A Domain Decomposition Non-Intrusive Reduced Order Model for Turbulent Flows

D. Xiao[a,b,c,*], C.E. Heaney[a], F. Fang[a,b], L. Mottet[a,d], R. Hu[a], D. A. Bistrian[e], E. Aristodemou[f,a], I.M. Navon[g], C.C. Pain[a,b]

[a]*Applied Modelling and Computation Group, Department of Earth Science and Engineering, Imperial College London, Prince Consort Road, London, SW7 2BP, UK*
[b]*Data Assimilation Lab, Data Science Institute, Imperial College London, Prince Consort Road, London, SW7 2BP, UK*
[c]*ZCCE College of Engineering, Swansea University, Bay Campus, Fabian Way, Swansea SA1 8EN, UK*
[d]*Department of Architecture, University of Cambridge, 1-5 Scroope Terrace, Trumpington Street, Cambridge, CB2 IPX, UK*
[e]*Politehnica University of Timisoara, Hunedoara, 331128, Romania*
[f]*School of Engineering, London South Bank University, London, UK*
[g]*Department of Scientific Computing, Florida State University, Tallahassee, FL, 32306-4120, USA*

Abstract

In this paper, a new Domain Decomposition Non-Intrusive Reduced Order Model (DDNIROM) is developed for turbulent flows. The method works by partitioning the computational domain into a number of subdomains in such a way that the summation of weights associated with the finite element nodes within each subdomain is approximately equal, and the communication between subdomains is minimised. With suitably chosen weights, it is expected that there will be approximately equal accuracy associated with each subdomain. This accuracy is maximised by allowing the partitioning to occur through areas of the domain that have relatively little flow activity, which, in this case, is characterised by the pointwise maximum Reynolds stresses.

A Gaussian Process Regression (GPR) machine learning method is used to construct a set of local approximation functions (hypersurfaces) for each subdomain. Each local hypersurface represents not only the fluid dynamics over the subdomain it belongs to, but also the interactions of the

*Corresponding author
Email address: dunhui.xiao@swansea.ac.uk (D. Xiao)

flow dynamics with the surrounding subdomains. Thus, in this way, the surrounding subdomains may be viewed as providing boundary conditions for the current subdomain.

We consider a specific example of turbulent air flow within an urban neighbourhood at a test site in London and demonstrate the effectiveness of the proposed DDNIROM.

Keywords: Non-Intrusive Reduced Order Modelling, Domain Decomposition, Machine Learning, Gaussian Process Regression, Urban flows, Turbulent flows, Finite Element Method.

1. Introduction

The Reduced Order Model (ROM) approach is a very powerful numerical technique that can be extremely useful for real-time decision making and operational modelling as it enables predictions to be obtained in a time that is many orders of magnitude faster than conventional computational methods. Moreover, ROM solutions can be obtained within fractions of a second as opposed to days or weeks with conventional numerical methods. It is particularly popular in engineering and has been successfully applied in various areas of research such as fluid dynamics [1, 2, 3, 4, 5, 6], data assimilation [7], fracture modelling [8], host-parasitoid system [9], air pollution [10], aerospace design [11], and haemodynamics [12].

ROMs can be classified in two groups based on the modification (or not) of the original source code; Non-Intrusive ROM (NIROM) refers to the ROM when the original source code is *not* modified, whilst intrusive ROM (IROM) refers to the ROM when the original source code is modified. NIROM has become particularly popular over the last decade as it has been shown to avoid instabilities and to overcome non-linear inefficiency issues encountered in IROM [13, 14, 15, 16, 17, 18]. It also has the advantages over IROM of being easier to implement, modify and extend to use with complicated problems, as it does not rely on having access to the source code of the original model.

Xiao *et al.* proposed a number of NIROMs [19, 20, 21, 22] which have been applied within the field of fluid dynamics, fluid-structure interactions [23] and multiphase porous media flow [24].

