation Code - *MINOS*

MINOS (V 5.1) is regarded as one of the most effective optimisation codes for large scale optimisation. The code has been interfaced to *SPEEDUP* by the *SPEEDUP* team at the Imperial College.

The *MINOS* code separates the variables into linear and nonlinear variables and also identifies the constraints as linear and nonlinear constraints. It also provides a number of options. The options examined here are those for scaling the linear and nonlinear variables, the option for choosing the initial step length parameter for the search direction, and the option for solving the linearly constrained subproblem partially or completely. These are essential options which can be specified by the users.

Table 4.15 shows that the performance of *MINOS* on these test problems is very sensitive to the selected values of the optional parameters. Scaling in general improves the efficiency the optimisation. However, it is not trivial to provide a suitable scaling scheme for a given optimisation problem. The numerical results on test problem 12 shows that the automatic scaling may actually be harmful. An initial step size should be kept small (less than one) for a highly curved optimisation problem e.g. problem 9. Solving the linearly constrained subproblem partially is found to be more effective for these test problems.

Comparing Table 4.11 and Table 4.15 shows that the proposed *SRQP* algorithm performs well in these test problems in terms of efficiency and robustness. The results also show that the proposed algorithms are not sensitive to the tuning parameters and the options used. This desirable feature is essential to develop a user friendly optimisation code.

Table 4.15 Performance of the MINOS (v. 5.1)

Problem	Scale	Step	Comp	NT(MT)	NF	CV	CPU
P 8	No	2.(D)	F	12(55)	117	10^{-13}	3.3
	No	2.(D)	P	11(49)	101	10^{-13}	2.3
	ALL	2.(D)	F	7(29)	52	10^{-7}	2.4
	ALL	2.(D)	P	8(25)	49	10^{-14}	2.4
P 9	No	2.(D)	P	Failed			
	No	2.(D)	F	27(309)	633	10^{-11}	10.7
	No	0.5	F	31(364)	739	10^{-7}	12.7
	No	0.5	P	117(424)	1217	10^{-6}	20.3
	Lin	0.5	F	31(365)	734	10^{-5}	13.1
	ALL	0.5	P	18(90)	194	10^{-9}	4.4
	ALL	0.5	F	15(133)	334	10^{-9}	6.5
	ALL	2.	P	23(135)	462	10^{-9}	8.183
	ALL	2.(D)	F	14(125)	294	10^{-9}	6.1
	ALL	0.3	F	19(173)	400	10^{-12}	7.9
P 10	No	2(D)	F	Failed			
	ALL	2(D)	F	Failed			
	No	0.3	F	Failed			
(ss)	No	2(D)	P	17(72)	120	10^{-8}	13.5
(ss)	No	2(D)	F	17(77)	124	10^{-8}	13.8
(ss)	ALL	2(D)	P	16(41)	39	10^{-8}	8.1
(ss)	ALL	2(D)	F	9 (39)	27	10^{-8}	6.5
P 11	No	2(D)	P	24(101)	207	10^{-12}	12.9
	No	2(D)	F	23(114)	222	10^{-12}	13.2
	ALL	2(D)	P	8(27)	19	10^{-9}	3.6
	ALL	2(D)	F	8(27)	19	10^{-9}	3.6

Table 4.15 The performance of the MINOS (continued)

Problem	Scale	Step	Comp	NT(MT)	NF	CV	CPU
P 12	No	2(D)	P	Failed			
	No	0.3	P	28(360)	572	10^{-7}	30.4
	No	0.3	F	34(596)	1032	10^{-9}	52.3
	ALL	0.3	P	Failed			
	ALL	0.3	F	Failed			
	No	1.0	F	Failed			
P 13	No	0.3	P	30(618)	951	10^{-12}	57.7
	No	0.3	F	27(657)	1082	10^{-8}	60.7
	No	2(D)	P	18(168)	222	10^{-12}	17.4
	ALL	2(D)	P	Failed			
	ALL	0.3	P	36(762)	1151	10^{-12}	72.6
	ALL	0.3	F	24(576)	745	10^{-13}	50.2
P 14 (ss)	No	Failed					
(ss)	Yes	2	F	12(151)	168	10^{-12}	109.3
P 15 (ss)	No	2	F	28(517)	1771	10^{-13}	2908.3
(ss)	ALL	2	F	9(219)	749	10^{-13}	1253.5

Key

Scale	Scale option for variables.
All	Scale all variables.
Lin	Scale linear variables.
Step	Damping parameters for the step length.
Comp	Solution methods of the subproblem.
F	The subproblem is solved completely.
P	The subproblem is solved partially.
NT	Number of the major iterations.
MT	Total number of iterations.
NF	Number of function evaluations.
CV	Constraint violations.
CPU	CPU time (sec)

Machine	<i>IBM 4341</i>
D	Default value
ss	Steady state starting points.

4.5 Discussion and Conclusions

In this chapter, the optimisation methods developed in previous chapters were successfully implemented and studied on two types of simulators.

In a sequential modular simulator, the efficiency of flowsheet optimisation depends mainly on the strategy for gradient evaluations, since the unit calculations are in general expensive in terms of computation time and gradients are in practice obtained from perturbation. The results show that the flowsheet perturbation in the feasible path variant approach is not as prohibitively expensive as it is commonly thought to be. It appears that the tear equations in the simulation level are relatively easy to converge when the design variables are perturbed. The total computational time required by this strategy depends on the number of design variables but not on the number of tear variables. This strategy becomes useful when the number of design variables is far smaller than the number of tear variables. The *STE* for these problems varies from 5 to 30, which is comparable to those results cited in the literature and acceptable to the industry.

The use of line search and trust region methods can achieve similar efficiency and robustness.

For these test problems, the three models of the *SRQP* method are effectively implemented and can be used to generate the search directions. They can achieve similar efficiency.

The proposed line search methods by using either the absolute value exact penalty function or augmented Lagrangian function appear to be effective for these test problems.

The proposed penalty function method is found to be useful to deal with constraint inconsistency in both simulators. This implies that the constraints can be classified as 'soft' and 'hard' constraints.

The proposed projection method for the feasibility improvement is found to be helpful. This feasibility improvement improves the robustness and/or efficiency especially in the case of highly nonlinear constraints at the expense of extra function evaluations. The overall efficiency of the optimisation method with feasibility improvement depends on the cost of gradient evaluations.

Constrained optimisation implicitly involves two objectives, namely improving the objective function and satisfying constraints. This feature complicates the issues in developing a robust optimisation method unless all the constraints are linear. This work however shows that a robust method can be developed by exploiting the properties of these two objectives.

From the numerical results in Chapters 2, 3 and 4, it can be observed that the performance of optimisation methods strongly depends on the algorithmic implementations and types of test problems. The results in Chapter 2 show that it is necessary to impose the step bounds and/or feasibility refinement, and scale the initial Hessian and/or the objective function for some cases. The effectiveness of these improvements depends on the nonlinearity and conditioning of the problems.

Exploiting the structure of the problem results in an efficient method, as shown in Chapter 3. Consistent results obtained with default tuning parameters for the proposed algorithms indicate that these parameters are not critical.

In summary, the following options are recommended for the flowsheet optimisation on *MO* approach,

1. identify the constraints as ' hard constraints ' and ' soft constraints ' from the physical properties of the problem. Transform the ' hard constraints ' into equality constraints and simple bounds via artificial variables where possible.
2. choose tear variables automatically or by heuristic rules mentioned in this chapter.
3. obtain a sensible initial estimate, if not available, by one or two direct substitutions.
4. estimate the perturbation size by balancing the truncation error, procedure error and rounding error.
5. select the gradient evaluation strategy by comparing their cost.
6. scale the initial Hessian by equation (2.64).
7. impose the step bounds as suggested in this chapter.

8. impose the feasibility steps for hard constraints and highly nonlinear constraints.
9. perform feasibility improvement when the line search fails.

For flowsheet optimisation on an *EO* approach, the following options are recommended

1. analyse and identify the linearity of variables.
2. choose the nonlinear variables as design variables where possible. Automatic choice of design variables can be used when the types of variables are not available.
3. obtain reasonable starting points by a partial simulation.
4. scale the initial Hessian by using equation (2.65).
5. perform feasibility improvements when expensive procedures are present in order to reduce the number of gradient evaluations.

Chapter 5

Sensitivity Analysis

5.1 Introduction

In Chapter 2, the basic and improved *SQP* methods were examined. A class of *SRQP* methods for the solution of large scale optimisation problems was proposed in Chapter 3. Numerical results on flowsheet optimisation problems showed the effectiveness of both methods in Chapter 4. So far, the solution of isolated problems was considered. However, process engineers would typically like to perform multiple case studies and to relate solution changes to data changes because of the uncertainties and possible variations involved in process parameters. This motivation leads to a parametric sensitivity analysis of the optimisation solutions. This problem has been recently reviewed by Fiacco [1983].

It is also useful to relate the change of the objective function to the variations of decision variables. This leads to the development of a decision variable sensitivity analysis technique, which was proposed by McKeown [1980], using the eigenvalue system of the Hessian of the objective function in the reduced space of points that satisfy the constraints.

It is also desirable to use the sensitivity information to solve a set of similar problems efficiently. Similarly, it is also possible to devise an efficient algorithm by using information from previous iterations. The warm start technique arose for this purpose [Gill *et al.* 1985 (b)].

In addition, since a rigorous process model may be difficult and expensive to evaluate, a simplified (approximate or reduced) model is often used in the flowsheet simulation [Evans, 1982]. However, in flowsheet optimisation, Biegler *et al.* [1985 (a)] showed that the use of a simplified model may cause the optimisation methods to identify a different set of active constraints and to converge to a sub-optimal point or non-optimal point, or even cycle instead of converging. It is, therefore, necessary to re-examine the algorithms that use a simplified model.

These topics, i.e. parametric sensitivity, decision variable sensitivity, warm start procedure and the use of simplified models for flowsheet optimisation, constitute the theme of this chapter. The benefit of this extra information may help

process engineers to analyse and understand their optimisation problems and to solve them more efficiently.

In this chapter, we concentrate on obtaining sensitivity information from the SQP methods. We start with the consideration of the parametric sensitivity and decision variable sensitivity technique. The implementation of the warm start procedure for the SQP method is then discussed. In addition, the use of simplified models for flowsheet optimisation is investigated. Recently, Ganesh and Biegler [1987 (b)] raised an interesting topic on the sensitivity of the optimum to model variations. A pitfall involved in their techniques is remarked. Numerical results which use these techniques are presented at the end of this chapter.

5.2 Parametric Sensitivity Analysis

Consider the following parametric optimisation problem,

$$\begin{array}{ll}
 \min_x & f(x, b) \\
 \text{P E.I} \quad \text{s.t.} & h_i(x, b) = 0 \quad i = 1, \dots, meq \\
 & g_j(x, b) \geq 0 \quad j = 1, \dots, mi
 \end{array} \quad (5.1)$$

where

x is an n -dimensional vector of variables.

$f(x, b)$ is a scalar objective function.

b is a set of nb parameters.

$h_i(x, b)$ is an equality constraint.

$g_j(x, b)$ is an inequality constraint.

meq is the number of equality constraints.

mi is the number of inequality constraints.

We are interested in finding how the optimal value of the objective function $f(x^*)$, and the solution vector x^* change with the parameters b without re-solving the optimisation problems.

Linearising the first order optimality conditions of the parametric optimisation problem P E.I with respect to the parameters b at the optimum x^* and ignoring the higher order terms gives [Fiacco, 1983]

$$M(b) X(b) = N(b) \quad (5.2)$$

where

$$M(b) = \begin{pmatrix} \nabla_{xx}^2 L & \dots & -\nabla_x h_i^T & \dots & \dots & -\nabla_x g_j^T & \dots \\ \vdots & & & & & & \\ -\nabla_x h_i & & 0 & & & 0 & \\ \vdots & & & & & & \\ \vdots & & \ddots & & & & \\ -\nu_j \nabla_x g_j & & g_j & & & 0 & \\ \vdots & & & & \ddots & & \end{pmatrix}$$

$$X(b)^T = (\nabla_b x^T, \nabla_b \lambda^T, \nabla_b \nu^T)$$

$$N(b) = (-\nabla_{bx}^2 L^T, \dots, -\nabla_b h_i^T, \dots, \dots, -\nu_j \nabla_b g_j^T, \dots)$$

with

$$i = 1, \dots, meq.$$

$$j = 1, \dots, mi.$$

b is an (nb) -dimensional set of parameters.

M is an $(n + meq + mi)$ square matrix.

X is an $(n + meq + mi) \times nb$ matrix.

N is an $(n + meq + mi) \times nb$ matrix.

Based on the assumption that the constraints are linearly independent at the optimum, the Lagrangian multipliers for the inactive constraints are zero at the optimum because of the complementary slackness conditions,

$$\begin{aligned} \lambda_i h_i(x) &= 0, & i &= 1, \dots, meq \\ \nu_j g_j(x) &= 0, & j &= 1, \dots, mi \end{aligned} \quad (5.3)$$

It can be observed from equations (5.2) and (5.3) that the derivatives of those inactive inequality constraints with respect to parameters b are zero, i.e.

$$\nabla_b \nu_k = 0, \quad k \in K_g$$

where K_g is defined as

$$K_g = \{k \mid g_k(x, b) > 0, \quad k = 1, \dots, mi\}$$

This implies that the set of active constraints remains unchanged and hence the inactive constraints can be ignored. The system equation (5.2) can thus be reformulated as follows,

$$\widehat{M}(b) \widehat{X}(b) = \widehat{N}(b) \quad (5.4)$$

where

$$\begin{aligned}\widehat{M}(b) &= \begin{pmatrix} \nabla_{xx}^2 L & -\nabla_x c^T \\ -\nabla_x c & 0 \end{pmatrix} \\ \widehat{X}(b)^T &= (\nabla_b x^T, \nabla_b \pi^T) \\ \widehat{N}(b) &= (-\nabla_{bx}^2 L^T, \nabla_b c^T)\end{aligned}$$

ma is the number of active constraints.

$\widehat{M}$ is an $(n + ma)$ square matrix.

$\widehat{X}$ is an $(n + ma) \times nb$ matrix.

$\widehat{N}$ is an $(n + ma) \times nb$ matrix.

c is an (ma) -dimensional vector of active constraints.

$\pi = \{\lambda_i, i = 1, \dots, meq; \nu_k, k \in K_h\}$

$K_h = \{j \mid g_j(x, b) = 0, j = 1, \dots, mi\}$

It is assumed that the Hessian of the Lagrangian function is non-singular and the number of active constraints is less than the number of variables. The system can be rewritten as

$$\widehat{X}(b) = \bar{M}(b)\widehat{N}(b) \quad (5.5)$$

where

$$\begin{aligned}\bar{M}(b) &= \begin{pmatrix} U_1 & -U_2 \\ -U_2^T & U_3 \end{pmatrix} \\ X(b)^T &= (\nabla_b x^T, \nabla_p \pi^T) \\ \widehat{N}(b) &= (-\nabla_{bx}^2 L^T, \nabla_b c^T) \\ U_1 &= Z(Z^T \nabla_{xx}^2 L Z)^{-1} Z^T \\ U_2 &= (I - U_1 \nabla_{xx}^2 L)Y \\ U_3 &= -Y^T \nabla_{xx}^2 L U_2 \\ Z^T \nabla_x c &= 0 \\ Y^T \nabla_x c &= I\end{aligned}$$

$\bar{M}$ is an $(n + ma)$ square matrix.

U_1 is an $n \times n$ matrix.

U_2 is an $n \times ma$ matrix.

U_3 is an $ma \times ma$ matrix.

I is an $n \times n$ identity matrix.

Z is an $n \times (n - ma)$ matrix.

Y is an $n \times (n - ma)$ matrix.

These equations (5.4) and (5.5) can be used to calculate the sensitivity of the solution vector (x^*, λ^*, ν^*) with respect to parameters b .

In fact, the sensitivity of the variables x with respect to parameters b can be obtained from solving the following QP problem

$$\begin{aligned} \min_{\nabla_{b,x}} \quad & \nabla_{b,x}^2 L^T \nabla_{b,x} + \frac{1}{2} \nabla_{b,x_i}^T \nabla_{xx}^2 L \nabla_{b,x} \\ \text{s.t.} \quad & \nabla_{b,c} + \nabla_x c^T \nabla_{b,x} = 0 \\ \text{for } & i = 1, \dots, nb. \end{aligned} \quad (5.6)$$

When the Hessian of the Lagrangian function is singular at the solution, the following linear program may be solved,

$$\begin{aligned} \min_{\nabla_{b,x}} \quad & \nabla_{b,x}^2 L^T \nabla_{b,x} \\ \text{s.t.} \quad & \nabla_{b,c} + \nabla_x c^T \nabla_{b,x} = 0 \\ \text{for } & i = 1, \dots, n. \end{aligned} \quad (5.7)$$

It is noted that the parametric problem may not be well defined, when the constraints are linearly dependent. Let us consider the following optimisation problem,

$$\begin{aligned} \min_{x_1, x_2} \quad & (x_1 - 4)^2 + (x_2 - 6)^2 \\ \text{s.t.} \quad & x_1 + x_2 \geq 2 \\ & x_1 - x_2 \geq 4 \\ & x_1 \leq b_1 \\ \text{with } & b_1 = 3 \end{aligned}$$

It can be observed that the system has no solution when the parameter b_1 is perturbed to 2.99.

The Hessian in the SQP methods or the projected Hessian in the SRQP method is approximation to the Hessian of the Lagrangian function or the projected Hessian of the Lagrangian function respectively. In general, this (projected) Hessian approximation does not converge to the true (projected) Hessian of the Lagrangian and hence cannot be employed here. In practice the Hessian of the Lagrangian function can be obtained from the following second order difference scheme,

$$\frac{f(x + \zeta_i e_i + \zeta_j e_j) - f(x + \zeta_i e_i) - f(x + \zeta_j e_j) + f(x)}{\zeta_i \times \zeta_j}; \quad i = 1, \dots, n, \quad j = i, \dots, n. \quad (5.8)$$

where e_i is a unit vector, and the perturbation size ζ can be chosen as

$$\zeta_i = \eta_1 \max(\eta_2, |x_i|); \quad i = 1, \dots, n \quad (5.9)$$

where η_1 can be selected as a cubic root of machine precision when no iterative procedures are involved and η_2 can be selected as a small number (say 1).

This strategy requires at least $\frac{n \times (n-1)}{2}$ extra function evaluations. This may be too expensive to be implemented for flowsheet optimisation. The projected Hessian can be obtained from the method proposed by Locke *et al.* [1983 (b)] or from the decomposition of the following augmented system (Appendix E.1)

$$\begin{pmatrix} \nabla_x c & \Lambda & \nabla_{xx}^2 L \\ 0 & & (\nabla_x c \ \Lambda)^T \end{pmatrix} \quad (5.10)$$

where Λ is a permutation matrix. The projected Hessian can also be obtained by perturbing only the independent variables. A flowsheet simulation is required for each perturbation in this case. Because the approximate solution will satisfy the active constraints to the specified tolerance, Ganesh and Biegler [1987 (b)] suggested that the dependent variables be estimated from the linearised active constraints when the independent variables are perturbed. That is

$$\hat{v} + \Delta \hat{v} = \hat{v} + \nabla_{\hat{v}} c(x)^{-T} \nabla_{\hat{u}} c(x)^T \Delta \hat{u} \quad (5.11)$$

where $x = (\hat{u}, \hat{v})$

Schemes that only perturb independent variables are more attractive when a large number of active constraints are present. However, the choice of the perturbation size poses a serious problem especially when the functions are subject to noise and this strategy is thus not considered further.

It can be proved that the sensitivity of the objective function with respect to the parameters, $\nabla_b f$, can be evaluated from the following equation

$$\begin{aligned} \nabla_b f(b) &= \nabla_b f + \nabla_x f \nabla_b x \\ &= \nabla_b f - \lambda^{*T} \nabla_b h - \nu^{*T} \nabla_b g \end{aligned} \quad (5.12)$$

In fact, this can be easily implemented by reformulating the original optimisation problem as follows via extra variables,

$$\begin{aligned} & \min_{x, e} f(x, e) \\ & \text{s.t. } h_i(x, e) = 0 \quad i = 1, \dots, meq \\ & \quad g_j(x, e) \geq 0 \quad j = 1, \dots, mi \\ & \quad e - b = 0 \end{aligned} \quad (5.13)$$

P E.II

where

x is an n -dimensional vector of variables.
 $f(x, e)$ is a scalar objective function.
 b is an nb -dimensional vector of parameters.
 e is an nb -dimensional vector of extra variables.
 $h_i(x, e)$ is an equality constraint.
 $g_j(x, e)$ is an inequality constraint.
 meq is the number of equality constraints.
 mi is the number of inequality constraints.

The Lagrangian multiplier λ_1 corresponding to the equations,

$$e - b = 0$$

gives the sensitivity of the objective function with respect to the parameters b of interest. That is,

$$\nabla_b f(b) = -\lambda_1 \quad (5.14)$$

The objective function at the perturbed point can hence be predicted from the following equation

$$\begin{aligned}
 f(b + \Delta b) - f(b) &\approx \nabla_b f(b)^T \Delta b \\
 &= -\lambda_1^T \Delta b
 \end{aligned} \quad (5.15)$$

This equation (5.15) is of course only valid for small values of Δb since higher order terms are ignored.

This formulation is conceptually simple and is easily implemented in any simulator without changing the interface between the optimisation solver and the simulator. It increases the number of variables but not the number of design variables. When the parameters only appear on the right hand side of the constraints, the Lagrangian multipliers reflect the sensitivity of the objective function with respect to the right hand side of the corresponding constraints.

5.3 Decision Variable Sensitivity Analysis

It is also interesting to determine the changes of the optimum with respect to independent variables. In practice, a local minimum which is sensitive to small variations in the independent variables may be less desirable than one which has higher function values but is less sensitive. This leads to the analysis of the eigenvalues and eigenvectors of the Hessian of the objective function in the reduced space of feasible points [McKeown, 1980]. This is in turn equivalent to analysing

the eigenvalue system of the reduced (projected) Hessian of the Lagrangian function on the active constraints.

The Lagrangian function on the neighbourhood of the optimal point x^* can be approximated as follows,

$$\begin{aligned} f(x^* + \delta x) &= L(x^* + \delta x, \lambda^*, \nu^*) \\ &= L(x^*, \lambda^*, \nu^*) + \\ &\quad \nabla_x L(x^*, \lambda^*, \nu^*)^T \delta x + \delta x^T \nabla_{xx}^2 L(x^*, \lambda^*, \nu^*) \delta x \\ &\quad + o(\|\delta x\|) \end{aligned} \quad (5.16)$$

$$c(x^*) + \nabla_x c(x^*)^T \delta x = 0 \quad (5.17)$$

It is assumed that the approximate solution satisfies the constraints. Equation (5.17) is thus reduced to

$$\nabla_x c(x^*)^T \delta x = 0 \quad (5.18)$$

It is also assumed the Jacobian $\nabla_x c(x^*)$ is full rank, and there exists a matrix, Z , such that the variable δx can be expressed in terms of the independent variables δu . Thus

$$\delta x = Z \delta u \quad (5.19)$$

where

$$Z^T \nabla_x c = 0.$$

Z is an $(n \times (n - ma))$ matrix.

u is a set of $(n - ma)$ independent variables.

Substitution of equation (5.19) into equation (5.16) yields

$$\begin{aligned} L(x^* + \delta x, \lambda^*, \nu^*) &= f(x^*) + \nabla_x L(x^*, \lambda^*, \nu^*)^T Z \delta \hat{u} + \\ &\quad \delta \hat{u}^T Z^T \nabla_{xx}^2 L(x^*, \lambda^*, \nu^*) Z \delta \hat{u} \end{aligned} \quad (5.20)$$

where the higher order terms are ignored and $Z^T \nabla_{xx}^2 L(x^*, \lambda^*, \nu^*) Z$ is the projected (reduced) Hessian of the Lagrangian function. If the estimated solution x^* is close to the true optimum, the norm of the projected gradient, $\| Z^T \nabla_x L(x^*, \lambda^*, \nu^*) \|$, will be small. The projected Hessian of the Lagrangian function is also assumed to be positive semidefinite at the estimated solution. By using a singular value decomposition of the projected Hessian, equation (5.20) is reduced to

$$L(x^* + \delta x, \lambda^*, \nu^*) \approx f(x^*) + \delta \hat{u}^T S^T \hat{D} S \delta \hat{u} \quad (5.21)$$

where S is the matrix whose columns are the normalised eigenvectors of the projected Hessian of the Lagrangian function and $\hat{D}$ is the diagonal matrix whose diagonal elements are the eigenvalues.

It can be observed that the eigenvectors represent the principal axes of the elliptical contour of the Lagrangian function in the reduced space and the length of each principal axis is inversely proportional to the square root of the corresponding eigenvalues. The eigenvector of maximum eigenvalue corresponds to the direction of maximum sensitivity for the objective function, since the projected gradients are approximately zero. If the eigenvectors should happen to coincide with the axes, so that S is a unit matrix, then the eigenvalues reflect the change of the objective function with respect to the corresponding design variables. Otherwise, they represent how a linear combination of independent variables affect the objective function.

5.4 Warm Start Procedures

In practice, it may be required to solve a flowsheet optimisation problem for several values of a set of parameters. To do this efficiently, a solution is first produced with a given value of parameters, and then the sensitivity information of the solution vector with respect to these parameters may be used to initialise the flowsheet optimisation for new values of the parameters.

The *SQP* method requires initial values of variables, the initial Hessian of the Lagrangian function and initial penalty parameters. In addition, it is helpful when the active constraint set is known in advance. The solution of the system, equation (5.5), can be used to initialise the variables and the Lagrangian multipliers.

The previous values of the approximate Hessian of the Lagrangian function, of the penalty parameters, and the predicted active set can be employed. However, this warm start procedure should be discarded when the following conditions occur : when the values of the penalty parameters is unacceptably large, when the Hessian of the Lagrangian function is ill-conditioned; or when the predicted active set results in constraint inconsistency.

A warm start procedure can be stated as

Algorithm E.I — Warm Start Procedure

1. Formulate the optimisation problem in the form of equation (5.13) via the artificial variables.
2. Solve this optimisation problem for a given set of parameters by an optimisation method described in Chapter 3 or 4.
3. Store the following information.
 - i. index of the active constraints.
 - ii. values of the Lagrangian multipliers.
 - iii. approximate Hessian of the Lagrangian multipliers.
 - iv. penalty parameters.

These values will be used to initialise the next optimisation calculation.

4. Obtain the true Hessian of the Lagrangian function.
5. Solve the equation (5.4) or (5.6) for the sensitivity of the variables with respect to parameters.
6. Update the values of the variables and the Lagrangian multipliers subject to the changes of the parameter values, i.e.

$$x^+(b + \delta b) = x(b) + \nabla_b x \delta b.$$

$$\pi^+(b + \delta b) = \pi(b) + \nabla_b \pi \delta b.$$

These values can be used to initialise the next optimisation calculation.

Since it may be difficult to evaluate the true Hessian of the Lagrangian function, the approximate Hessian at the solution after the previous problem is used in this study.

In the SQP it is necessary to solve a QP subproblem at every optimisation step and each subproblem can also be viewed as perturbed problem. It is also possible to use the previous information to solve the quadratic program efficiently, and this has been reviewed by Gill *et al.* [1986].

5.5 Use of Simplified Models

As processing models have become more complex, engineers often resort to simplified (approximate, reduced, or shortcut) models in order to solve the problem efficiently. These models may either be self-contained or updated from the rigorous model. These techniques have been successfully applied to flowsheet simulation. One well known technique of this kind is termed the inside-out technique. For flowsheet simulation, this technique requires two iteration loops. The outer iteration loop deals with updating the simplified models and matching the simplified models to rigorous models, while flowsheet simulation is performed in the inner iteration loop. Applying this technique to flowsheet optimisation, Biegler *et al.* [1985 (a)] showed that it can lead to the detection of false optima or convergence failure. This situation arises from the fact that the gradients of the simplified models may not be the same as the gradients of the rigorous model. It is, therefore, not expected that they should exhibit the similar behaviour. This implies that not only the residuals of the simplified models but also the gradient information from the simplified model must match those of rigorous models for an optimal solution. Furthermore, in order to match the gradients at all the iterative points, it is necessary to evaluate the rigorous models as often as the simplified models is evaluated at the optimisation level. Clearly, this concept is not helpful.

Evans *et al.* [1985] suggested using nonlinear simplified model based on the rigorous model. Kisala *et al.* [1987] suggested guidelines for the construction of nonlinear simplified models. They constructed a simplified model in such a way that it is simple to evaluate and no iterative procedures are involved. The parameters of the model are obtainable from the results of the rigorous model by an analytical expression. Gradients associated with the model are generated analytically and are the same as that of the rigorous model. These guidelines suggest that the simplified model should be built in terms of the physical properties of the rigorous model. Obviously these nonlinear simplified models may be as difficult to construct as the rigorous model depending on the optimisation problems. In general, the gradients generated from the nonlinear simplified model may not be the same as those from the rigorous models and hence the optimal solution may be different. In this case, the authors suggested using linear simplified model, whose gradients are constructed from the rigorous model, to locate the optimal solution. In their implementation, the optimal solution from the linear simplified models also serves as a starting point for the flowsheet optimisation using nonlinear simplified models.

Chen and Stadtherr [1985 (b)] reported the performance of the optimisation is only slightly affected when approximate derivatives of physical property models are generated from the simplified model.

Ganesh and Biegler [1987 (a)] proposed a rigorous-simplified algorithm for optimisation based on simplified models. In their approach, the residuals are evaluated from the rigorous models and the gradients are evaluated from the simplified model at every optimisation step, unless the search directions are not descent directions for an absolute value penalty merit function. In the latter case, the gradients from the rigorous model are evaluated. A large value of the penalty parameter is used to force the iterates to the feasible region, and the line search is based on the rigorous models. The authors reported a significant reduction in the computational time when this technique is used. The drawback is that the value of the penalty function based on the rigorous model may not be reduced along the search direction even though the search direction is the predicted descent direction of the penalty function. This is because the gradients are based on the simplified models. In this case a rigorous model should be used. A line search failure is often observed from inaccurate gradients in my experience. The gradients evaluated from the simplified model at the solution point may not be the same as the gradients from rigorous models at the optimal point. Therefore, it becomes necessary to use the rigorous model to verify the solution obtained by this technique. In addition, a large penalty parameter may result in a line search difficulty, as mentioned in Chapter 2. This technique is potentially useful for problems where the gradients of the rigorous models are expensive to evaluate, and it becomes unattractive when analytical procedure derivatives for the rigorous models are available.

5.6 Sensitivity to Model Variations

Usually, there are many alternatives available for a model of the same process. Engineers would like to know the sensitivity of the objective function and of the solution vector under the variation of the process models or the thermodynamic models. In fact, this problem is equivalent to solving the following mixed integer nonlinear programming problem

$$\begin{aligned}
& \min_{x,b} f(x,b) \\
& s.t. \quad z'_i h_i(x,b) = 0 \quad i = 1, \dots, meq \\
& \quad \quad z''_j g_j(x,b) \geq 0 \quad j = 1, \dots, mi \\
& \quad \quad b_k - \sum_{l=1}^{nm} z_{lk} \Phi_{lk}(x) = 0 \quad k = 1, \dots, mp \\
& \quad \quad \sum_{l=1}^{nm} z_{lk} = 1 \quad k = 1, \dots, nb
\end{aligned} \tag{5.22}$$

P E.III

where

$\Phi_{lk}(x)$ is an alternative model l that defines the parameter b_k .
 nm is the number of alternatives for each parameter b_k .
 z, z', z'' are integer parameters with the value either 0 or 1.

It is interesting to determine the changes of the objective function and of the optimal solution with respect to changes of the set of integer parameters, z, z', z'' . Here, interest is limited to the interchange of similar models. It may not be appropriate to perform a sensitivity analysis when a reactor is replaced by a distillation column.

Ganesh and Biegler [1987 (b)] suggested that the change of the optimum be predicted from solving the following QP problem :

$$\begin{aligned}
& \min_x \nabla_x f(x) \Delta x + \Delta x \nabla_{xx}^2 L(x, \lambda, \nu) \Delta x \\
& s.t \quad h(x) + \nabla_x h(x)^T \Delta x = 0 \\
& \quad \quad g(x) + \nabla_x g(x)^T \Delta x \geq 0
\end{aligned} \tag{5.23}$$

P E.IV

where the objective function and constraints are evaluated from the new model. The Hessian of the Lagrangian function in this QP subproblem is based on both the curvature information of the new model together with the Lagrangian multipliers from the old model, and is obtained from perturbing only the independent variables. The solution Δx thus gives the predicted changes of the solution, caused by the changes of the model.

Since the active constraints may change under model variations, the authors suggested updating the Hessian of the Lagrangian function by using the Lagrangian multipliers predicted from equation (5.23), and resolving the quadratic programming problem until the estimated Lagrangian multipliers are all positive.

In some cases, the authors admitted that the formulation $P E.IV$ may perform poorly and they [Ganesh and Biegler, 1987 (b)] proposed a mixed integer nonlinear program ($MINLP$) to select the active constraints.

In fact, their approach simply takes a Newton step in the space of the new model from the optimal solution of the original model. This prediction may not be meaningful unless the two models resemble each other and the solution obtained from both models are quite close. The projected Hessian of the Lagrangian function based on the new models may not be positive definite, and hence the problem $P E.IV$ may not have a unique solution. The non-negativity of the Lagrangian multipliers does not ensure the prediction of the correct active set since the constraints may not be satisfied. Therefore, the prediction of the optimal solution is suspicious. In fact, the active constraints can be selected so that the projected Hessian on the active constraints is positive definite when the estimated solution is near the optimum of the optimisation problem based on the new model. In addition, the $MINLP$ formulation involves the calculation of the Hessian of the Lagrangian function and it gives rise to a combinatorial problem. It is expected that the effort of solving the mixed integer nonlinear programming problem be larger than the effort of solving the original optimisation problem with warm start procedures.

Therefore it is suggested to use a warm start procedure to solve this problem with model variations.

5.7 Numerical Experiences

Three chemical processes are used to test these techniques, namely an alkylation process, the Williams-Otto process and a simple refrigeration process.

5.7.1 Optimisation of the Alkylation Process

The process shown in Figure 3.1 is to produce alkylate with the greatest profit. The detailed description of the process is given in Appendix C.1. The version proposed by Sauer *et al.* [1964] and the standard starting point are used.

We use this example to show the effects of a warm start procedure on the SQP methods since we will use this procedure to solve the parametric optimisation problems. The SQP method took 26 gradient evaluations and 26 function evaluations to locate the optimal point with an identity matrix as the initial Hessian and the exact penalty function with the watchdog technique as merit function. However, it

only took 9 iterations and 9 function evaluations to reach the same optimal point when the correct set of active constraints was provided and the same condition were used. This result shows that the efficiency of the *SQP* can be improved when the correct set of active constraints is predicted. This result is important when a set of optimisation problems are to be solved for perturbed parameters.

5.7.2 Sensitivity Analysis of the Williams — Otto Process

The well known Williams-Otto process serves the second test problem for the parametric analysis. The reason of choosing this test problem is that this process has been studied by many researches, as listed in Table 5.1, and different results were obtained (Appendix C.2).

Table 5.1 Reviews of Optimisation on the Williams-Otto Process

Author(s)	Year
Williams and Otto	1960
Dibella and Stevens	1965
Ahlgren and Stevens	1966
Christensen	1970
Jung, Mirosh, and Ray	1971
Adelman and Stevens	1972
Luss and Jaakola	1973
Luss and Jaakola	1974
Sriram and Stevens	1973
Rijckaert and Martens	1974
Findley	1974
Wang and Luus	1978
Vinante and Valladares	1985
Rangaiah	1985
Ostrovsky, Ostrovsky, and Berezhinsky	1988

This process, shown in Figure 3.2 is to manufacture 4763 *lbs/hr* of pure product P_w with maximum investment return. The system involves a cooler, a decanter, and a distillation column. The pure reactants, A_w and B_w are fed separately to the reactor and the pure product P_w , intermediates C_w , and the by-product E_w and G_w are produced. The reactions are assumed to be second order and

irreversible. The undesirable byproduct G_w is discarded from the decanter. Part of the product P_w forms an azeotrope with the byproduct E_w in the distillation column.