However, despite its computational speed, reduced order modelling has certain difficulties. One of these is that it is hard to capture the details of a small region within a much larger computational domain e.g. capturing details of flow around a building in a town or a city. In order to overcome such deficiencies, methods which divide the domain into a number of subdomains have been applied to reduced order modelling. This idea of Domain Decomposition (DD) goes back to the work presented by Schwarz [25] on overlapping domain decomposition. The idea was extended to non-overlapping DD in the 1960s [26, 27, 28]. More recently there has been a lot of research carried out into combining DD methods with solvers, efficient preconditioners and parallel computing technologies, see for example [29, 30, 31]. For our purposes we exploit the basic idea of domain decomposition as our aim is to come up with an effective way of splitting the domain. Pain *et al.* [32] also presented a domain decomposition method based on neural networks. Lucia *et al.* [33, 34] firstly introduced the subdomain idea into ROM for tracking a moving shock wave, whilst other applications of the subdomain approach into ROM include the work of Baiges *et al.* [35], Kerfriden *et al.* [36], Amsallem *et al.* [37] and Chaturantabut [38]. Antil *et al.* [39, 40] also used a subdomain idea to construct balanced truncation based ROMs. Xiao *et al.* [41] develop a Domain Decomposition NIROM (DDNIROM) which splits the computational domain into subdomains geometrically and calculates a local approximation to the governing equations within each subdomain. The DDNIROM is solved iteratively in order to pass information from one subdomain to another. DDNIROM can then be applied to large-scale computational problems for capturing complex flows in detail e.g. turbulent flows within the atmospheric boundary layer, flows with shocks, or flows at a city scale. DDNIROM has the advantage of allowing the construction

of a set of local basis functions based on details of local forward flow solutions. This implies that the subdomains with less complicated physics can be modelled using fewer basis functions, whilst the others with more complicated flow patterns may be represented with a larger number of basis functions to enable capturing of the detailed flow and the turbulent eddies.

However, none of these methods optimise the partitioning for the ROM according to the physics of the problem. Therefore, we extend our previous DDNIROM approach [41] to do just this. What distinguishes the current work from previous work on DD with ROM is that it carefully chooses the subdomains so that the accuracy of the ROM is improved. This is achieved through a weighting constraint which aims to achieve an equal accuracy in each subdomain as well as to minimise the dynamic activity between subdomains. The domain is decomposed so that the work load is balanced equally across each subdomain and the communication load (dynamic activity) between subdomains is minimised. Choosing weights that represent both the communication load and the load on each subdomain allows the subdomains to be more isolated (due to minimising the communication). The subdomains will not be independent (completely isolated) but perhaps more weakly linked than a random domain decomposition would produce. This careful choice of subdomains, and their boundaries, is the key to the success of the proposed method. This approach can be used for both intrusive and non-intrusive ROMs.

The main advantages of the DDNIROM approach can be summarised as follows: (a) it may be used for large-scale (km scale) problems with many degrees of freedom; (b) it is characterised by enhanced accuracy for the local hypersurface formations; (c) smaller Singular Value Decompositions (SVDs) within each subdomain as opposed to the larger SVDs required for the entire domain (SVDs are used to generate the basis functions); (d) Gaussian Process Regression (GPR), used in DDNIROM, being lower dimensional surface fits than those used in NIROM, may be more

accurate because they surface fit in lower dimensions; (e) the potential ability to form the NIROMs for each subdomain independently by running models across a subdomain or group of subdomains independently. This makes possible the large-scale modelling of, for example, an entire city, even when the forward model is based on expensive computational methods.

The structure of the present paper is as follows: Section 2 presents the governing equations of fluid flow problems. Section 3 describes the theory of the NIROMs. Section 4 derives the domain decomposition based NIROM (DDNIROM) approach. Section 5 demonstrates the performance of the DDNIROM for a specific turbulent flow test case: urban flows. Finally, in section 6, the conclusions are presented.

2. Governing equations

The three-dimensional continuity and non-hydrostatic Navier-Stokes equations which describe the conservation of mass and momentum of a fluid are given by

$$\nabla \cdot \boldsymbol{u} = 0, \tag{1}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \nabla \cdot \boldsymbol{\tau}, \tag{2}$$

where $\boldsymbol{u} \equiv (u, v, w)^T$ is the velocity vector, t is the time, $p = \tilde{p}/\rho_0$ is the normalised pressure ($\tilde{p}$ being the pressure and ρ_0 being the constant reference density), and $\boldsymbol{\tau}$ denotes the stress tensor. An anisotropic Smagorinsky model [42, 43] is used to calculate the large eddy scale (LES) sub-grid scale viscosity which is included in the stress tensor. We apply filtering to the Navier-Stokes equations to model behaviour on the fine scale, for instance, fluctuations that occur on scales smaller than the grid scale.