It is interesting to study the parameter sensitivity of the objective function to the reactor volume V_r , the amount of the product, F_p and the weight fraction, R_f , of product P_w removed with the byproduct, E_w in the distillation column. The Lagrangian multipliers corresponding to these design constraints, listed in the Table 5.2, indicate the sensitivity of the objective function to these parameters.

Table 5.2 The Lagrangian Multipliers to the Design Constraints

Design constraints	Lagrangian multipliers
$V_r - 60. = 0$	0.8697
$R_f - 0.1 = 0$	2567.008
$F_p - 4763 = 0$	-0.01094

According to this analysis, the factor R_f has a significant effect on the objective function. In order to assess the validity of this sensitivity analysis, this process was solved for various values of parameters with a warm start procedure. Table 5.3 gives a comparison of the optimal values of the objective function, Obj_t , calculated from the solution of the parametric optimisation with the optimal values, Obj_p , predicted from equation (5.15). This table also shows the effect of the warm start procedure. These runs are performed in sequence so that the information obtained from run I initialise the calculations for run II and so on.

The warm start procedure reduces the number of gradient and function evaluations.

At the base case, the projected Hessian of the Lagrangian function on the active constraints is

$$\begin{pmatrix} 4.735 \times 10^{-8} & -2.786 \times 10^{-8} & 2.131 \times 10^{-9} \\ -2.786 \times 10^{-8} & 1.747 \times 10^{-8} & -3.926 \times 10^{-9} \\ 2.131 \times 10^{-9} & -3.926 \times 10^{-9} & 2.508 \times 10^{-8} \end{pmatrix}$$

and the unscaled 2-norm of the projected gradient of the objective function is 1.1×10^{-6} .

The subroutine *RSG* of the *EISPACK* package [Garbow *et al.*, 1977], was used to calculate the following eigenvalues of the projected Hessian,

$$(9.66 \times 10^{-9}, 2.47 \times 10^{-8}, 5.55 \times 10^{-8}).$$

The tiny eigenvalues indicates that the objective function is very flat near the solution and can explain the disagreement among the results obtained from many researchers.

Table 5.3 Parametric Optimisation of the Williams-Otto Process

Parameters	run I	run II	run III	run IV	run V
V_R	60	50	60	60	57
R_f	0.1	0.1	0.105	0.1	0.1001
F_P	4763	4763	4763	5763	4793
$(NG)_c$	70	69	70	71	69
$(NG)_w$	-	32	33	49	41
$(NF)_c$	70	72	71	73	70
$(NF)_w$	-	34	36	49	43
Obj_p	-	107.375	85.828	109.621	101.358
Obj_t	98.678	107.756	85.631	108.146	101.302
error (%)	-	0.35	0.23	1.3	0.05

where

- $(NF)_c$ Number of function evaluations obtained without using warm start procedure.
- $(NF)_w$ Number of gradient evaluations obtained by using the warm start procedure.
Algorithm E.I.
- $(NG)_c$ Number of gradient evaluations obtained without using starting procedure.
- $(NG)_w$ Number of gradient evaluations obtained using the warm start procedure.
Algorithm E.I.

5.7.3 Sensitivity Analysis of the Refrigeration Process

The third test problem, given by Rudd and Watson [1968] and studied by Hutchison, Jackson, and Morton [1986] is a refrigeration process. The process, shown in Figure 5.1 consists of three exchangers. It is assumed that the overall heat coefficient (U_h), the specific heats (CP) of the hot stream and the latent heat (H_r) of refrigerant are constant, and the logarithmic mean temperature difference can be applied to this system. The objective of this process is to minimise the capital cost and the operating cost. The design specifications of the process are listed in Table 5.4.

Figure 5.1 Simple Refrigeration Process Flowsheet

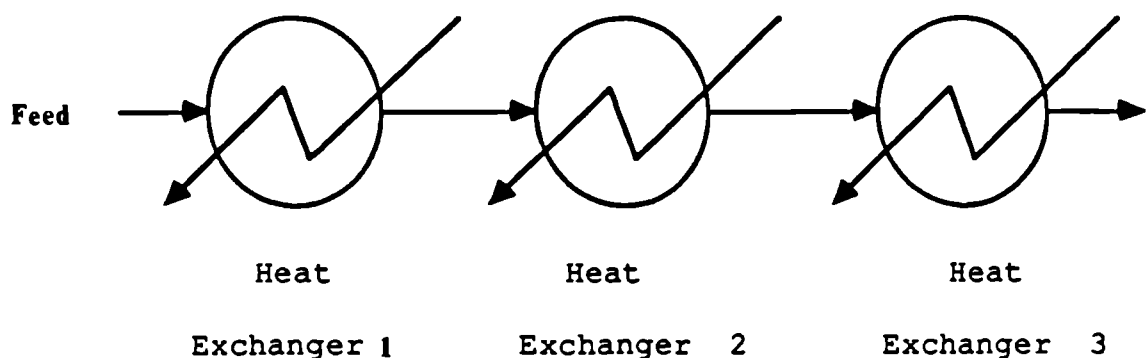


Table 5.4 Specifications for a Simple Refrigeration Process Flowsheet

Heat transfer area of heat exchanger i , ft^2	A_i			
Outlet temperature of heat exchanger i , ($^{\circ}F$)	T_o			
Feed Flow 1000 $\frac{lb}{hr}$ temperature $50^{\circ}F$	specific heat $1.0 \frac{Btu}{lb \cdot ^{\circ}F}$			
T_{o3} -70	Refrigerant latent heat ($\frac{Btu}{lb \cdot ^{\circ}F}$)	100		
Heat Exchanger	1	2	3	
Refrigerant temperature ($^{\circ}F$)	0	-40	-80	
Overall heat transfer coefficient ($\frac{Btu}{hr \cdot ft^2 \cdot ^{\circ}F}$)	200	200	200	
Capital cost constant ($\frac{\$}{hr \cdot ft^2}$)	0.05	0.05	0.15	
Operating cost constant ($\frac{\$}{lb}$)	2×10^{-4}	3×10^{-4}	4×10^{-4}	
Objective function (minimize) $\sum_{i=1}^3 a_i (A_i)^{0.5} + b_i (\text{refrigerant flow})$				

Table 5.5 The Lagrangian Multipliers Corresponding to to the Design Constraints.

Design constraints	Lagrangian multipliers
$C_p - 1. = 0$	4.2854
$U_h - 200 = 0$	-0.003294
$H_r - 100. = 0$	-0.036267
$T_{s1} - 50. = 0$	0.02292
$T_{s3} + 70. = 0$	-0.05382
$Tr_1 = 0$	0.0099
$Tr_2 + 40 = 0$	0.0099
$Tr_3 + 80 = 0$	0.0111

Numerical results obtained by Huchison *et al.* [1986 (b)] showed that the optimisation algorithm may converge to three different solutions from three different starting points. Two of them have a similar objective function (0.01% difference). Thus it is interesting to study the design variable sensitivity of this process. We also study the parametric sensitivity of the objective function to the specific heat of refrigerant (C_p), the overall heat coefficient (U_h), the latent heat of refrigerant (H_r), the inlet temperature of the hot stream in the heat exchanger 1 (T_{s1}), the exit temperature of the hot stream (T_{s3}) and the refrigerant temperatures of three heat exchangers (Tr_1, Tr_2, Tr_3).

The Lagrangian multipliers of the corresponding design constraints, listed in Table 5.5, show the sensitivity of the objective function to these parameters.

In this process, the specific heat of the hot stream is a dominant factor for the objective function.

Table 5.6 compares the optima of the objective function Obj_t from the solution of this process with the values of the optima Obj_p predicted from equation (5.15).

Table 5.6 Parametric Optimisation on a Refrigeration Process

Parameters	Base case	Run I	Run II	Run III
CP	1.0	1.05	1.0	1.0
U_h	200	200	200	200
H_e	100	100	100	100
T_{s1}	50	50	50	50
T_{s3}	-70	-70	-75	-75
Tr_1	0	0	0	0
Tr_2	-40	-40	-40	-45
Tr_3	-80	-80	-85	-85
$(NF)_o$	6	7	7	9
$(NF)_w$	-	2	5	6
$(NG)_o$	6	7	7	9
$(NG)_w$	-	2	5	6
Obj_p	-	5.1583	5.2647	5.2152
Obj_t	4.9441	5.1579	5.1567	5.1071
error (%)	-	0.001	1.8	2.12

This process has two degrees of the freedom, and the outlet temperature of the hot stream from heat exchanger 1 (To_1) and that from heat exchanger 2 (To_3) are chosen as design variables for studying the decision variable sensitivity.

The unscaled 2-norm of the reduced gradient of the objective function at the base case is $2. \times 10^{-6}$ and the reduced Hessian of the Lagrangian function is

$$\begin{pmatrix} 6.7675 \times 10^{-4} & 8.1912 \times 10^{-5} \\ 8.1912 \times 10^{-5} & 6.8089 \times 10^{-4} \end{pmatrix}$$

The subroutine *SVD* of the *EISPACK* package [Garbow *et al.*, 1977], which implements a singular value decomposition algorithm, is used to calculate the eigenvalues and the corresponding eigenvectors.

The eigenvalues of this reduced Hessian are

$$(5.968 \times 10^{-4}, 7.6708 \times 10^{-4})$$

and the corresponding eigenvectors are

$$(-0.716, -0.6981) \text{ and } (-0.6981, -0.716)$$

respectively.

Both design variables have the same order of magnitude and hence have similar effect on the objective function. The small eigenvalues indicate that the objective function is not sensitive to them near the optimal point. The minimum of this problem is flat. However, this can be improved by scaling the objective function. A scale factor of 10^4 on the objective function is recommended for this problem.

5.8 Discussion and Conclusions

This chapter showed that the sensitivity of the objective function with respect to both the parameter variations and the design variables can be obtained from the solution of the *SQP* method. The analysis of the eigenvalue system of the Lagrangian function is helpful for understanding the optimisation problem. The warm start procedure increases the efficiency of the *SQP* method.

A warm start procedure is also recommended for solving the optimisation of process with model variations.

Chapter 6

Conclusions

6.1 Summary

The main objective of this research was to develop an effective *SQP* method for flowsheet optimisation and test it on problems of industrial significance.

Optimisation methods for small scale and for large scale optimisation are examined in Chapters 2 and 3 respectively. Various flowsheet optimisation strategies are investigated in Chapter 4. Numerical results in Chapter 4 demonstrate the capability of the algorithms and show that flowsheet optimisation can be performed in a reliable and effective way. In addition, Chapter 5 shows that extra valuable information can be obtained from the solution of the flowsheet optimisation problems.

The results of this research are summarised as follows.

It is useful to divide the search direction of *SQP* methods into two directions : one towards the linearised model of the nonlinear constraints, and the other towards the minimum of the quadratic model of the Lagrangian function in the subspace orthogonal to the constraint normals. The former step can hence be referred to as the 'feasibility step', and the latter as the 'optimality step'. Each step can be generated from the solution of a corresponding *QP* problem for large scale optimisation problems. The advantages of this formulation are that it is straightforward to include inequality constraints and step bounds in either the feasibility step or the optimality step. It is possible to perform feasibility improvement for selected constraints of interest. It is also possible to introduce into the quadratic program for the feasibility step an inequality constraint which ensures the descent direction of the Lagrangian function. For example, the following constraints can be included,

$$\nabla_x L(x, \lambda, \nu)^T p_s \leq 0 \quad (6.1)$$

where $L(x, \lambda, \nu)$ is the Lagrangian function, λ and ν are Lagrangian multipliers, and p_s is the feasibility step. It is relatively easy to provide scale factors for both optimisation and feasibility steps.

Based on this formulation, various algorithms can be derived. When feasible points are obtained by repeatedly solving the corresponding quadratic program in conjunction with line search strategies at each iteration, the resulting method corresponds to established feasible direction methods (e.g. projection methods). This feature is an option in the current implementation. Numerical results in Chapter 2 show that approximate gradients (either constant or from a quasi-Newton update) of the constraints can be used for the feasibility step. This feasibility improvement also allows the algorithm to terminate at a feasible point whenever it is possible, which has its own practical value. The usefulness of the feasibility improvement depends on the convexity of the objective function and nonlinearity of the constraints. It is preferable to satisfy constraints more closely for problems with an unbounded objective function and highly curved ' hard ' constraints.

Numerical results show that a proper choice of the initial Hessian can dramatically improve the performance of the SQP methods. The proposed strategy that only uses the physical magnitude of the variables is proved to be effective in practice and it is simple enough to be implemented. The assumption behind this strategy is that an estimate of the order of magnitude of the variables is available from a reasonable starting point . A reasonable initial starting point can be obtained from a partial or complete flowsheet simulation. Based on our experience, a scaling strategy based on the gradient of the objective function is not recommended for the sequential modular approach, since the performance may be degraded by the noise in the gradients.

Strategies that use penalty function methods have been proved useful to deal with constraint inconsistency. This formulation allows to deal as well with linearly dependent constraints.

It is desirable to impose step bounds (trust region bounds) in the QP subproblem since this merely means that the quadratic model is valid within the range of interest. The range of the step bound depends on the type of the objective function and on the nonlinearity of constraints. A small range for the step bound is preferred with an unbounded objective function and highly nonlinear constraints. Step bounds are automatically adjusted in the proposed trust region method. Step bounds chosen as the order of magnitude of the variables work well in practice for the line search methods. In this study, step bounds are also constructed such that all the variables satisfy their corresponding simple bounds since all the simple bounds are considered to be ' hard ' constraints and ' nonlinear hard

constraints' can be converted to equality constraints and simple bounds through extra variables. This improves the robustness of the optimisation algorithm in practice.

Both the absolute value penalty function with safeguards and the augmented Lagrangian function can serve as merit functions in the line search and trust region method. Excessively large penalty parameters with the absolute value penalty function may degrade the performance of the *SQP* methods especially when the iterates are near the feasible region. In practice, the watchdog technique or the relaxation method presented in Chapter 4 can be used to reduce the risk of slow convergence. A feasibility step can also be employed for this purpose. The use of the augmented Lagrangian function as a merit function may suffer from insufficiently large penalty parameters near the solution. A proper choice of these penalty parameters requires some curvature information of the objective function and constraints. The trust region method can also be used to force convergence. Numerical results showed that the performance of trust region methods with feasibility steps is comparable to that of line search methods.

Updating the reduced Hessian of the Lagrangian function results in a method that depends on the basis of the reduced space and may lead to irregular local convergence. An automatic scheme that chooses the independent variables by decomposing a rectangular Jacobian matrix of the constraints is found to be effective in practice. It is also helpful to choose linear variables as dependent variables, where possible, for the large scale system. The feasibility step is found to be useful in the case of irregular local convergence.

With sequential modular approaches, it is helpful to initialise the optimisation calculation by a few direct substitutions. It is also desirable to use a complete or partial simulation for initialising all the variables with equation oriented approaches.

Numerical results on a variety of problems show that the optimisation of chemical processes using an industrial sequential modular simulator requires the equivalent time of 10 to 30 simulations which is comparable to those in the literature. Chemical process problems are found to be much more demanding than the standard mathematical problems used in testing optimisation algorithms.

One of the merits of the *SQP* method is that the sensitivity of the objective function to its parameters can be generated without too much effort.

It is necessary to use rigorous models to verify the optimal solution when simplified models are used for flowsheet optimisation.

Numerical results also show that process optimisation problems can be solved on both modular oriented and equation oriented simulators with relatively equal success.

It is hoped that this research provides a better understanding of optimisation methods and will be beneficial in the further development of optimisation techniques for process synthesis, process design under uncertainty, dynamic optimisation and multiple objective optimisation.

6.2 Contributions

In summary, the main contributions of this work are

1. Based on the property of the *SQP* algorithm, a general representation for constrained optimisation has been proposed. This representation decomposes the search direction of the constrained optimisation into two steps : a feasibility step towards the feasible region of the constraints, and an optimisation step towards the optimal point of the objective function.
2. The proposed representation has proved useful for developing a class of large scale optimisation algorithm based on the projected Hessian.
3. A strategy has been suggested for dealing with a system of the equations with inequality constraints.
4. Strategies for dealing with inconsistent constraints have been suggested and have been proved useful.
5. A class of augmented Lagrangian functions as line search objective functions has been suggested and proved useful for forcing convergence.
6. A trust region method for constrained optimisation has been tested and its effectiveness has been shown.
7. An effective implementation of a parametric sensitivity method has been established.

8. Systematic testing of the *SQP* algorithm has been performed. It has proved useful in the algorithmic development of effective optimisation methods and results in a robust and efficient optimisation code.
9. The advantages of flowsheet optimisation have been demonstrated on two types of commercial simulators on optimisation problems of industrial significance.

6.3 Future Research

The results of this research expose some issues and potential areas for further investigation and development in the field of optimisation methods and flowsheet optimisation.

The *SQP* method is based on the Lagrangian system and hence it is assumed that all functions are differentiable and that the constraints are linearly independent. The relaxation of these assumptions on the optimality conditions may be worthy of further research. Recently attention has been paid to the development of nondifferentiable optimisation [Polak, Mayne and Ward, 1983], [Yuan, 1985 (b)], [Fletcher, 1988]. The application of the *SQP* methods to non-differentiable optimisation was also proposed [Bell, 1984]. Application of nondifferentiable optimisation in engineering design has been recently reviewed by Polak [1987]. The evaluation and application of these techniques require further investigation.

The performance of the *SQP* method strongly depends on the efficiency and robustness of the underlying quadratic program. The efficiency of solving a quadratic program with mixed equality and inequality constraints depends on the active set strategy. The dual active set algorithm of Goldfarb and Idnani [1983] is effective when few extra constraints are to be added to the active set, since it does not require an initial feasible point. However, this method is restricted to strictly convex quadratic programs. In contrast, the primal active set algorithm is capable of solving semi-definite and indefinite quadratic programs, but it requires an initial feasible point. It is thus desirable to have a quadratic program algorithm which allows primal and dual methods. This feature is useful when it is required to solve a sequence of problems in which either the set of active constraints or the right hand side of the constraints is varied. These situations can require a step to restore both feasibility and optimality, and may occur in the application of the warm start procedures for the *SQP* algorithm. Goldfarb and Idnani [1982] presented such an algorithm for solving strictly convex quadratic programs. Further study is required on the indefinite quadratic program. Goldfarb [1986] suggested a condition that

allows several constraints to be dropped simultaneously. Unfortunately, it may require a substantial amount of work to check this condition. This feature is useful for large scale quadratic programs and deserves further investigation.

Iterative methods (e.g. successive over-relaxation (SOR) methods) may be feasible for large convex quadratic programs and they are reviewed by Lin and Pang [1987]. The advantage of iterative methods is that they may require less storage. The computational effort may not depend strongly on the size of the problem but it is governed by the convergence rate. The numerical stability and convergence properties require further research since these methods are strongly dependent on the eigenvalue system of the Lagrangian function.

Only the projected Hessian of the Lagrangian function needs to be positive definite at the solution. Keeping the full Hessian positive definite may result in ill-conditioning [Powell, 1985 (b)]. The use of a positive approximation to the Hessian of the augmented Lagrangian function requires an arbitrary penalty parameters [Tapia, 1977]. Another possibility is to recur a positive definite approximation to the projected Hessian of the Lagrangian function. Numerical results showed that the decomposition method that uses the reduced Hessian is effective for problems with relatively few independent variables. When there are many independent variables, a Hessian update scheme that preserves the symmetry, sparsity, positive definiteness on the active constraints and good convergence properties is required. When the Hessian of the Lagrangian function is not positive definite, a trust region method should be used. However, the trust region bound should be inactive eventually in order to obtain a fast convergence rate. This implies that the approximate Hessian is positive definite on the space of the active constraints when the iterates are near an optimum. A procedure to ensure this condition should be developed.

Both the line search methods and trust region methods require either arbitrary parameters or heuristic rules to improve and achieve global convergence. The difficulty arises from the need to compromise between the ' optimality ' and ' feasibility ' and from the need to construct a convex merit function that retain the information of the objective function and the constraints. Further work may be directed towards relaxing these conditions in order to increase the robustness of the optimisation algorithm.

Calculation of modules within the flowsheet simulators may involve iterative procedures and hence the functions may be subject to noise. In this study, it was

attempted to minimise the level of function noise. On the other hand, it may be worth developing an optimisation algorithm which directly deals with optimisation with function noise. The research along this line includes work of Poljak [1978], Ruszczyński [1984] and Pai and Hughes [1987].

The comparison of various optimisation algorithms is largely affected by implementation detail and by the characteristics of test problems. It was shown that an optimisation algorithm may be partitioned into certain fundamental steps. A standard definition for these steps and a standard implementation of these steps are required.

This work showed that the exploitation of linearity in the problem can profoundly improve the performance of the optimisation algorithms and that optimisation problems with a flat minimum may cause difficulty to an optimisation algorithm. It is thus necessary to classify the test problems and construct a set of process test problems with various characteristics. Bartels and Mahdavi-Amiri [1986] shed some light on this problem and further investigation can be beneficial.

It is possible to incorporate other criteria into flowsheet optimisation in order to reach an optimal condition that makes the process operable, controllable, reliable, and flexible to deal with unexpected problems. This suggests a fruitful research area for further development in flowsheet optimisation with multiple objective functions.

Finally, new advances in computer architectures and parallel processing offer new opportunity for algorithm and software developments. Further work in the application of parallel computations to both flowsheeting and optimisation should lead to both improved performance and more efficient optimisation algorithms and flowsheet architectures.

Appendix B.1

This appendix lists the characteristics of the test problems used in this thesis and the performance of the major optimisation codes on these test problems. The characteristics of the mathematical test problems are listed in Table B.1 and those of the chemical process problems are in Tables B.2 and B.3.

Table B.4 lists the performance of three optimisation codes, namely *OPXRQP*, modified *VMCDW* and *NLPQL*, on the mathematical test problems.

Table B.1 Characteristics of the Mathematical Test problems

Problem No.	N	MEQ	M	NB	OPT	Source
M1	2	0	0	1	0	SH(1)
M2	2	0	0	1	0.05	SH(2)
M3	2	0	0	1	0	SH(3)
M4	2	0	0	2	2.667	SH(4)
M5	2	0	0	4	-1.913	SH(5)
M6	2	2	1	0	0.	SH(6)
M7	2	1	0	0	-1.731	SH(7)
M8	2	2	0	0	-1.	SH(8)
M9	2	1	0	0	-0.5	SH(9)
M10	2	0	0	0	-1.	SH(10)
M11	2	0	0	0	-8.49	SH(11)
M12	2	0	1	0	-30.	SH(12)
M13	2	0	1	2	1.	SH(13)
M14	2	1	1	0	1.39	SH(14), HM(1)
M15	2	0	2	1	306.5	SH(15)

Table B.1 Characteristics of the Mathematical Test Problems (Cont'd)

Problem No.	N	MEQ	M	NB	OPT	Source
M16	2	0	2	3	0.25	SH(16)
M17	2	0	2	3	1.	SH(17)
M18	2	0	2	4	5.	SH(18)
M19	2	0	2	4	-6961.81	SH(19)
M20	2	0	3	2	38.199	SH(20)
M21	2	0	1	4	-99.96	SH(21)
M22	2	0	2	0	1.	SH(22), HM(24)
M23	2	0	5	4	2.	SH(23)
M24	2	0	3	2	-1.	SH(24)
M25	2	0	1	2	0.0284	SH(57)
M26	2	0	3	4	-7.804	SH(59)
M27	3	0	0	6	0.	SH(25), HM(21)
M28	3	1	0	0	0.	SH(26)
M29	3	1	0	0	0.04	SH(27)
M30	3	1	0	0	0.	SH(28)
M31	3	0	0	0	-22.627	SH(29)
M32	3	0	1	6	1.	SH(30)
M33	3	0	1	6	6.	SH(31)
M34	3	1	1	3	1.	SH(32)
M35	3	0	2	4	-4.485	SH(33)
M36	3	0	2	6	0.83	SH(34)
M37	3	0	1	3	0.11	SH(35)
M38	3	0	1	6	-3300.	SH(36)
M39	3	0	2	6	-3456.	SH(37)
M40	3	1	0	6	0.0325	SH(60)

Table B.1 Characteristics of the Mathematical Test Problems (Cont'd)

Problem No.	N	MEQ	M	NB	OPT	Source
M41	3	2	0	0	-143.64	SH(61)
M42	3	1	0	6	-26272.52	SH(62)
M43	3	2	0	3	961.715	SH(63), HM(5)
M44	3	0	1	3	6299.84	SH(64)
M45	3	0	1	6	0.9535	SH(65)
M46	3	0	2	6	0.518	SH(66)
M47	3	0	14	6	-1162.04	SH(67), CV(8)
M48	4	0	0	8	0.	SH(38), CV(4)
M49	4	2	0	0	-1.	SH(39)
M50	4	3	0	0	-0.25	SH(40)
M51	4	1	0	8	1.9259	SH(41)
M52	4	2	0	0	13.858	SH(42)
M53	4	0	3	0	-44.	SH(43)
M54	4	0	6	4	-15.	SH(44)
M55	4	0	1	8	0.0075	SH(70), HM(9)
M56	4	1	1	8	17.014	SH(71)
M57	4	0	2	8	727.679	SH(72)
M58	4	1	2	4	29.894	SH(73)
M59	4	3	2	8	5126.498	SH(74)
M60	4	3	2	8	5174.413	SH(75)

Table B.1 Characteristics of the Mathematical Test Problems (Cont'd)

Problem No.	N	MEQ	M	NB	OPT	Source
M61	4	0	3	4	-4.6818	SH(76)
M62	5	0	0	10	1.	SH(45)
M63	5	2	0	0	0.	SH(46)
M64	5	3	0	0	0.	SH(47)
M65	5	2	0	0	0.	SH(48)
M66	5	2	0	0	0.	SH(49)
M67	5	3	0	0	0.	SH(50)
M68	5	3	0	0	0.	SH(51)
M69	5	3	0	0	5.326	SH(52)
M70	5	3	0	10	4.093	SH(53)
M71	5	2	0	0	0.241	SH(77)
M72	5	3	0	0	-2.919	SH(78)
M73	5	3	0	0	0.0787	SH(79)
M74	5	3	0	10	0.0539	SH(80)
M75	5	3	0	10	0.0539	SH(81)
M76	5	0	3	10	-30665.5	SH(83), CV(3)
M77	5	0	3	10	-5.28E6	SH(84)
M78	5	0	38	10	-1.905	SH(85), HM(12)
M79	5	0	10	5	-32.34	SH(86), CV(1)
M80	6	1	0	12	0.908	SH(54)

Table B.1 Characteristics of the Mathematical Test Problems (Cont'd)

Problem No.	N	MEQ	M	NB	OPT	Source
M81	6	6	0	12	6.333	SH(55)
M82	6	4	0	12	8927.59	SH(87),CV(6)
M83	6	0	2	6	135.07	SH(93)
M84	6	0	4	12	0.1562	SH(95), HM(22)
M85	6	0	4	12	0.1562	SH(96)
M86	6	0	4	12	3.1358	SH(97)
M87	6	0	4	12	3.1358	SH(98)
M88	7	4	0	0	-3.456	SH(56)
M89	7	2	0	14	-8.311E10	SH(99)
M90	7	0	4	0	680.63	SH(100)
M91	7	0	6	14	1809.764	SH(101)
M92	7	0	6	14	911.88	SH(102)
M93	7	0	6	14	543.667	SH(103)
M94	8	0	6	16	3.951	SH(104), DM(4)
M95	8	0	1	16	1138.416	SH(105)
M96	8	0	6	16	7049.33	SH(106), DM(5)
M97	9	6	0	8	5055.012	SH(107)
M98	9	0	13	1	-0.866	SH(108), HM(16)
M99	9	4	6	16	5362.07	SH(109)
M100	10	0	0	20	-45.778	SH(110),HM(17)

Table B.1 Characteristics of the Mathematical Test Problems (Cont'd)

Problem No.	N	MEQ	M	NB	OPT	Source
M101	10	3	0	20	-47.761	SH(111), HM(4a)
M102	10	3	0	10	-47.707	SH(112) HM(4)
M103	10	0	8	0	24.306	SH(113)
M104	15	0	5	15	32.348	SH(117), CV(2)
M105	15	0	29	30	664.82	SH(118)
M106	15	0	10	0	-8252.0	CO(15)
M107	15	0	10	0	-8310.26	CO(16)
M108	15	0	10	0	-8315.28	CO(17)
M109	15	0	11	0	-8250.14	CO(18)
M110	15	0	11	0	-8164.37	CO(19)
M111	15	0	15	0	-5819.92	CO(20)
M112	15	0	15	0	-5809.712	CO(21)
M113	16	0	8	32	244.899	SH(119), CV(7)
M114	24	14	6	24	38.198	HM(20)

Key

- N** total number of variables.
MEQ total number of equality constraints.
M total number of constraints except simple bounds.
NB total number of simple bounds.
OPT optimal value of the objective function.
CV (i) test problem i from Colville [1968].
CO (i) test problem i from Cornwell *et al.* [1978].
DM (i) test problem i from Dembo [1976].
SH (i) test problem i from Hock and Schittkowski [1981].

Table B.2 Descriptions of Chemical Processes Test problems

Problem No.	Processes	Source
C1.	Through-circulation drier	Luus and Jaakola [1973]
C2.	Fluidised reactor-heater system	Goulcher and Long [1978]
C3	Batch process	Goulcher and Long [1978]
C4	Fuel oil and BFG in Power generation	Goulcher and Long [1978]
C5	Waste-heat boiler train	Doherty <i>et al.</i> [1982]
C6	Double effect evaporator	Newell [1980]
C7	Alkylation process (a)	Bracken and McCormick [1968]
C8	CSTR train	Rangaiah [1985]
C9	Alkylation process (b)	Sauer <i>et al.</i> [1964]
C10	Cross-current extraction process with product recycle	Luus [1975]
C11	Williams-Otto process (a)	Williams and Otto [1960]
C12	Williams-Otto process (b)	Sriam and Stevens [1973]
C13	Williams-Otto process (c)	Jung <i>et al.</i> [1971]
C14	Three stage membrane separation	Dembo [1976]
C15	Cross-current extraction process with wash stream recycle	Luss [1975]
C16	Butane isomerisation process	Himmelblau [1963]
C17	Five stage membrane separation	Dembo [1976]

Table B.3 Characteristics of the Chemical Process Test Problems

Problem No.	N	MEQ	M	NB	OPT
C1	2	0	3	0	-172.48
C2	3	1	3	5	146435.
C3	4	1	1	4	126303.
C4	4	1	2	8	3.052
C5	4	2	2	4	-290415
C6	6	3	3	12	0.564
C7	10	3	11	20	-1768.8
C8	10	4	4	20	-0.549
C9	10	7	7	20	-1162.3
C10	11	8	19	8	-0.1015
C11	12	8	8	4	-98.68
C12	13	8	8	6	-121.5
C13	13	8	8	6	-50.82
C14	13	0	15	26	97.58
C15	15	10	10	15	-0.1084
C16	15	11	0	23	-918.99
C17	16	0	21	32	174.768

Table B.4 Comparison of Three Optimization Codes
on the Mathematical Test Problems

Problem No.	OPXRQP NG(NF)	VMCWD NG(NF)	NLPQL NG(NF)
M1	21(39)	17(32)	20(44) [*]
M2	15(15)	15(24)	14(20)
M3	16(48)	8(8)	10(10)
M4	16(16)	2(2)	2(2)
M5	8(10)	9(13)	6(8)
M6	14(14)	10(13)	9(10)
M7	16(17)	13(14)	10(11)
M8	6(8)	F2	5(5)
M9	14(19)	6(7)	6(6)
M10	17(17)	12(12)	12(13)
M11	16(16)	9(11)	9(10)
M12	16(24)	9(10)	8(9)
M13	30(30) [*]	51(51)	F3
M14	16(16)	6(6)	7(13)
M15	17(17)	5(6)	4(4)
M16	17(26)	6(6)	6(9)
M17	33(38)	12(16)	F3
M18	18(20)	8(8)	21(36)
M19	36(56)	7(7)	7(7)
M20	18(20)	F2	17(20)

Table B.4 Comparison of Three Optimization Codes
on the Mathematical Test Problems (Cont'd)

Problem No.	OPXRQP NG(NF)	VMCWD NG(NF)	NLPQL NG(NF)
M21	16(16)	4(4)	4(4)
M22	16(16)	6(7)	6(7)
M23	21(21)	7(7)	7(7)
M24	23(27)	5(5)	5(5)
M25	F4	2(4) ^d	3 5) ^d
M26	22(28) ^d	12(14) ^d	15 16) ^d
M27	1(1) ^d	1(1) ^d	1(1) ^d
M28	18(20)	15(16)	18(20)
M29	26(40)	35(64)	19(25)
M30	5(6)	4(5)	4(5)
M31	35(38)	14(15)	12(13)
M32	17(17)	14(14)	14(14)
M33	26(40)	15(16)	8(10)
M34	19(19)	3(3)	3(3)
M35	38(52)	5(5) ^e	5(5) ^e
M36	36(49)	8(8)	8(8)
M37	23(27)	7(7)	7(7)
M38	18(19)	3(3)	3(3)
M39	19(21)	9(11)	9(11)
M40	18(21)	7(9)	10(11)

Table B.4 Comparison of Three Optimization Codes
on the Mathematical Test Problems (Cont'd)

Problem No.	OPXRQP NG(NF)	VMCWD NG(NF)	NLPQL NG(NF)
M41	19(24)	10(12)	12(24)
M42	28(64)	8(14)	10(31)
M43	18(19)	11(11)	F3
M44	F6	37(42)	18(18)
M45	24(33)	8(8)	8(8)
M46	18(20)	7(7)	7(7)
M47	44(53) ^a	22(22)	21(21)
M48	300(4321) ^a	83(106)	82(113)
M49	20(20)	12(14)	12(14)
M50	19(21)	6(6)	6(6)
M51	15(15)	7(7)	7(7)
M52	17(17)	8(10)	8(10)
M53	19(23)	12(15)	11(14)
M54	21(21)	6(6)	6(6)
M55	52(76) ^a	33(34)	34(37)
M56	20(20)	5(5)	5(5)
M57	63(111)	29(33)	23(23)
M58	21(22)	4(4)	4(4)
M59	35(54)	10(10)	10(10)
M60	47(88)	9(9)	9(9)

Table B.4 Comparison of Three Optimization Codes
on the Mathematical Test Problems (Cont'd)

Problem No.	OPXRQP NG(NF)	VMCWD NG(NF)	NLPQL NG(NF)
M61	17(17)	6(6)	6(6)
M62	22(22)	8(8)	8(8)
M63	121(122)	12(14)	12(14)
M64	67(111)	19(23)	24(30)
M65	10(11)	8(9)	8(9)
M66	22(57) ^a	6(9)	6(9)
M67	14(21)	11(15)	12(15)
M68	8(8)	3(5)	3(5)
M69	16(18)	6(9)	6(8)
M70	15(15)	7(8)	7(8)
M71	19(26)	15(16)	15(16)
M72	18(19)	8(9)	8(8)
M73	16(16)	9(10)	9(10)
M74	14(14)	7(8)	7(7)
M75	14(14)	21(34)	8(8)
M76	20(20)	5(5)	5(5)
M77	18(43)	5(6)	46(66)
M78	57(85)	27(27)	57(92)
M79	24(30)	5(6)	5(6)
M80	3(3) ^d	2(2) ^d	2(2) ^d

Table B.4 Comparison of Three Optimization Codes
on the Mathematical Test Problems (Cont'd)

Problem No.	OPXRQP NG(NF)	VMCWD NG(NF)	NLPQL NG(NF)
M81	15(15)	2(2) ^d	2(2) ^d
M82	51(84) ^a	13(13)	16(20)
M83	22(29)	13(15)	11(16)
M84	29(31)	2(2)	2(2)
M85	27(29)	2(2)	2(2)
M86	40(46)	7(7)	7(7)
M87	27(33)	7(7)	7(7)
M88	18(21)	15(15)	13(15)
M89	F4	24(48) ^a	31(72)
M90	18(25)	14(20)	13(30)
M91	53(94)	45(72)	32(55)
M92	49(77)	65(104)	39(59)
M93	46(72)	43(72)	31(54)
M94	29(29)	17(18)	17(19)
M95	97(122) ^a	43(51) ^a	52(65) ^a
M96	F4	34(36)	234(636)
M97	30(40)	9(11)	13(23)
M98	30(107) ^a	19(19)	16(18)
M99	64(124)	F3	F3
M100	10(20)	8(11)	8(11)
M101	93(610)	62(81)	52(54)
M102	39(53)	20(41)	25(45)
M103	34(37)	12(16)	13(16)
M104	53(64)	16(16)	16(16)
M105	30(30)	18(18)	20(20)

Table B.4 Comparison of Three Optimization Codes
on the Mathematical Test Problems (Cont'd)

Problem No.	OPXRQP NG(NF)	VMCWD NG(NF)	NLPQL NG(NF)
M106	50(90)	48(74)	29(45)
M107	56(107)	51(82)	24(53)
M108	57(105)	45(70)	25(53)
M109	53(82)	54(84)	33(64)
M110	84(162)	62(99)	32(62)
M111	45(68)	37(54)	28(60)
M112	54(87)	36(52)	23(41)
M113	28(38)	10(13)	10(13)
M114	31(36)	14(14)	17(20)

Key

OPRQP	The OPRQP optimisation code is used.
VMCWD	The modified VMCWD optimisation code is used.
NLPQL	The NLPQL optimisation code is used.
NG	Number of gradient evaluations.
NF	Number of function evaluations.
F1	Line search failure.
F2	Failed because of an uphill search direction.
F3	Failed because of inconsistent constraints.
F4	Failed because of no progress.
F5	Failed because of more than 500 function evaluations.
F6	Failed because of divergence.
<i>d</i>	Converged at different optimal points.
<i>s</i>	Terminated abnormally at the point near an optimum.
Machine	CDC Cyber 180-855 .