3. Non-Intrusive Reduced Order Modelling

This section describes the construction and use of NIROM for three dimensional urban flows using Proper Orthogonal Decomposition (POD) and the GPR machine learning method.

Typically there are two stages linked to constructing a reduced order model: off-line (training) and on-line. The off-line stage, which is expected to be computationally expensive, involves finding the POD basis functions and approximating the governing equations in reduced space. The on-line stage is much less computationally intensive due to the reduced dimension of the model. During this stage the governing equations in reduced space are solved, which, in this case, amounts to mere function evaluations. The computational cost of these stages varies, depending on the application: we will give specific figures in the results section.

NIROM can be deployed in two different ways, either to predict the flow at times beyond those used in the training of the model, or to predict the flow for parameters not used in the training. To explore the first case, we previously developed a NIROM for turbulent urban flows [44]. We ran the high-fidelity model until the flow statistics had reached a steady state and we showed that, when predicting beyond the time of the training period, the NIROM retained the same flow statistics as the high-fidelity model. In this paper we also run the DDNIROM beyond the training time period and monitor the flow statistics. In the future we aim to develop parametrised NIROMs.

In this section we first describe how POD is used to obtain basis functions from the high-fidelity LES model. We give a brief description of GPR and then explain how the NIROM is constructed by training a neural network using GPR with data from the high-fidelity model.

3.1. Proper Orthogonal Decomposition

POD, also known as principal component analysis (PCA), or empirical orthogonal functions (EOF) in oceanography, is one of the most widely used model reduction techniques. The aim of POD is to find a subspace that is able to approximate a given data set in an optimal least-squares sense, i.e. the method is based on finding a subspace from a full space such that the error of projecting from the full space onto the subspace is minimised [45].

In order to sample the high dimensional space we use the Method of Snapshots [46]. Snapshots are solutions of the discretised governing equations at certain time levels and are vectors of length N, where N is the number of nodes. If we have taken $\mathcal{N}_s$ snapshots of the velocities, we construct snapshot matrices which have the following form

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{u}^1 & \boldsymbol{u}^2 \cdots & \boldsymbol{u}^{\mathcal{N}_s} \end{bmatrix} \tag{3}$$

where the discretised solutions $\boldsymbol{u}^i$ are vectors containing values of the streamwise velocity at the nodes for the ith snapshot. An SVD is then applied to $\boldsymbol{S}$ which results in three matrices,

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{4}$$

where $\boldsymbol{U}$ is an N by N matrix containing the left singular vectors, $\boldsymbol{V}$ is an $\mathcal{N}_s$ by $\mathcal{N}_s$ matrix containing the right singular vectors and $\boldsymbol{\Sigma}$ is an N by $\mathcal{N}_s$ matrix which contains the singular values on the leading diagonal. The columns of matrix $\boldsymbol{U}$ contain the POD basis functions or modes. The singular values give an indication as to how important each basis function is, and the square of each singular value represents the amount of energy captured by the corresponding basis function,

so if the singular value is low its corresponding basis function can be discarded. Based on this, a common criterion for choosing how many POD basis functions to include is as follows: for a tolerance $\eta \lesssim 1$, find the smallest integer m such that

$$\frac{\sum_{i=1}^{m} \sigma_i^2}{\sum_{i=1}^{\mathcal{N}_s} \sigma_i^2} \geqslant \eta, \tag{5}$$

where σ_i represents the ith singular value and $m \leqslant \mathcal{N}_s$ (although the method works well when $m \ll \mathcal{N}_s$). Considering the criterion, for example, if η takes the value 0.99, this means that 99% of the energy of the snapshots would be captured by the m leading POD basis functions.

We find POD basis functions and singular values for each of the three velocity components in turn, because, in this work, the reduced order model will be formulated in terms of velocity alone. As the problem is incompressible we can eliminate pressure from the NIROM formulation and pressure POD basis functions are not required.