Appendix C.1

Alkylation Process

A simplified alkylation process, shown in Figure 3.1, was proposed by Payne [1958] and optimised by Sauer, Colville and Burwick [1964].

The objective function and process equations are described as follows

Objective function : (Maximise)

Product (Product Values) - Feed (Feed Cost) - Operating Cost

$$0.063 X_4 X_7 - 5.04 X_1 - 0.035 X_2 - 10 X_3 - 3.36 X_5$$

Process Equations :

Mass Balance on Isobutane

$$X_2 + X_5 - X_1 X_8 = 0 \quad (1)$$

Mass Balance on Acid

$$1000 X_3 (98 - X_6) - X_4 X_6 X_9 = 0 \quad (2)$$

Empirical Relation for Alkylate Reaction

$$100 X_4 = X_1 (112 + (13.167 - 0.6667 X_8) X_8) \quad (3)$$

Mass Balance on Alkylate

$$X_5 = (1 + 0.22) X_4 - X_1 \quad (4)$$

Empirical Relation for Octane Number

$$X_7 = 86.35 + (1.098 - 0.038 X_8) X_8 + 0.325 (X_6 - 89) \quad (5)$$

Empirical Relation for Acid Dilution Factor

$$X_9 = 35.82 - 0.222 X_{10} \quad (6)$$

Empirical Relation for Performace Number

$$X_{10} = -133 + 3 X_7 \quad (7)$$

The variables are described as follows

	Starting value	Lower bound	Upper bound	optimal value
Variables				
X1, Olefin Feed, ($\frac{B}{Day}$)	1745	0	2000	1728.92
X2, Isobutane Recycle, ($\frac{B}{Day}$)	12000	0	16000	16000
X3, Acid Addition Rate, ($\frac{MLB}{Day}$)	110	0	120	98.16
X4, Alkylate Yield, ($\frac{B}{Day}$)	3048	0	5000	3056.49
X5, Isobutane Make-Up, ($\frac{B}{Day}$)	1974	0	2000	2000
X6, Acid Strength, (Wt %)	89.2	85	93	90.616
X7, Motor Octane No,	92.8	90	95	94.1877
X8, Isobutane / Olefin Ratio,	8.0	3	12	10.4111
X9, Acid Dilution Factor,	3.6	0	4	2.61694
X10, Performance Number,	145	145	162	149.563

Appendix C.2

Williams — Otto Process

The Williams-Otto process, shown in Figure 3.2 and proposed by Williams and Otto [1964], is to manufacture 4763 *lbs/hr* of pure product P_w with maximum investment return. The system involves a cooler, a decanter, and a distillation column. The pure reactants, A_w and B_w are fed separately to the reactor and the pure product P_w , intermediates C_w , and the by-product E_w and G_w are produced. The reactions are assumed to be second order and irreversible. The undesirable byproduct G_w is discarded from the decanter. Part of the product P_w forms an azeotrope with the byproduct E_w in the distillation column.

The process equations are described as follows.

Description of the Process (Mass Balance Only)						
Reactions						
(1) $A_w + B_w \xrightarrow{\kappa_1} C_w$						
(2) $B_w + C_w \xrightarrow{\kappa_2} P_w + E_w$						
(3) $P_w + C_w \xrightarrow{\kappa_3} G_w$						
Reaction coefficients						
$\kappa_i = \gamma_i \exp(-\theta_i/T_r) \quad i = 1, \dots, 3$						
Reaction Expressions		$\gamma \text{ (hr}^{-1}\text{)}$		$\theta \text{ (}^\circ\text{R)}$		
$R_1 = \kappa_1 \frac{F_{RA} F_{RB}}{F_R^2} V_r \rho$		5.9755		12000		
$R_2 = \kappa_2 \frac{F_{RB} F_{RC}}{F_R^2} V_r \rho$		2.5962		15000		
$R_3 = \kappa_3 \frac{F_{RC} F_{RP}}{F_R^2} V_r \rho$		9.6283		20000		
Molecular weight	W_A	W_B	W_C	W_E	W_G	W_P
<i>lb</i>	100	100	200	200	300	100

Process Equations (Mass Balance Only)

Definition of total reactor outlet rate

$$F_R = F_{RA} + F_{RB} + F_{RC} + F_{RE} + F_{RP} + F_G \quad (1)$$

Mass Balance for species A_w

$$F_A = \alpha_s F_{RA} + R_1 \quad (2)$$

Mass Balance for species B_w

$$F_B = \alpha_s F_{RB} + R_1 + R_2 \quad (3)$$

Mass Balance for species C_w

$$\alpha_s F_{RC} = \frac{W_C}{W_B} R_1 - \frac{W_C}{W_B} R_2 - R_3$$

Mass Balance for species E_w

$$\alpha_s F_{RE} = \frac{W_E}{W_B} R_2 \quad (4)$$

Mass Balance for species P_w

$$F_P + \alpha_s (F_{RP} - F_P) = R_2 - \frac{W_P}{W_C} R_3 \quad (5)$$

Mass Balance for species G_w

$$F_G = \frac{W_G}{W_C} R_3 \quad (6)$$

Distillation column efficiency

$$F_{RP} - 0.1 F_{RE} - F_P = 0 \quad (7)$$

Total mass Balance

$$F_A + F_B - F_G - F_P - F_D = 0 \quad (8)$$

Definition of the split factor of the splitter

$$\alpha_s = F_D / (F_R - F_G - F_P)$$

Physical limitation

$$580 \leq T \leq 680$$

Objective Function

Gross Return, $Cost1$

$$0.3F_P + 0.0068F_D - 0.02F_A - 0.03F_B - 0.01F_G$$

Utility Charge $Cost2$

$$2.22F_R$$

Sales, Administration, Research, and Engineering Charge, $Cost3$

$$0.124(8400)(0.3F_P + 0.0068F_D)$$

Investment (a), $Cost4$

$$30V_r$$

Investment (b), $Cost5$

$$30V_r + 1.3F_R$$

Case I

Maximise — Objective function — (a)

$$(Cost1 + Cost2 + Cost3 + Cost4) / Cost4$$

Design Constraints

$$V - 60 = 0$$

Case II

Maximise — Objective function — (b)

$$(Cost1 + Cost2 + Cost3 + Cost4) / Cost4$$

Case III

Maximise — Objective function — (c)

$$(Cost1 + Cost2 + Cost3 + Cost5) / Cost5$$

The initial point and optimal point obtained from different researchers for the objective function (a) are described as follows.

Variables	Initial point	Optimal point (case I)		
		This study	Jung <i>et al.</i>	Vinante Valladares
F_A	11540	12521.96	12518	12367
F_B	31230	28358.45	28365	28494
F_R	92640	326882	326428	318521
F_{RA}	8820	38515	38420	35897
F_{RB}	29910	121863	121754	119742
F_{RC}	2360	7315	7296	6754
F_{RE}	33160	137743	137541	134904
F_{RP}	7890	18537	18517	18251
F_G	2010	2907	2908	2971
F_P	4763	4763	4763	4761
F_D	36010	33209	33212	33128
T_r	610	654	654	656
V_r	60	60	60	60
Obj	108.5	98.68	98.68	98.87
CV	-	10^{-7}	0.24	0.001

Variables	Initial	Optimal point (case II)		
	point	This study	Adelmann <i>et al.</i>	Rijkaert <i>et al.</i>
F_A	11540	13505.8	13338	13505
F_B	31230	30780.8	30759	30782
F_R	92640	370476.	360538	370411
F_{RA}	8820	47450.5	45977	47399
F_{RB}	29910	147110.	142784	147039
F_{RC}	2360	7771.4	7725	7793
F_{RE}	33160	145602.9	141919	145598
F_{RP}	7890	19323.2	18955	19322
F_G	2010	3212.9	3175	3212
F_P	4763	4763	4763	4763
F_D	36010	36310.8	36159	36312
T_r	610	674.4	672.61	674
V_r	60	30.74	32.0	30.75
Obj	108.5	121.52	121.66	121.54
CV	-	10^{-7}	-	-

Variables	Initial point	Optimal point (case III)		
		This study	Sriram <i>et al.</i>	Rijkaert <i>et al.</i>
F_A	11540	12506.7	12353	12507
F_B	31230	28780.3	28716	28780
F_R	92640	144456.6	147057	144458
F_{RA}	8820	15651.92	15420	15653
F_{RB}	29910	52572.54	53641	52653
F_{RC}	2360	3408.593	3339	3409
F_{RE}	33160	59081.37	60724	59082
F_{RP}	7890	10671.04	10835	10671
F_G	2610	3071.143	3096	3071
F_P	4763	4763	4763	4763
F_D	36010	33453.1	33210	33454
T_r	610	634.046	634.9	634
V_r	60	122.045	122.4	122
Obj	61	50.82	50.82	50.82
CV	-	10^{-7}	-	-

key

F_A	Fresh flow rate of A_w , $\frac{lbs.}{hour}$
F_B	Fresh flow rate of B_w , $\frac{lbs.}{hour}$
F_D	Flow rate of D_w sent as fuel, $\frac{lbs.}{hour}$
F_G	Reactor outlet rate of G_w , $\frac{lbs.}{hour}$
F_G	Production rate of P_w , $\frac{lbs.}{hour}$
F_{RA}	Reactor outlet rate of A_w , $\frac{lbs.}{hour}$
F_{RB}	Reactor outlet rate of B_w , $\frac{lbs.}{hour}$
F_{RC}	Reactor outlet rate of C_w , $\frac{lbs.}{hour}$
F_{RE}	Reactor outlet rate of E_w , $\frac{lbs.}{hour}$
F_{RP}	Reactor outlet rate of P_w , $\frac{lbs.}{hour}$
V_r	Reactor volume, ft^3
T_r	Reactor temperature, $^{\circ}R$
Obj	Objective function
CV	Constraint violations, i.e. $\sum_{i=1}^{meq} h_i(x) + \sum_{j=1}^{mi} \max(0, -g_j(x))$

Appendix D.1

This appendix is concerned with the estimation of the perturbation size for gradient evaluations.

Consider a system of functions $h(x)$ and an element of the gradients $\frac{\partial x_i h_i(x)}{\partial x_j}$ generated from a two-point finite difference scheme, i.e.

$$\frac{h_i(x + \zeta_j e_j) - h_i(x)}{\zeta_j}$$

There are in general three kinds of errors associated with this perturbation technique for modular oriented approach, namely rounding error, procedure error and truncation error. The rounding error is due to the limited machine precision, the procedure error is introduced by the iterative procedures and the truncation error results from approximating the functions.

The relative rounding error may be estimated from

$$\left\| \frac{h_{cal}(x + \zeta_j e_j) - h_{cal}(x)}{\zeta_j} \right\|_2 = \frac{\xi_1 \|h(x)\|_2}{\zeta_j}$$

where ξ_1 is an estimated relative error.

The relative procedure error may take the form

$$\left\| \frac{h_{cal}(x + \zeta_j e_j) - h_{cal}(x)}{\zeta_j} \right\|_2 = \frac{\xi_2 \|h(x)\|_2}{\zeta_j}$$

where ξ_2 is an estimated relative error.

The relative truncation error may be expressed as

$$\frac{\zeta_j}{2} \left\| \frac{\partial^2 h(x')}{\partial x_j^2} \right\|_2$$

where $x \leq x' \leq x + \zeta_j$.

The perturbation size can be selected so that these errors are minimised, i.e. by solving

$$\min_{\zeta_j} \frac{\zeta_j}{2} \left\| \frac{\partial^2 h(x')}{\partial x_j^2} \right\|_2 + \frac{\xi_1 \|h(x)\|_2}{\zeta_j} + \frac{\xi_2 \|h(x)\|_2}{\zeta_j}$$

The solution of this problem is

$$\zeta_j^* = \sqrt{\frac{2(\xi_1 + \xi_2) \|h(x)\|_2}{\left\| \frac{\partial^2 h(x')}{\partial x_j^2} \right\|_2}}$$

In practice the estimate the error ξ_2 is larger than the error ξ_1 and it is difficult to calculate both the relative procedure error and the second order term $\left\| \frac{\partial^2 h(x')}{\partial x_j^2} \right\|_2$ without extra function evaluations.

In this study, ξ_3 approximates the relative tolerance of the flowsheet simulation in the feasible path and the average procedure tolerance in the infeasible path. The optimal perturbation size ζ_j become

$$\zeta_j^* \approx \sqrt{\xi_3} \max(1, |x_j|)$$

Appendix E.1

Chapter 3 shows that a reduced QP subproblem can be formulated as

$$\begin{aligned} \min_{p_z} \quad & -p_z^T \bar{Z}^T H Y h(x) + \frac{1}{2} p_z^T Z^T H Z p_z \\ \text{s.t.} \quad & (g(x) - \nabla_x g(x)^T Y h(x)) + \nabla_x g(x)^T Z p_z \geq 0 \end{aligned}$$

where

- H is the Hessian of the Lagrangian function.
- Y is an $(n \times meq)$ matrix.
- Z is an $(n \times (n - meq))$ matrix.
- $h(x)$ is an meq -dimensional vector of equality constraints.
- $g(x)$ is an mi -dimensional vector of inequality constraints.
- p_z is an $(n - meq)$ -dimensional vector of search directions.

$$\begin{aligned} \nabla_x h(x)^T Y &= I \\ \nabla_x h(x)^T Z &= \hat{O} \end{aligned} \quad (3.4)$$

This appendix shows that the matrices Y , Z , $Z^T H Z$ and $Z^T H Y$ can be efficiently obtained by solving the following equations with a given matrix $\underline{A}$.

$$\begin{matrix} n & \overbrace{\left(\begin{matrix} \nabla_x h & \underline{A} \\ 0 & \end{matrix} \right)}^n & \overbrace{\left(\begin{matrix} H \\ (\nabla_x h \ \underline{A})^T \end{matrix} \right)}^n \end{matrix} \left(\begin{pmatrix} Y^T \\ Z^T \\ 0 \end{pmatrix} - \begin{pmatrix} Y^T \\ Z^T \\ (Y \ Z) \end{pmatrix} H (Y \ Z) \right) = I \quad (e.1)$$

i.e.

$$\begin{pmatrix} \nabla_x h & \underline{A} \\ 0 & \end{pmatrix} \begin{pmatrix} H \\ (\nabla_x h \ \underline{A})^T \end{pmatrix} \begin{pmatrix} Y^T & -Y^T H Y & -Y^T H Z \\ Z^T & -Z^T H Y & -Z^T H Z \\ 0 & (Y \ Z) \end{pmatrix} = I \quad (e.2)$$

where $\underline{A}$ is an $(n \times (n - meq))$ matrix such that

$$\begin{aligned} \underline{A}^T Y &= \hat{O} \\ \underline{A}^T Z &= I \end{aligned} \quad (e.3)$$

From equation (3.4) and (e.3), the following condition is satisfied

$$\begin{pmatrix} \nabla_x h^T \\ \underline{A}^T \end{pmatrix} (Y \ Z) = I \quad (3.43)$$

From equation (3.4) and (3.43), equation (e.1) and (e.2) are satisfied.

In general the matrix $\underline{\Lambda}$ can be chosen as either

$$\underline{\Lambda}^T = (\hat{O} \ I) \quad (e.4)$$

or

$$\underline{\Lambda}^T = Q_2. \quad (e.5)$$

where Q_2 is a matrix whose columns are an orthonormal basis of the null space of $\nabla_x h(x)^T$ and can be obtained from an orthogonal decomposition, e.g. equation (3.7).

The first choice results in the direct variable elimination and the matrix Y and Z obtained from solving equation (3.43) become

$$\begin{aligned} Z &= \begin{pmatrix} -\nabla_v h(x)^{-T} & \nabla_u h(x)^T \\ & I \end{pmatrix} \\ Y &= \begin{pmatrix} \nabla_v h(x)^{-T} \\ \hat{O} \end{pmatrix} \end{aligned} \quad (3.6)$$

In practice the matrix $\underline{\Lambda}$ can be chosen as the matrix Λ with exactly one 1 in each row and column corresponding to the position of a design variable and zero elsewhere. For example, consider the following equality constraints

$$x_1 + x_2 + x_3 + x_4 = 0 \quad (e.6)$$

$$x_1 + 2x_2 + 3x_3 + x_4 = 0 \quad (e.7)$$

If the variables x_1 and x_4 are selected as design variables then $\underline{\Lambda}$ become

$$\underline{\Lambda} = \begin{matrix} & \begin{matrix} x_1 & x_2 & x_3 & x_4 \end{matrix} \\ \begin{matrix} e.6 \\ e.7 \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix}$$

The second choice results in the projection method and the matrix Y and Z become

$$\begin{aligned} Y &= Q_1 R^{-T} \\ Z &= Q_2 \end{aligned} \quad (3.8)$$

where

R an $(meq \times meq)$ upper triangular matrix.

Q_1 an $(n \times meq)$ matrix.

Q_2 an $(n \times (n - meq))$ matrix.

Other choices of $\underline{\Lambda}$ are still possible.

References

Part I. Optimisation Methods

- Abadie, J. and Carpentier, J. [1969],
“ Generalisation of the Wolfe Reduced Gradient Method to the Case of Non-linear Constraints, ”
in *Optimisation* (R. Fletcher, ed), Academic Press, New York, pp. 37-47.
- Armijo, L. [1966],
“ Minimisation of Function Having Lipschitz Continuous First Partial Derivatives, ”
Pacific J. Math., 16, pp. 1-3.
- Bachem, A., Grötschel, M. and Korte B. (eds) [1983],
“ Mathematical Programming - The State of the Art, Bonn, 1982, ”
Springer-Verlag, Berlin.
- Bartels, R. H. and Mahdavi-Amiri, N [1986],
“ On Generating Test Problems for Nonlinear Programming Algorithms, ”
SIAM J. Sci. Stat. Comput., 7, pp. 769-798.
- Bartholomew-Biggs, M. C. [1972],
“ Constrained Minimisation Using Recursive Equality Quadratic Programming, ”
in *Numerical Methods for Nonlinear Optimisation* (F. A. Lootsma, ed.), Academic Press, London, pp. 411-428.
- Bartholomew-Biggs, M. C. [1982],
“ Recursive Quadratic Programming Methods for Nonlinear Constraints, ”
in *Nonlinear Optimisation, 1981* (M.J.D. Powell, ed.), Academic Press London, pp. 213-221.
- Bartholomew-Biggs, M. C. [1985],
“ Matrix Updating in Nonlinear Programming Calculations, ”
Technical Report No. 156 Numerical Optimisation Centre, The Hatfield Polytechnic.
- Bartholomew-Biggs, M. C. [1986],
“ Numerical Examples of the Behavior of *REQP* on Nonlinear Programming Problems Involving Linear Dependence among the Constraint Normals, ”
J. Optim. Theor. Appl., 48, pp. 215-227.
- Bartholomew-Biggs, M. C. [1987],
“ Recursive Quadratic Programming Methods Based on the Augmented Lagrangian, ”
Math. Prog. Study, 31, pp. 21-41.

- Beale, E. M. L. [1967],
 “ Numerical Methods, ”
 in *Nonlinear Programming* (J. Abadie, ed.), North-Holland, Amsterdam,
 pp. 135-205.
- Belegundu, A. D. and Arora J. S. [1984],
 “ A Recursive Quadratic Programming Method with Active Set Strategy for
 Optimal Design, ”
 in *Int. J. Numer. Methods Eng.* 20, pp. 803-816.
- Bell, B. M. [1984],
 “ Nonsmooth Optimisation by Successive Quadratic Programming ”
 Ph. D. Thesis, University of Washington.
- Bertsekas, D. P. [1982],
 “ Constrained Optimisation and Lagrangian Multiplier Methods ”
 Academic Press, London and New York.
- Boggs, P. T. and Tolle, J. W. [1980],
 “ Augmented Lagrangians Which are Quadratic in the Multiplier, ”
J. Optim. Theor. Appl., 31, pp. 17-26.
- Boggs, P. T. and Tolle, J. W. [1982],
 “ Merit Functions for Nonlinear Programming Problems, ”
 in *Numerical Analysis, Lecture Notes in Mathematics*, 909 (J. P. Hennart,
 ed.), Springer-Verlag, Berlin, pp. 1-10.
- Boggs, P. T., Tolle, J. W., and Wang, P. [1982],
 “ On the Local Convergence of Quasi-Newton Methods for Constrained Opti-
 misation, ”
 in *SIAM J. Control. Optim.*, 20, pp. 161-171.
- Boggs P. T. and Tolle, J. W. [1984],
 “ A Family of Descent Functions for Constrained Optimisation, ”
SIAM J. Numer. Anal., 21, pp. 1146-1161.
- Boggs, P. T., Byrd R. H. and Schnabel, R. B. (eds.) [1985],
 “ Numerical Optimisation, 1984, ”
 SIAM Publications, Philadelphia.
- Boggs P. T. and Tolle, J. W. [1986],
 “ An Efficient Strategy for Utilising a Merit Function in Nonlinear Program-
 ming Algorithms, ”
 in *Numerical Analysis, Proceedings, Guanajuato, Mexico, 1984, Lecture
 Notes in Mathematics* 1230, (Hennart, J. P. ed.), Spring-Verlag, Berlin,
 pp. 127-137.

- Broyden, C. G. and Attia, N. F. [1984],
 " A Smooth Sequential Penalty Function Method for Solving Nonlinear Programming Problems, "
 in *System Modeling and Optimisation, Lecture Notes in Control and Information Sciences*, 59, (A. V. Balakrishnan and M. Thomas, eds) Springer-Verlag, Berlin, pp. 237-245.
- Burchett, R. C. and Happ, H. H., and Vierath, D. R. [1984],
 " Quadratically Convergent Optimal Power Flow, "
IEEE Transaction on Power Apparatus and Systems, PAS-103, pp. 3267-3271.
- Byrd, R. H. [1985],
 " An Example of Irregular Convergence in Some Constrained Optimisation Methods that Use the Projected Hessian, "
Math. Prog., 32, pp. 232-237.
- Byrd, R. H., and Schnabel, R. B. [1986],
 " Continuity of the Null Space Basis and Constrained Optimisation, "
Math. Prog., 35, pp. 32-41.
- Byrd, R. H., Schnabel, R. B. and Shultz, G. A. [1985],
 " A Trust Region Algorithm for Nonlinear Constrained Optimisation, "
 Report CU-CS-313-85, University of Colorado.
- Celis, M. R., Dennis, J. E. and Tapia, R. A. [1985],
 " A Trust Region Strategy for Nonlinear Equality Constrained Optimisation, "
 in *Numerical Optimisation, 1984* (P.T. Boggs, R.H. Byrd and R.B. Schnabel, eds.), SIAM Publications, Philadelphia, pp. 71-82.
- Chamberlain, R.M. [1979],
 " Some Examples of Cycling in Variable Metric Methods for Constrained Minimisation, "
Math. Prog. 16, pp. 378-383.
- Chamberlain, R.M., Lemaréchal, C., Pedersen H. C. and Powell, M.J.D. [1982],
 " The Watchdog Technique for Forcing Convergence in Algorithms for Constrained Optimisation, "
Math. Prog. Study, 16, pp. 1-17.
- Coleman, T. F. [1984],
 " Large Sparse Numerical Optimisation, "
Lecture Notes in Computer Science, (G. Goos and J. Hartmanis eds.), Springer-Verlag, Berlin.
- Coleman, T. F. and Conn., A. R. [1982 (a)],
 " Nonlinear Programming via an Exact Penalty Function : Asymptotic Analysis, "
Math. Prog., 24, pp. 123-136.

- Coleman, T. F. and Conn., A. R. [1982 (b)],
 “ Nonlinear Programming via an Exact Penalty Function : Global Analysis, ”
Math. Prog., 24, pp. 137-161.
- Coleman, T. F. and Conn., A. R. [1984],
 “ On the Local Convergence of a Quasi-Newton Method for the Nonlinear Programming, ”
SIAM J. Numer. Anal., 22, pp. 755-769.
- Coleman, T. F. and Sorensen, D. C. [1984],
 “ A Note on the Computation of an Orthonormal Basis for the Null Space of a Matrix, ”
Math. Prog., 29, pp. 234-242.
- Colville, A. R. [1968],
 “ A Comparative Study on Nonlinear Programming Codes, ”
 Report 320-2949, IBM New York Scientific Centre, June.
- Conn, A. R., Gould, N.I.M. and Toint, Ph. L [1988],
 “ Global Convergence of a Class of Trust Region Algorithms for Optimisation with Simple Bounds, ”
SIAM J. Numer. Anal., 25, pp. 433-460.
- Cornwell, L. W., Hutchison, P. A., Minkoff, M. Schultz, H. k.[1978],
 “ Test Problems for Constrained Nonlinear Mathematical Programming Algorithms, ”
 Technical Memorandum No 320, Argonne National Laboratory, Applied Mathematics Division, October, Argonne, Illinois.
- Curtis, A. R., Powell, M. J. D. and Reid, J. K. [1974],
 “ On the Estimation of Sparse Jacobian Matrices, ”
J. Inst. Math. Appl., 13, pp. 117-119.
- Dembo, R. S. [1976],
 “ A Set of Geometric Programming Test Problems and Their Solutions, ”
Math Prog., 10, pp. 192-213.
- Dembo, R. S. [1987],
 “ A Primal Truncated Newton Algorithm with Application to Large-Scale Nonlinear Network Optimisation, ”
Math Prog. Study, 31, pp. 43-71.
- Dembo, R. S. and Tulowitzki, U. [1985],
 “ Sequential Truncated Quadratic Programming Methods, ”
 in *Numerical Optimisation 1984* (P.T. Boggs, R.H. Byrd and R.B. Schnabel, eds.), SIAM Publications, Philadelphia, pp. 83-101.
- Di Pillo, G. and Grippo, L. [1979],
 “ A New Class of Augmented Lagrangians in Nonlinear Programming, ”
SIAM J. Control Optim., 17, pp. 618-628.

- Di Pillo, G. and Grippo, L. [1985],
 “ A Continuously Differentiable Exact Penalty Function for Nonlinear Programming Problems with Inequality Constraints, ”
SIAM J. Control Optim., 23, pp. 72-84.
- Di Pillo, G. and Grippo, L. [1986],
 “ An Exact Penalty Function Methods with Global Convergence Properties for Nonlinear Programming Problems, ”
Math. Prog., 36, pp. 1-18.
- Dixon, L. C. W., Spedicato, W. and Szegő, G. P. (eds) [1980],
 “ Nonlinear Optimisation – Theory and Algorithms, ”
 Birkhäuser, Boston.
- Drud, A. [1985],
 “ CONOPT – a GRG Code for Large Sparse Dynamic Nonlinear Optimisation Problems, ”
Math. Prog., 31, pp. 153-191.
- Duff, I. S. [1977],
 “ MA28 – A Set of Fortran Subroutines for Sparse Unsymmetric Linear Equations, ”
 AERE R8730, HMSO, London.
- Eason, E. D. and Fenton, F. G. [1974],
 “ A Comparison of Numerical Optimisation Methods for Engineering Design, ”
Trans. ASME, Ser. B, 96, pp. 196-200.
- Escudero, L. F. [1986],
 “ A Motivation for Using the Truncated Newton Approach in a Very Large Scale Nonlinear Network Problem, ”
Math. Prog. Study, 26, pp. 240-244.
- Evtushenko, Y. G. [1985],
 “ Numerical Optimisation Techniques, ”
 Optimisation Software, Inc., New York. (Engl. transl.)
- Fan, Y., Sarkar, S. and Lasdon, L. S. [1988]
 “ Experiments with Successive Quadratic Programming Algorithms, ”
J. Optim. Theor. Appl., 56, pp. 359-383.
- Fiacco, A. V. [1983],
 “ Introduction to Sensitivity and Stability Analysis in Nonlinear Programming, ”
 Academic Press, New York and London.
- Fiacco, A. V. and McCormick, G. P. [1968],
 “ Nonlinear Programming : Sequential Unconstrained Minimisation Techniques, ”
 John Wiley and Sons, New York and Toronto.

- Fletcher, R. [1970],
 “ A Fortran Subroutine for General Quadratic Programming, ”
 Report AERE-R 6370, Harwell.
- Fletcher, R. [1971],
 “ A General Quadratic Programming Algorithm, ”
J. Inst. Math. Appl., 7, pp.76-91.
- Fletcher, R. [1973],
 “ An Exact Penalty Function for Nonlinear Programming with Inequalities, ”
Math. Prog., 5, pp. 129-150.
- Fletcher, R. [1981],
 “ Numerical Experiences with an Exact l_1 Penalty Function Method, ”
 in *Nonlinear Programming 4* (O.L. Mangasarian, R.R. Meyer, and S.M. Robinson, eds), Academic Press, London and New York, pp. 99-129.
- Fletcher, R. [1982],
 “ Second Order Corrections for Nonlinear Differentiable Optimisation, ”
 in *Numerical Analysis, Lecture Notes in Mathematics, 912* (G. A. Watson, ed.), Springer-Verlag, Berlin, pp. 85-114.
- Fletcher, R. [1985],
 “ An l_1 Penalty Method for nonlinear Constraints, ”
 in *Numerical Optimisation 1984* (P.T. Boggs, R.H. Byrd and R.B. Schnabel eds), SIAM Publications, Philadelphia, pp. 26-40.
- Fletcher, R. [1988],
 “ Practical Methods of Optimisation, ”
 2nd edition, John Wiley and Sons, New York.
- Fletcher, R. and Sainz de la Maza, E. [1987],
 “ Nonlinear Programming and Nonsmooth optimisation by Successive Linear Programming, ”
 Technical Report No. NA/100, March, University of Dundee, Scotland.
- Fontecilla, R. [1985],
 “ On Inexact Newton Methods for Constrained Optimisation, ”
 in *Numerical Optimisation 1984*, (P.T. Boggs, R.H. Byrd and R.B. Schnabel, eds.), SIAM Publications Philadelphia, pp. 102-118.
- Fontecilla, R., Steihaug, T. and Tapia, R. A. [1987],
 “ A Convergence Theory for a Class of Quasi-Newton Methods for Constrained Optimisation, ”
SIAM J. Numer. Anal., 24, pp. 1133-1151.
- Fukushima, M. [1986],
 “ A Successive Quadratic Programming Algorithm With Global and Superlinear Convergence Properties, ”
Math. Prog., 35, pp. 253-264.

- Gabay, D [1982],
 “ Reduced Quasi-Newton Methods with Feasibility Improvement for Nonlinear Constrained Optimisation, ”
Math. Prog. Study, 16, pp. 18-44.
- Gabriele, G. A. [1980],
 “ Large Scale Nonlinear Programming Using the Generalised Reduced Gradient Method, ”
 Ph.D. Thesis, Purdue University.
- Gafni, E. M. and Bertsekas D. P. [1984],
 “ Two Metric Projection Methods for Constrained Optimisation, ”
SIAM J. Control Optim., 22, pp. 936-964.
- Garbow, B. S., Boyle, J. M., Dongarra, J. J., and Moler, C. B. [1977],
 “ Matrix Eigensystem Routines - EISPACK Guide Extension, ”
 Lecture Notes in Computer Science, Springer-Verlag, Berlin.
- Garcia-Palomares, U. M. and Mangasarian, O. L. [1976],
 “ Superlinearly Convergent Quasi-Newton Algorithms for Nonlinearly Constrained Optimisation Problems, ”
Math. Prog., 11, pp. 1-13.
- Gay, D. M. [1984],
 “ A Trust Region Approach to Linearly Constrained Optimisation, ”
 in *Numerical Analysis, 1983, Lecture Notes in Mathematics, 1066* (D. F. Griffiths, eds), Springer-Verlag, Berlin, pp. 72-105.
- Gill, P. E. and Murray, W. (eds) [1974],
 “ Numerical Methods for Constrained Optimisation, ”
 Academic Press, London and New York.
- Gill, P. E. and Murray, W. [1979],
 “ The Computation of Lagrangian Multiplier Estimates for Constrained Minimisation, ”
Math. Prog., 17, pp. 32-60.
- Gill, P. E., Murray, W. and Wright, M. H. [1981 (a)],
 “ Practical Optimisation, ”
 Academic Press, London and New York.
- Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. [1981 (b)],
 “ QP - Based Methods for the Large-Scale Nonlinearly Constrained Optimisation, ”
 in *Nonlinear Programming 4*, (Mangasarian, O. L., R. R. Meyer, and S. M. Robinson, eds), Academic Press, London and New York, pp. 57-98.

- Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. [1983 (a)],
 “ User’s Guide for QPSOL (Version 3.2) : A Fortran Package for Quadratic Programming, ”
 Report SOL 84-6, Department of Operations Research, Standford University, California.
- Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. [1983 (b)],
 “ Computing Forward Difference Intervals for Numerical Optimisation, ”
SIAM J. Sci. Stat. Comput., 4, pp. 310-321.
- Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. [1984 (a)],
 “ Trends in Nonlinear Programming Software, ”
European J. Operational Research, 17, pp. 141-149.
- Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. [1984 (b)],
 “ Sparse Matrix Methods in Optimisation, ”
SIAM J. Sci. Stat. Comput., 5, pp. 562-589.
- Gill, P. E., Murray, W., Saunders, M. A., Stewart, G. W. and Wright, M. H. [1985 (a)],
 “ Properties of a Representation of a Basis for the Null Space, ”
Math. Prog., 33, pp. 172-186.
- Gill P. E., Murray, W., Saunders, M. A. and Wright, M. H. [1985 (b)],
 “ Model Building and Practical Aspects of Nonlinear Programming, ”
 in *Computational Mathematical Programming* (K. Schittkowski, ed.)
 NATO ASI Series, Springer-Verlag, Berlin, pp. 209-247.
- Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. [1986],
 “ Considerations of Numerical Analysis in a Sequential Quadratic Programming Method, ”
 in *Numerical Analysis, Proceedings, Guanajuato, Mexico 1984*, (J. P. Hennart, ed.,) *Lecture Notes in Mathematics 1230*, Spring-Verlag, Berlin, pp. 46-62
- Goldfarb, D. [1986],
 “ Efficient Primal Algorithms for Strictly Convex Quadratic Programs, ”
 in *Numerical Analysis, Proceedings, Guanajuato, Mexico 1984*, (J. P. Hennart ed.,) *Lecture Notes in Mathematics 1230*, Spring-Verlag, Berlin, pp. 11-25.
- Goldfarb, D. and Idnani, A. [1982],
 “ Dual and Primal-Dual Methods for Solving Strictly Convex Quadratic Programs, ”
 in *Numerical Analysis, Proceedings, Cocoyoc, Mexico 1984*, (J. P. Hennart ed.,) *Lecture Notes in Mathematics 909*, Spring-Verlag, Berlin, pp. 226-239.