In this paper we obtain the snapshots of velocity by solving the discretised governing equations. After performing the SVD we have $\mathcal{N}_s$ possible POD basis functions. After truncation, based on the energy criterion given in equation (5), we will have a set of m POD basis functions which we represent as $\{\boldsymbol{\phi}_j\}_{j=1}^{m}$. An approximation to the velocity can be expressed as a linear combination of the POD basis functions,

$$\sum_{j=1}^{m} \alpha_j \boldsymbol{\phi}_j \tag{6}$$

where α_j represents the POD coefficients.

3.2. Gaussian Process Regression

Gaussian Process Regression (GPR) (see [47]) uses a series of Gaussian-shaped basis functions to provide the surface representations used here to form the NIROM. We note that GPR produces smooth hypersurfaces which we believe regularises the solution to some extent, thereby increasing the accuracy of the results, see [48]. GPR often does not require as much training data as feed forward Neural Networks, and thus, because we have only 500 data points, we use GPR rather than feed forward Neural Networks. We adopt the GPR from within the open-source scikit-learn library [49] to form our DDNIROMs. Especially for small data sets, GPR can be susceptible to over-fitting, where the model learns the training data well but predicts poorly for 'unseen' data. To reduce this effect, leave-p-out cross-validation is used, in which 80% of the data is used for training and the remaining 20% is used to judge the performance of the model ('validation'). We allow the width of the Gaussian radial basis functions to be in the range $[10^{-2}, 10^{2}]$ and the variance to be in the range $[10^{-3}, 10^{3}]$. Both quantities are optimised by the GPR algorithm.

3.3. Training the Non-Intrusive Reduced Order Model

After obtaining the basis functions as described in section 3.1, the neural network based on GPR is trained with the snapshot solutions from the LES simulations that have been projected onto the reduced space by the POD basis functions.

The training process results in a set of hypersurfaces which will represent the governing equations, geometry, boundary conditions and model parameters in reduced space. The hypersurfaces are represented by a function f_j for each POD coefficient, which maps the POD coefficients from

one time level to the next, for example

$$\alpha_j^k = f_j(\boldsymbol{\alpha}^{k-1}) = f_j(\alpha_1^{k-1}, \alpha_2^{k-1}, \ldots, \alpha_m^{k-1}), \qquad \forall k \in \{1, 2, \ldots, \mathcal{N}_s\}\,. \tag{7}$$

For each POD coefficient, the neural network is therefore trained with the following inputs and outputs

$$\text{input:} \qquad \boldsymbol{\alpha}^{k-1} = \left(\alpha_1^{k-1}, \alpha_2^{k-1}, \ldots, \alpha_m^{k-1}\right) \tag{8}$$

$$\text{output:} \qquad \alpha_j^k\,, \tag{9}$$

for all $k \in \{1, 2, \ldots, \mathcal{N}_s\}$. Through this training process we obtain a set of hypersurfaces (functions $\{f_j\}_{j=1}^m$) which concludes the off-line stage, the main steps of which can be summarised as follows:

- run the high-fidelity LES model over the time period $[0, T]$ and generate the snapshots matrix $\boldsymbol{S}$;
- calculate a set of POD basis functions by performing the SVD of the snapshots matrix $\boldsymbol{S}$;
- calculate the POD coefficients corresponding to each snapshot;
- train the GPR model using the POD coefficients of the snapshots, see equations (8) and (9).

3.4. Using NIROM

During the online stage, the governing equations are approximated by the hypersurfaces $\{f_j\}_{j=1}^m$ which are treated as response functions, allowing the NIROM solutions at the current time level to

be predicted given those at a previous time level

$$\alpha_j(t + \Delta t) = f_j(\boldsymbol{\alpha}(t)) \quad \forall j \in \{1, 2, \ldots, m\}\,. \tag{10}$$

We remark that when using the NIROM to predict, the time step, Δt, will necessarily coincide with the time between two successive snapshots. In this work we predict for times beyond those used in training the model, i.e. for times $t > T$. The procedure of using the NIROM is summarised in Figure 1.