- Goldfarb, D. and Idnani, A. [1983],
 “ A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Programs, ”
Math. Prog., 27, pp. 1-33.
- Gould, N. I. M. [1986],
 “ On the Accurate Determination of Search Directions for Simple Differentiable Penalty Functions, ”
IMA J. Numer. Anal., 6, pp. 357-372.
- Griewank, A. and Toint, Ph. L. [1982 (a)],
 “ Partitioned Variable Metric Updates for Large Structured Optimisation Problems, ”
Numer. Math., 39, pp. 119-137.
- Griewank, A. and Toint, Ph. L. [1982 (b)],
 “ Local Convergence Analysis for Partitioned Quasi-Newton Updates, ”
Numer. Math., 39, pp. 429-448.
- Griffith, R. E. and Stewart, R. A. [1961],
 “ A Nonlinear Programming Technique for the Optimisation of Continuous Processing Systems, ”
Manage. Sci. 7, pp. 379-392.
- Han, S. P. [1977],
 “ A Globally Convergent Method for Nonlinear Programming, ”
J. Optim. Theor. Appl., 22, pp. 297-310.
- Himmelblau, D. M. [1972],
 “ Applied Nonlinear Programming, ”
 McGraw-Hill, New York.
- Himmelblau, D. M. and Lindsay, J. W. [1980],
 “ An Evaluation of Substitute Methods for Derivatives in Unconstrained Optimisation, ”
Operations Research, 28, pp. 668-686.
- Hock, W. and Schittkowski, K. [1981],
 “ Test Examples for Nonlinear Programming Codes, ”
Lecture Note in Economics and Mathematical System, 187, Springer-Verlag, Berlin.
- Iserles, A. and Powell, M. J. D. (eds.) [1987],
 “ The State of the Art in Numerical Analysis, ”
 Clarendon Press Oxford.
- Karmarkar, N. [1984],
 “ A New Polynomial-Time Algorithm for Linear Programming, ”
Combinatorica, 4, pp. 373-385.

- Kuhn, H. W. [1976],
 “ Nonlinear Programming : A Historical View, ”
 in *Nonlinear Programming*, (R. W. Cottle and C. E. Lemke, eds.), SIAM-AMS Proceedings, New York City, 9, pp. 1-26.
- Lasdon, L. S. [1985],
 “ Nonlinear Programming Algorithms - Applications, Software, and Comparison, ”
 in *Numerical Optimisation 1984* (P.T. Boggs, R.H. Byrd and R.B. Schnabel, eds.), SIAM Publications, Philadelphia, pp. 40-70.
- Lasdon, L. S., Waren, A. D., Jain, A. D. and Ratner, M. 1978],
 “ Design and Testing of a Generalised Reduced Gradient Software for Nonlinear Programming, ”
ACM Transactions on Mathematical Software, March, 1978, pp. 34-50.
- Lemke, C. E. [1962],
 “ A Method of Solution for Quadratic Program, ”
Manage. Sci. 8, pp. 442-455.
- Lim, O. K. and Arora J. S. [1986],
 “ An Active Set RQP Algorithm for Engineering Design Optimisation, ”
 in *Comp. Methods Appl. Mech. Eng.* 57, pp. 51-65.
- Lin, Y. Y. and Pang, J. S. [1987],
 “ Iterative Methods for Large Convex Quadratic Programs : A Survey, ”
 in *SIAM J. Control. Optim.*, 25, pp. 383-411.
- Lootsma, F. A. [1985],
 “ Comparative Performance Evaluation, Experimental Design, and Generation of Test Problems in Nonlinear Optimisation, ”
 in *Computational Mathematical Programming* (K. Schittkowski, ed.)
 NATO ASI Series, Springer-Verlag, Berlin, pp. 249-260.
- Luenberger, D. G. [1984],
 “ Linear and Nonlinear Programming, ”
 2nd edition, Addison-Wesley, Massachusetts, London.
- Maany, Z. A. [1987],
 “ A New Algorithm for Highly Curved Constrained Optimisation, ”
Math. Prog. Study, 31, pp. 139-154.
- Mangasarian, O. L., Meyer R. R. and Robinson S. M. (eds.) [1981],
 “ Nonlinear Programming 3, ”
 Academic Press, New York.
- Maratos, N. [1978],
 “ Exact Penalty Function Algorithm for Finite Dimensional and Control Optimisation Problems, ”
 Ph.D. Thesis, University of London.

- Mayne, D. Q. and Polak, E. [1982],
 “ A Superlinearly Convergent Algorithm for Constrained Optimisation Problems, ”
Math. Prog. Study, 16, pp. 45-61.
- McCormick, G. P. [1983],
 “ Nonlinear Programming – Theory, Algorithms, and Applications, ”
 John Wiley and Sons, New York and Toronto.
- McKeown, J. J. [1980],
 “ An Approach to Sensitivity Analysis, ”
 in *Numerical Optimisation of Dynamic System*, (L. C. W. Dixon and G. P. Szegő, eds.), North-Holland, Amsterdam, pp. 349-362.
- Mine, H. , Fukushima, M. and Tanaka, Y. [1984],
 “ On the Use of ϵ – Most Active Constraints in an Exact Penalty Function Method for Nonlinear Optimisation, ”
IEEE Transactions on Automatic Control, AC-29, pp. 1040-1042
- Minkoff, M. [1981],
 “ Methods for Evaluating Nonearly Programming Software, ”
 in *Nonlinear Programming 4* (O.L. Mangasarian, R.R. Meyer, and S.M. Robinson, eds), Academic Press, London and New York, pp. 519-548.
- Moré J. J. [1983],
 “ Recent Developments in Algorithms and Software for Trust Region Methods, ”
 in *Mathematical Programming; The State of the Art, Bonn, 1982*, (A. Bachem, M. Grötschel, and B. Korte, eds), Springer-Verlag, Berlin, pp. 258-287.
- Morison, K. R. [1984],
 “ Optimal Control of Processes as Described by Systems of Differential Algebraic Equations, ”
 Ph. D. Thesis, University of London.
- Mulvey, J. M. (ed.) [1982],
 “ Evaluating Mathematical Programming Techniques, ”
Lecture Note in Economics and Mathematical System, 199, Springer-Verlag, Berlin.
- Murtagh, B. A. and Saunders, M. A. [1982],
 “ A Projected Lagrangian algorithm and the Implementation for Sparse Nonlinear Constraints, ”
Math. Prog. Study, 16, pp. 84-117.
- Murtagh, B. A. and Saunders, M. A. [1987],
 “ MINOS Users Guide, ”
 Report SOL 83-20R, Department of Operations Research, Stanford University, California.

- Nocedal, J. and Overton, M. L. [1985],
 " Projected Hessian Updating Algorithms for Nonlinearly Constrained Optimisation, "
SIAM J. Numer. Anal., 22, pp. 821-850.
- Palacios-Gomez, F., Lasdon, L. S. and Enquist, E. [1982],
 " Nonlinear Programming by Successive Linear Programming, "
Manage. Sci., 28, pp. 1106-1120.
- Pang, J. S. [1983],
 " Methods for Quadratic Programming : A Survey, "
Comput. Chem. Engng., 7, pp. 583-594.
- Pantoja, J. F. A. De O. [1984],
 " An Algorithm for Constrained Optimisation Problems, "
 Ph.D. Thesis, University of London.
- Polak, E. [1987],
 " On the Mathematical Foundations of Nondifferentiable Optimisation in Engineering Design, "
SIAM Review, 29, pp. 21-89.
- Polak, E., Mayne, D. Q. and Ward, Y. [1983],
 " On the Extension of Constrained Optimisation Algorithm from Differentiable to nondifferentiable problems, "
SIAM J. Control Optim., 21, pp. 179-204.
- Poljak, B. T. [1978],
 " Nonlinear Programming Methods in the Presence of Noise, "
Math. Prog., 14, pp. 87-97.
- Powell, M. J. D. [1978 (a)]
 " A Fast Algorithm for Nonlinearly Constrained Optimisation Calculations, "
 In *Numerical Analysis, Dundee, 1977, Lecture Notes in Mathematics* 630, (G. A. Watson ed.) Springer-Verlag, Berlin, pp. 144-157.
- Powell, M. J. D. [1978 (b)],
 " The Convergence of Variable Metric Methods for Nonlinearly Constrained Optimisation Calculations, "
 in *Nonlinear Programming* 3, (O. L. Mangasarian, R. R. Meyer and S. M. Robinson eds.), Academic Press, New York, pp. 27-63.
- Powell, M. J. D. [1980],
 " Gradient Conditions and Lagrangian Multipliers in Nonlinear Programming,"
 in *Nonlinear Optimisation - Theory and Algorithms*, (L. C. W. Dixon, W. Spedicato and G. P. Szegö, eds), Birkhäuser, Boston, pp. 202-220.
- Powell, M. J. D. (ed.) [1982 (a)],
 " Nonlinear Optimisation, 1981, "
 Academic Press, London and New York.

- Powell, M. J. D. [1982 (b)],
 “ Extensions to Subroutine VF02AD, ”
 in *System Modeling and Optimisation, Lecture Notes in Control and Information Sciences*, 38, (R. F. Drenick and F. Kozin eds),
 Springer-Verlag, New York, pp. 529-538.
- Powell, M. J. D. [1983 (a)],
 “ Variable Metric Methods for Constrained Optimisation, ”
 in *Mathematical Programming; The State of the art, Bonn, 1982*, (A. Bachem, M. Grötschel, and B. Korte, eds), Springer-Verlag, Berlin, pp. 288-311.
- Powell, M. J. D. [1983 (b)],
 “ VMCWD : a Fortran Subroutine for Constrained Optimisation, ”
SIGMAP, Bulletin, 32, April, pp. 4-16,
- Powell, M. J. D. [1983 (c)],
 “ ZQPCVX : a Fortran Subroutine for Convex Quadratic Programming, ”
 Report DAMTP/1983/NA17, Department of Applied Mathematics and Theoretical Physics, University of Cambridge.
- Powell, M. J. D. [1985 (a)],
 “ On the Quadratic Programming Algorithm of Goldfarb and Idnani, ”
Math. Prog. Study, 25, pp. 46-61.
- Powell, M. J. D. [1985 (b)],
 “ The Performance of Two Subroutines for Constrained Optimisation on Some Difficult Test Problems, ”
 in *Numerical Optimisation 1984* (P.T. Boggs, R.H. Byrd and R.B. Schnabel, eds.), SIAM Publications, Philadelphia, pp. 160-177.
- Powell, M. J. D. [1986 (a)],
 “ Convergence Properties of Algorithms for Nonlinear Optimisation, ”
SIAM Review, 28, pp. 487-500.
- Powell, M. J. D. [1986 (b)],
 “ How Bad are the BFGS and DFP Methods When the Objective Function is Quadratic, ”
Math. Prog., 34, pp. 34-47.
- Powell, M. J. D. [1987],
 “ Methods for Nonlinear Constraints in Optimisation Calculations, ”
 in *The State of the Art in Numerical Analysis* (A. Iserles and M. J. D. Powell, eds), Clarendon Press Oxford, pp. 325-357.
- Powell, M. J. D. and Yuan, Y. [1986],
 “ A Recursive Quadratic Programming Algorithm that Uses Differentiable Exact Penalty Functions, ”
Math. Prog., 35, pp. 265-278.

- Pshenichnyj, B. N. [1970],
 “ An Algorithm for the General Problem of Mathematical Programming, ”
Kibernetika (Kiev), 6, pp. 120-125.
- Pshenichnyj, B. N. [1987],
 “ The linearisation Method, ”
Optimisation, 18, pp. 179-196.
- Reklaitis, G. V., Ravindran, A. and Ragsdell, K. M. [1983],
 “ Engineering Optimisation – Methods and Applications, ”
John Wiley and Sons, New York.
- Renfro, J. G. [1986],
 “ Computational Studies in the Optimisation of the Systems described by Differential/Algebraic Equations, ”
 Ph.D. Thesis, University of Houston.
- Robinson, S. M. [1972],
 “ A Quadratic Convergence Algorithm for General Nonlinear Programming Problems, ”
Math. Prog., 3, pp. 145-156.
- Robinson, S. M. [1974],
 “ Perturbed Kuhn-Tucker Points and Rates of Convergence for a Class of Non-linear Programming Algorithms, ”
Math. Prog., 7, pp. 1-16.
- Rustem, B. [1981],
 “ Projection Methods in Constrained and Applications to Optimal Policy Decisions, ”
Lecture Notes in Control and Information Science, (A. V. Balakrishnan and M. A. Thoma), Springer-Verlag, Berlin.
- Ruszczynski, A. [1984],
 “ A Recursive Quadratic Programming Algorithm for Constrained Stochastic Programming Problems, ”
Control and Cybernetics, 13, pp. 59-72.
- Sahba, M. [1987],
 “ Global Convergence Algorithm for Nonlinearly Constrained Optimisation Problems, ”,
J. Optim. Theor. Appl., 2, pp. 291-309.
- Sandgren, E. and Ragsdell, K. M. [1980],
 “ The Utility of Nonlinear Programming Algorithms : A Comparative Study – Part 1, and 2, ”
ASME J. Mech. Des, 102, pp. 540-551.

- Sargent, R. W. H. [1974],
 " Reduced Gradient and Projection Methods for Nonlinear Programming, ",
 in *Numerical Methods for Constrained Optimisation* (P. E. Gill and W.
 Murray, eds), Academic Press, London and New York, pp. 149-174.
- Sargent, R. W. H. [1978],
 " An Efficient Implementation of the Lemke Algorithm and Its Extension to
 Deal with Upper and Lower Bounds, "
Math. Prog. Study, 7, pp. 36-54.
- Sargent, R. W. H. [1980],
 " A Review of Optimisation Methods for Nonlinear Problems, "
 in *Computer Applications to Chemical Engineering*, (R. G. Squires and G.
 V. Reklaitis eds), ACS Symposium Series, 124, pp. 37-52.
- Sargent, R. W. H. [1982],
 " Recursive Quadratic Programming Algorithm and their Properties, "
 in *Numerical Analysis, 1981 Lecture Notes in Mathematics 909* (J. P. Hen-
 nart, ed), Springer-Verlag, Berlin, pp. 208-225.
- Sargent, R. W. H. [1985],
 private note.
- Sargent, R. W. H., and Murtagh, B. A. [1973],
 " Projection Methods for Nonlinear Programming, "
Math. Prog., 4, pp. 245-268.
- Schittkowski, K. [1980],
 " Nonlinear Programming Codes, "
Lecture Note in Economics and Mathematical System, 183, Springer-Verlag,
 Berlin.
- Schittkowski, K. [1981],
 " The Nonlinear Programming Method of Wilson, Han, and Powell with an
 Augmented Lagrangian Type Line Search Function, Part I - Convergence
 Analysis,"
Numer. Math., 38, pp. 83-114.
- Schittkowski, K. [1983],
 " On the Convergence of a Sequential Quadratic Programming Method with
 an Augmented Lagrangian Line Search Function, "
Math. Operationsforschung u. Statistik. Ser. Optimisation, 14, pp. 197-
 216.
- Schittkowski, K. [1984],
 " NLPQL : a Fortran Subroutine for Solving Constrained nonlinear program-
 ming problems, "
 Working Paper, University of Stuttgart.

- Schittkowski, K. (ed.) [1985],
 " Computational Mathematical Programming, "
 NATO ASI Series, Springer-Verlag, Berlin.
- Shacham, M. [1986],
 " Numerical Solution of Constrained Nonlinear Algebraic Equations, "
Intl. J. Numer. Meth. Engng., 23, pp. 1455-1481.
- Shanno, D. F. and Phua, K. H. [1978],
 " Matrix Conditioning and Nonlinear Optimisation, "
Math. Prog., 14, pp. 149-160.
- Skrobański, J. J. [1986],
 " Optimisation Subject to Nonlinear Constraints, "
 Ph.D. Thesis, University of London.
- SPEEDUP group [1988],
 SPEEDUP user's manual, Version 5.0, Department of Chemical Engineering
 and Chemical Technology, Imperial College, University of London.
- Stoer, J. [1985],
 " Principles of Sequential Quadratic Programming Methods for Solving Non-
 linear Programs, "
 In *Computational Mathematical Programming* , (K. Schittkowski, ed),
 NATO ASI series, 15, pp. 165-207.
- Stoer, J. and Tapia, R. A. [1987],
 " On the Characterisation of Q -Superlinear Convergence of Quasi-Newton
 Methods for Constrained Optimisation, "
Math. Comput., 49, pp. 581-584.
- Tapia, R. A. [1977],
 " Diagonalised Multiplier Methods and Quasi-Newton Methods for Constrained
 Optimisation, "
J. Optim. Theor. Appl., 22, pp. 135-194.
- Tseng, C. H. and Arora J. S. [1988]
 " On Implementation of Computational Algorithms for Optimal Design 1. Pre-
 liminary Investigation, "
 in *Intl. J. Numer. Meth. Engng.* 26, pp. 1365-1382.
- Tseng, C. H. and Arora J. S. [1988]
 " On Implementation of Computational Algorithms for Optimal Design 2. Ex-
 tensive Numerical Investigation, "
 in *Intl. J. Numer. Meth. Engng.* 26, pp. 1383-1402.
- Toint, Ph. L. [1986],
 " Global Convergence of the Partitioned BFGS Algorithm for Convex Partially
 Separable Optimisation, "
Math. Prog., 36, pp. 290-307.

- Toint, Ph. L. [1988],
 “ A View of Nonlinear Optimisation in a Large Number of Variables, ”
 in *Simulation and Optimisation of Large Systems* (A. Osiadacz, ed.), *IMA Conference Series*, 19, pp. 83-116.
- Tone, K. [1983],
 “ Revisions of Constraint Approximations in the Successive QP Method for Non-linear Programming Problems, ”
Math. Prog., 26, pp. 144-152.
- Vandergraft, J. S. [1983],
 “ Introduction to Numerical Computations, ”
 Academic Press, London and New York.
- Vardi, A. [1985],
 “ A Trust Region Algorithm for Equality Constrained Minimisation : Convergence Properties and Implementation, ”
SIAM J. Numer. Anal. 22, pp. 575-591.
- Waren, A. and Lasdon, L. S. [1979],
 “ The Status of Nonlinear Programming Software, ”
Operations Research, 27, pp. 431-456.
- Wilson, R. B [1963],
 “ A Simplicial Algorithm for Concave Programming, ”
 Ph.D. Thesis, Harvard University.
- Womersley R. S. and Fletcher R. [1986],
 “ An Algorithm for Composite Nonsmooth Optimisation Problems, ”
J. Optim. Theor. Appl., 48, pp. 493-523.
- Wright, S. J. [1986],
 “ Convergence of Projected Hessian Approximations in Quasi-Newton Methods for the Nonlinear Programming Problem ”
IMA J. Numer. Anal., 6, pp. 463-474.
- Yamashita, H. [1982],
 “ A Globally Convergent Constrained Quasi-Newton Method with an Augmented Lagrangian Type Penalty Function, ”
Math. Prog., 29, pp. 75-86.
- Yuan, Y. [1985 (a)],
 “ An Only 2-step Q-Superlinear Convergence Example for Some Algorithms that Use Reduced Hessian Approximations, ”
Math. Prog., 32, pp. 224-231.
- Yuan, Y. [1985 (b)],
 “ Some Theories and Algorithms in Nonlinear Programming, ”
 Ph.D. Thesis, Cambridge University.

Part II. Flowsheet Optimisation

- Adelman, A. and Stevens, W. F. [1972],
“ Process Optimisation by the Complex Method, ”
AIChE J. 18, pp. 20-24.
- Ahlgren, T. D. and Stevens, W. F. [1966],
“ Process Optimisation in the Presence of Error, ”
IEC Proc. Des. Dev., 5, pp. 291-297.
- Ballman, S. H. and Gaddy, J. L. [1977],
“ Optimisation of Methanol Process by Flowsheet Simulation, ”
IEC Proc. Des. Dev., 16, pp. 337-340.
- Banks, P [1984],
“ GENESIS, Generalised Simulation System User's Guide - Version 6.0 ”
Engineering and Technical Centre, British Petroleum International Limited.
- Berna, T. J., Locke, M. H. and Westerberg, A. W. [1980],
“ A New Approach to Optimisation of Chemical Processes, ”
AIChE J., 26, pp. 37-43.
- Biegler, L. T. [1983],
“ Simultaneous Modular Simulation and Optimisation, ”
in *Foundations of Computer Aided Process Design - II*, Snowmass, Colorado
(A. W. Westerberg and H. H. Chien, eds), CACHE Corp., Austin, Texas,
pp. 369-409.
- Biegler, L. T. [1985],
“ Improved Infeasible Path Optimisation for Sequential Modular Simulators –
I : The Interface, ”
Comput. Chem. Engng., 9, pp. 245-256.
- Biegler, L. T. [1988],
“ On the Simultaneous Solution and Optimisation of Large Scale Engineering
Systems, ”
Comput. Chem. Engng., 12, pp. 357-369.
- Biegler, L. T. and Hughes, R. R. [1981],
“ Approximation Programming of Chemical Processes with Q/LAP, ”
Chem. Engng. Prog., 77, April, pp. 76-83.
- Biegler, L. T. and Hughes, R. R. [1982],
“ Infeasible Path Optimisation with Sequential Modular Simulators, ”
AIChE J., 28, pp. 994-1002.
- Biegler, L. T. and Hughes, R. R. [1983],
“ Process Optimisation : A Comparative Case Study, ”
Comput. Chem. Engng., 7, pp. 645-661.

- Biegler, L. T. and Hughes, R. R. [1985],
 " Feasible Path Optimisation with Sequential Modular Simulators, " *Comput. Chem. Engng.*, 9, pp. 379-394.
- Biegler, L. T. and Cuthrell, J. E. [1985],
 " Improved Infeasible Path Optimisation for Sequential Modular Simulators – II : The Optimisation Algorithm, " *Comput. Chem. Engng.*, 9, pp. 257-267.
- Biegler, L. T. and Grossmann, I. E. [1985],
 " Strategies for the Optimisation of Chemical Processes, " *Reviews in Chem. Engng.*, 3, pp 1-48.
- Biegler, L. T., Grossmann, I. E. and Westerberg, A. W. [1985 (a)],
 " A Note on Approximation Techniques Used for Process Optimisation, " *Comput. Chem. Engng.*, 9, pp. 201-206.
- Bracken, J. and McCormick G. P. [1968],
 " Selected Applications of Nonlinear Programming, " John Wiley and Sons, New York and Toronto.
- Cavett, R. H. [1963],
 " Application of Numerical Methods to the Convergence of Simulated Processes Involving Recycle Loops, " *Am. Pet. Inst., Repr. No. 04-63.*
- Chan, W. K. and Prince, R. G. H. [1986],
 " Application of the Chain Rule of Differentiation to Sequential Modular Flowsheet Optimisation, " *Comput. Chem. Engng.*, 10, pp. 223-240.
- Chen, H. S. and Stadtherr, M. A. [1984],
 " Enhancements of the Han-Powell Method for Successive Quadratic Programming, " *Comput. Chem. Engng.*, 3/4, pp.229-234.
- Chen, H. S. and Stadtherr, M. A. [1985 (a)],
 " A Simultaneous Modular Approach to Process Flowsheeting and Optimisation. Part I : Theory and Implementation, " *AIChE J.*, 31, pp. 1843-1856.
- Chen, H. S. and Stadtherr M. A. [1985 (b)],
 " A Simultaneous Modular Approach to Process Flowsheeting and Optimisation. Part III : Performance on Optimisation Problems, " *AIChE J.*, 31, pp. 1868-1881.
- Christensen, J. M. [1970],
 " The Structuring of Process Optimisation, " *AIChE J.*, 16, 177-184.

- Di bella, C. W. and Stevens, W. F. [1965],
 " Process Optimisation by Nonlinear Programming, "
IEC Proc. Des. Dev., 4, pp. 16-20.
- Doering, F. J. and Gaddy, J. L. [1980],
 " Optimisation of the Sufuric Acid Process with a Flowsheet Simulator, "
Comput. Chem. Engng., 4, pp. 113-122.
- Doherty, M. F., Malone, M. F., Marquez, F. E., Davidson, D. L. and Douglas, J. M. [1982],
 " Estimating Bounds for Process Optimisation Problems. An Analogy to Rules of Thumb and Structural Modifications "
IEC Fundam., 21, pp. 289-298.
- De Leeuw Den Bouter, J.A., Perris, F. A., Rippin, D. W. T. and Theunissen, F. A. P. [1983],
 " Sample Problems for Flowsheeting Programs, "
Comput. Chem. Engng., 7, pp. 65-71.
- Edgar, T.F. and Himmelblau, D. M. [1988],
 " Optimisation of Chemical Processes, "
 McGraw-Hill Company, New York.
- Evans, L. B. [1981],
 " Advances in Process Flowsheeting System, "
 in *Foundations of Computer-Aided Chemical Process Design*, Vol. I, (R. S. H. Mah and W. D. Seider, eds), pp. 425-469.
- Evans, L. B. [1982],
 " Process Flowsheeting : a State-of-the-Art Review, "
 Presented at *CHEMCOMP 82* , Antwerp, Belgium, May.
- Evans, L. B., Trevino-Lozano, R. A., Britt, H. I. and Boston, J. F. [1985],
 " Simultaneous Modular Process Simulation and Optimisation, "
PSE '85, EFCE Publication Series, 40, pp. 25-36.
- Findley, M. E. [1974],
 " Modified One-at-a-Time Optimisation, "
AIChE J., 20, pp. 1154-1160.
- Friedman, P and Pinder K. L. [1972],
 " Optimisation of a Simulation Model of a Chemical Plant, "
IEC Proc. Des. Dev., 11, pp. 512-520.
- Gaines, L. D., and Gaddy, J. L. [1976],
 " Process Optimisation by Flow Sheet Simulation, "
IEC Proc. Des. Dev., 15, pp. 206-211.

- Gallier, P. W., and Kisala, T. P. [1987],
 " Process Optimisation by Simulation, "
Chem. Engng. Prog., 83, Aug. pp. 60-66.
- Ganesh, N. and Biegler, L. T. [1987 (a)],
 " A Robust Technique for Process Flowsheet Optimisation Using Simplified Model Approximations, "
Comput. Chem. Engng., 11, pp. 553-565.
- Ganesh, N. and Biegler, L. T. [1987 (b)],
 " A Reduced Hessian Strategy for Sensitivity Analysis of Optimal Flowsheets, "
AIChE J., 33, pp. 282-296.
- Goulcher, R. and Long, C. J. J. [1978],
 " The Solution of Steady-State Chemical Engineering Optimisation Problems Using a Random Search Algorithm, "
Comput. Chem. Engng., 2, pp. 33-36.
- Grauer, M., Gruhn, G., and Pollmer, L. [1979],
 " Optimisation of a Complex Plant by a GRG Algorithm, "
Comput. Chem. Engng., 3, pp. 597-602.
- Himmelblau, D. M. [1963],
 " Process Optimisation by Search Techniques, "
IEC Proc. Des. Dev., 2, pp. 296-300.
- Himmelblau, D. M. [1979],
 " A Nonlinear Programming Test Problem, "
SIGMAP, Bulletin, 27, July, pp. 34-44.
- Hoza, M. and Stadherr, M. A. [1987],
 " Successive Quadratic Programming Methods for Chemical Process Optimisation, "
 AIChE Annual Meeting, New York, Nov.
- Hutchison, H. P., Jackson, D. J. and Morton, W. [1986 (a)],
 " The Development of an Equation Oriented Flowsheet Simulation and Optimisation Package - I. The QUASILIN Program, "
Comput. Chem. Engng., 10, pp. 19-29.
- Hutchison, H. P., Jackson, D. J. and Morton, W. [1986 (b)],
 " The Development of an Equation Oriented Flowsheet Simulation and Optimisation Package - II. Examples and Results, "
Comput. Chem. Engng., 10, pp. 31-47.
- Jirapongphan, S. [1980],
 " Simultaneous Modular Convergence Concept in Process Flowsheet Optimisation, "
 ScD Thesis, MIT, Cambridge, Massachusetts.

- Johns, W. R., and Vadhwana [1985],
 " A dual-levels Flowsheeting System, "
PSE '85, EFCE Publication Series, 40, pp. 37-54.
- Jung, B. S., Mirosh, W., and Ray, W. H. [1971],
 " A Study of Large Scale Optimisation Techniques, "
Can. J. Chem. Engng. 49, pp. 845-852.
- Kaijaluoto, P. I. S. [1984],
 " Process Optimisation by Flowsheet Simulation, "
 Ph. D. Thesis, University of Cambridge.
- Kaijaluoto, P. I. S. and Morton W. [1987],
 " Numerical Calculation of Derivatives in Process Optimisation, "
Proceedings CEF 87, Giardini Naxos, Sicily, April, pp. 779-783.
- Kisala, T. P., Boston, J. F., Britt, H. I. and Evans, L. B. [1986],
 " An Infeasible Path Hybrid Method for Sequential Modular Optimisation, "
 Paper presented at *Word Congress III of Chemical Engineering in Computer Aided Design*, Tokyo, pp. 603-606.
- Kisala, T. P., Trevino-Lozano, Boston, J. F., Britt, H.I. and Evans, L.B. [1987],
 " Sequential Modular and Simultaneous Modular Strategies for Process Flow-sheet Optimisation, "
Comput. Chem. Engng., 11, pp. 567-579.
- Lang, Y. D. and Biegler, L. T. [1987],
 " A Unified Algorithm for Flowsheet Optimisation, "
Comput. Chem. Engng., 11, pp. 143-158.
- Locke, M. H. and Westerberg, A. W. [1983 (a)],
 " The ASCEND - II System - a Flowsheeting Application of a Successive Quadratic Programming Methodology, "
Comput. Chem. Engng., 7, pp. 615-630.
- Locke, M. H., Westerberg, A. W. and Edahl, R. H. [1983 (b)],
 " Improved Successive Quadratic Programming Algorithm for Engineering Design Problems, "
AIChE J., 29, pp. 871-874.
- Lucia, A and Kumar, A. [1986],
 " Distillation Optimisation, "
 Paper 55f at annual AIChE Meeting in Miami Beach Nov.
- Luus, R. [1975],
 " Optimisation of Multistage Recycle Systems by Direct Search, "
Can. J. Chem. Engng., 53, pp. 217-220.

- Luus, R. and Jaakola, T. H. I. [1973],
 " Optimisation of Nonlinear Functions Subject to Equality Constraints. Judicious Use of Elementary Calculus and Random Numbers, "
IEC Proc. Des. Dev., 12, pp. 380-383.
- Luus, R. and Jaakola, T. H. I. [1974],
 " A note on the Application of Nonlinear Programming to Chemical Programming Optimisation, "
Operations Research, 22, pp. 415-418.
- Macchietto, S., Maduabueke, G. I. and Szczepanski, R. [1988],
 " Efficient Implementation of VLE Procedures in Equation-Oriented Simulators, "
AIChE J., 34, pp. 955-963.
- Motard, R. L. and Lee, H. M. [1971],
 " CHESS, Chemical Engineering User's Guide, "
 3rd edn., University of Houston, Texas.
- Newell, R. B. [1980],
 " A Comparative Study of Model and Goal Coordination in the Multilevel Optimisation of a Double Effect Evaporator, "
Can. J. Chem. Engng., 58, pp.275-278.
- Ostrovsky, G. M., Ostrovsky, M. C., and Berezhinsky, T. A. [1988],
 " Optimisation of Chemical Plants with Recycles, "
Comput. Chem. Engng., 12, pp. 289-296.
- Pai, C. C. D. and Hughes, R. R. [1987],
 " Strategies for Formulating and Solving Two-Stage Problems for Process Design Under Uncertainty, "
Comput. Chem. Engng., 11, pp. 695-706.
- Pantelides, C. C. [1988 (a)],
 " SPEEDUP - Recent Advances in Process Simulation, "
Comput. Chem. Engng., 12, pp. 7, 745-755.
- Pantelides, C. C. [1988 (b)],
 " Symbolic and Numerical Techniques for the Solution of Large System of Non-linear Algebraic Equations, "
 Ph.D. Thesis, University of London.
- Parker, A. L. and Hughes, R. R. [1981 (a)],
 " Approximation Programming of Chemical Processes - 1 : Optimisation of FLOWTRAN® Models, "
Comput. Chem. Engng., 5, pp. 123-133.
- Parker, A. L. and Hughes, R. R. [1981 (b)],
 " Approximation Programming of Chemical Processes - 2 : Computational Difficulties, "
Comput. Chem. Engng., 5, pp. 135-142.

Payne, R. E. [1958]

- “ Alkylation - What You Should Know About This Process, ”
Petroleum Refiner, 37, pp. 316-329.

Perkins, J. D. [1983],

- “ Equation-Oriented Flowsheeting, ”
in *Foundations of Computer Aided Process Design - II*, Snowmass, Colorado
(A. W. Westerberg and H. H. Chien, eds), CACHE Corp., Austin, Texas,
pp. 309-365.

Rangaiah, G. P. [1985],

- “ Studies in Constrained Optimisation of Chemical Process Problems, ”
Comput. Chem. Engng., 9, pp. 395-404.

Richard, M. J. and Pike, R. W. [1987],

- “ Optimisation of Contact Sulfuric Acid Process Using the *CHEMSHARE*®
Flowsheeting System ”
Presented at 1987 AIChE Annual Meeting - New York, N.Y. Nov.

Rijckaert, M. J. and Martens, X. M. [1978],

- “ Analysis and Optimisation of Williams-Otto Process by Geometric Program-
ming, ”
AIChE J., 20, pp. 742-747.

Rudd, D. F. and Watson, C. C. [1968],

- “ Strategy of Process Engineering, ”
John Wiley and Sons, New York and Toronto.

Sarma, P. V. L. N. and Reklaitis, G. V. [1982]

- “ Optimisation of a Complex Chemical Process Using an Equation Oriented
Model, ”
Math. Prog. Study, 20, pp. 113-160.

Sauer, R. N., Colville, A. R. and Burwick, C. W. [1964],

- “ Computer Points the Way to More Profit, ”
Hydro. Proc. Petrol. Refiner, 43, pp. 84-92.

Shacham, M., [1986],

- “ Numerical Solution of Constrained Nonlinear Algebraic Equations, ”
Intl. J. Meth. Engng., 23, pp. 1455-1481.

Shacham, M., Macchietto, S., Stutzman, L. F., Babcock, P. [1982],

- “ Equation oriented approach to flowsheeting, ”
Comput. Chem. Engng., 6, pp. 79-95.

Shivaram, S. and Biegler, L. T. [1983],

- “ Improved Infeasible Path Methods for Sequential Modular Optimisation, ”
in *Proceedings, 3rd Congr s international, Paris, EFCE Publication Series*,
27, C-157, pp. 1-6.

- Sriram, M. and Stevens, W. F. [1973],
 " An Example of the Application of Nonlinear Programming to Chemical Process Optimisation, "
Operations Research 21, pp. 296-304.
- Stadtherr, M. A. and Chen, H. S. [1983],
 " Strategies for Simultaneous Modular Flowsheeting and Optimisation, "
 in *Foundations of Computer Aided Process Design - II*, Snowmass, Colorado (A. W. Westerberg and H. H. Chien, eds), *CACHE* Corp., Austin, Texas, pp. 863-887.
- Vasantharaja, S. and Biegler, L. T. [1987],
 " Large Scale Decomposition for Successive Quadratic Programming, "
 Present at 1987 AIChE Annual Meeting - New York, Nov.
- Vinante, C. and Valladares, E. [1985]
 " Application of the Method of Multipliers to the Optimisation of Chemical Processes, "
Comput. Chem. Engng., 9, pp.83-87.
- Wang B.C. and Luus R. [1978],
 " Reliability of Optimisation Procedures for Obtaining Global Optimum, "
AIChE J., 24, pp. 619-626.
- Westerberg, A. W. [1981],
 " Optimisation in Computer-Aided Design, "
 in *Foundations of Computer Aided Process Design, Vol I*, (R. S. H. Mah and W. D. Seider,) *AIChE*, New York, pp. 149-183.
- Westerberg, A. W. and Benjamin, D. R. [1985],
 " Thoughts on a Future Equation-Oriented Flowsheeting System, "
Comput. Chem. Engng., 9, pp. 517-526.
- Williams, T. J. and Otto, R. E. [1960],
 " A Generalised Chemical Processing Model for the Investigation of Computer Control, "
AIEE Transactions Part 1, Communications and Electronics, 73, pp. 458-473.
- Zudekevitch, D. and Joffe J. [1970],
 " Correlation and Prediction of Vapor-Liquid Equilibria with the Redlich-Kwong Equation of State, "
AIChE J., 16, pp. 113-119.