4. Domain Decomposition NIROM

4.1. Domain-decomposition-strategy

This section describes a novel method that combines Domain Decomposition and NIROM (DDNIROM). It builds on a previous paper [41] in which DDNIROM was first developed. There, the decomposition was calculated simply by splitting the domain geometrically. The domain decomposition method used in the current paper calculates a balanced k-way partitioning that minimises the total communication volume [50] and distributes the computational work load as equally as possible over the subdomains. An important aspect is the overall accuracy of the DDNIROM which will be enhanced if the activity level between the domains is small measured by the maximum value in the Reynolds stress tensor at each point in the domain. In addition, although achieving an equal loading between subdomains may seem a secondary issue, as one always has the option of using variable numbers of POD basis functions within each subdomain, balancing the load within each subdomain may give us a suitable aim for both edge cut minimisation and load balancing. One can, of course, try to form iteratively an approximate weight set while finding the

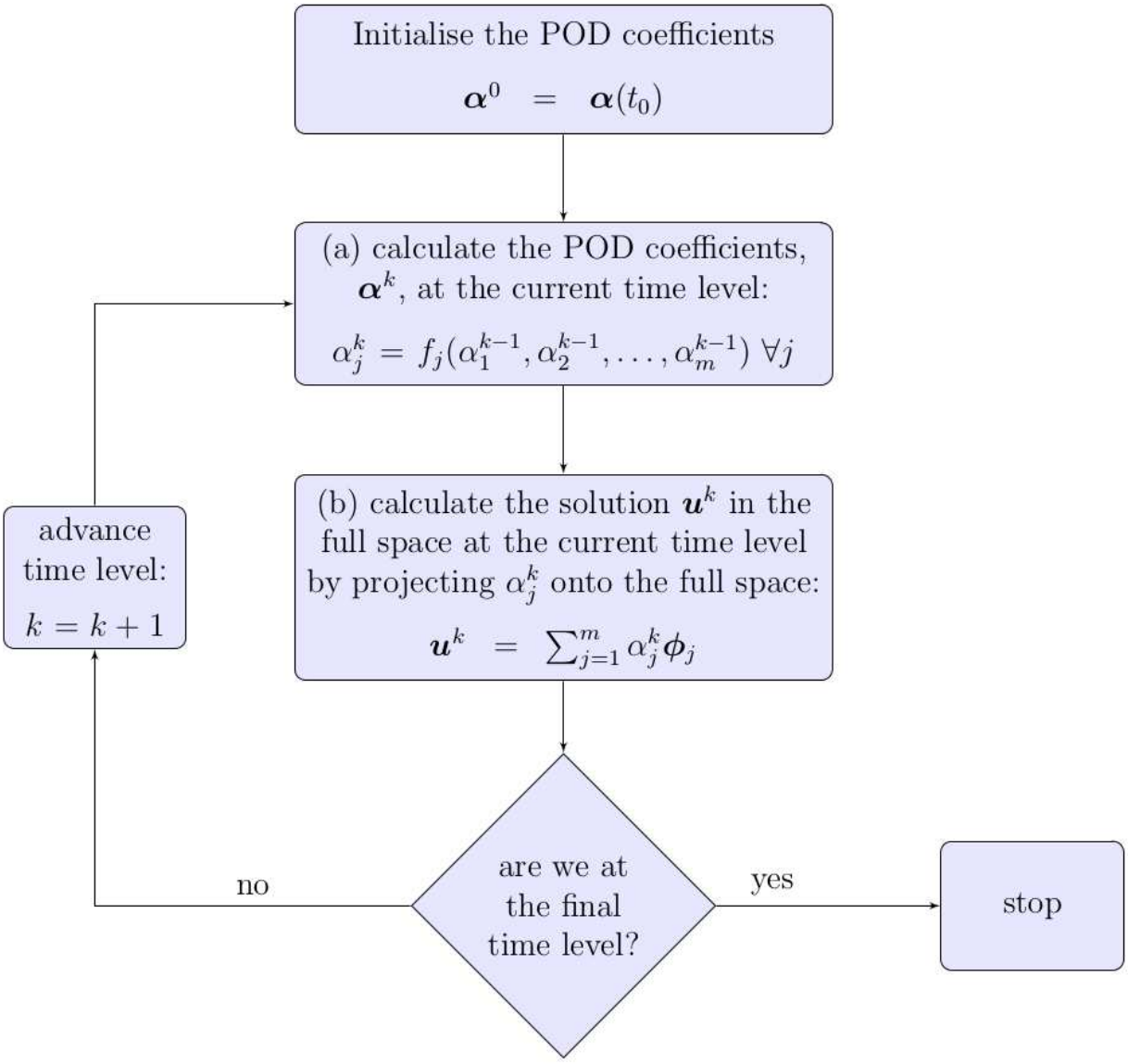


Figure 1: On-line NIROM calculation

subdomains, by iteratively adjusting the weights until there is a balance in accuracy within each subdomain. We have not pursued this here as it could be complex and computationally expensive.

Initially, weights are assigned to each vertex or node in the FE mesh and then the domain decomposition algorithm attempts to partition the domain into a number of subdomains, so that the sum of the weights in the interface regions between subdomains is minimised and the sums of the weights in each of the subdomains are approximately equal. This has the effect of minimising the communication and activity between subdomains and balancing the computational load across the subdomains. We use the ParMETIS library [51] to decompose the computational domain.

We quantify the communication load as follows. Consider a graph $G = (V, E)$ with a node set V and an edge set E. Let $\mathcal{P}$ be a vector of size $|V|$ and let $\mathcal{P}_i$ store the subdomain number to which vertex i belongs. For each vertex $i \in V$ let $\mathcal{A}_i$ be the number of subdomains other than $\mathcal{P}_i$ that the vertices adjacent to i belong. The total communication volume C of this partition is defined as:

$$C = \sum_{i \in V} w_i \mathcal{A}_i \,, \tag{11}$$

see [51]. Equation (11) corresponds to the total communication volume incurred by the partition because information associated with each interface vertex i needs to be sent to all of its $\mathcal{A}_i$ subdomains.

In this work, we investigate three ways of calculating the vertex weights: uniform weights; weights based on the maximum value of the (nodal) Reynolds stresses, and weights based on an approximation of the singular value of the largest mode discarded in the truncation of the SVD. In

the following, the x-component of the Reynolds Stresses defined as

$$\overline{u'u'} = \frac{1}{T_2 - T_1} \int_{T_1}^{T_2} (u - \overline{u})^2 dt\,, \tag{12}$$

for a time interval of $[T_1, T_2]$, where u denotes the pointwise x-component (streamwise direction) of velocity and $\overline{u}$ denotes a time-averaged quantity.

We first consider basing the nodal weights, w_i, on the maximum Reynolds stress to attempt to minimise the activity at subdomain interfaces:

$$w_i = \max_{k,l} \tau_i^{k\,l} =: \tau_i^{\max}. \tag{13}$$

The computational domain is subsequently decomposed so that the amount information sent between the subdomains is minimised. We conjecture that as one minimises the activity between subdomains then one maximises the accuracy of the overall domain decomposition approach. We expect, further, that the domain decomposition will isolate subdomains in terms of the dynamics of the system. The second criterion (solutions in the subdomains have a similar accuracy), is also met by requiring that the sum of the weights be similar in each subdomain. Thus if the activity level within a subdomain is relatively low (as characterised by the Reynolds stresses) then this subdomain will contain more nodes than other subdomains with a greater activity level.

Let us now consider basing the weights on the maximum singular value of the discarded modes of the SVD. We recall that the aim is to find a scalar field composed of the weights at each node, which can be used as a surrogate to approximate the communication between subdomains and the load within each subdomain. An equal load, activity or accuracy within each subdomain would

be achieved if, for a given number of POD basis functions, the corresponding singular values that are neglected are approximately equal, i.e. the approximation to the 'seen' data within each subdomain is approximately the same. Using the original global NIROM model we can try to form a correlation between the maximum error in the Reynolds stresses τ^{err} over the entire domain and the largest singular value λ^{max} associated with the neglected POD modes. Looking at figure 2, we postulate that the relationship takes the following form:

$$\tau^{\text{err}} = \alpha \left(e^{(\beta \lambda^{\text{max}})} - 1 \right). \tag{14}$$

Further we assume that (1) the maximum truncated singular value associated with each node can be calculated from the maximum value of the Reynolds stress tensor at each node, that is $\tau^{\text{err}} = \tau^{\text{max}}$; and (2) that the sum of these singular values associated with each node should be balanced within each subdomain in order for there to be a balance of load or truncation of the singular values within each subdomain. Thus, setting the weights, at each node, equal to these singular values, at each node, is a suitable choice.

Rearranging equation (14), and assuming that it holds at each node i, and using $\tau^{err} = \tau^{max}$ gives

$$\lambda_i^{\text{max}} = \frac{1}{\beta} \ln \left(\frac{\tau_i^{\text{max}}}{\alpha} + 1 \right), \tag{15}$$

in which α and β are two curve fitting scalars. We have found that $\alpha = 0.05$ and $\beta = 0.0019183$ are suitable choices, see figure 2.

We thus have three possible weights at each finite element node w_i, that is:

$$w_i = \tau_i^{\max} \quad \text{from equation (13)}, \tag{16}$$

$$w_i = \lambda_i^{\max} \quad \text{from equation (15)}, \tag{17}$$

$$\text{or} \quad w_i = 1, \quad \forall i. \tag{18}$$

It is worth noting that equation (17) is closer than equation (16) to the uniform weighting, i.e. it has less variation for a given Reynolds stress field.

If we have other fields (other than just the velocity) then we might form similar expressions to the above based on the standard deviation of these fields rather than Reynolds stresses, say. We are then presented with choices on how we balance the subdomains based on two or more weights at each node. A maximum criteria would be one option, for example.

4.2. Domain Decomposition NIROM

This section describes how domain decomposition can be applied to NIROM. In the DDNIROM approach, the computational domain Ω is divided into S subdomains, $\Omega^d, d \in \{1, 2, \cdots, S\}$ and each subdomain has local unknowns $\Psi^d \in \mathbb{R}^{M^d}$, where M^d is the number of nodes in subdomain d and $M = M^1 + \cdots + M^d + \cdots + M^S$ is the total number of nodes.

In this method, the solutions at subdomain Ω^d are used to form a set of local POD basis functions $\mathbf{\Phi}^d$, the jth column of which is $\boldsymbol{\phi}_j^d$. The local POD basis functions are calculated by performing an SVD on the nodal values of the snapshots from only this local subdomain.

For each subdomain Ω^d, we construct a set of hypersurfaces $\{f_i^d\}_{i=1}^{m^d}$ to represent the underlying dynamical system associated with this subdomain and the surrounding subdomains over the re-

duced space (where m^d is the number of POD basis functions in subdomain d). Each hypersurface has the form of

$$\alpha_i^{d,n} = f_i^d(\boldsymbol{\alpha}^{d,n-1}, \boldsymbol{\alpha}^{sd,n}), \tag{19}$$

where the vector $\boldsymbol{\alpha}^{d,n-1}$ denotes the complete set of POD coefficients (i.e. the POD coefficients for all the solution variables required which is u, v and w in this case) at previous time level $n-1$ for the subdomain Ω^d, $\boldsymbol{\alpha}^{sd,n}$ denotes the complete set of POD coefficients at time level n for the surrounding subdomains.

The NIROM method is then used to construct a set of hypersurfaces for each POD coefficient of each subdomain. The procedure is described in algorithms 1 (offline) and 2 (online). Algorithm 1 presents the offline computational procedure of how to construct a set of local hypersurfaces for each subdomain while Algorithm 2 describes the online computation of DDNIROM where interactions between a subdomain and its surrounding subdomains are taken into account. At every time level, information about the incompressibility must be communicated to all the subdomains, therefore we chose an implicit coupling resulting in a recurrent neural network. In Algorithm 2, the inputs of the function f_i^d include POD coefficients from the previous time level $\boldsymbol{\alpha}^{d,n-1}$ (known) and the current time level from surrounding subdomains $\boldsymbol{\alpha}^{sd,n}$. The latter might be unknown, in which case, we take the value from the previous time level. In order to solve the implicit system we introduce an iteration method to obtain $\boldsymbol{\alpha}^{d,n}$ at current time level n. The iteration loop (**for** it = 1 to $N_{iteration}$) ensures that the hypersurface incorporates the flow dynamics over the subdomain d, but also, includes the flow interaction between the subdomain and its neighbouring subdomains.

Convergence could be checked by using the following criterion

$$ {}^{(it)}\text{error}^{d,n} = \frac{||{}^{(it)}\boldsymbol{\alpha}^{d,n} - {}^{(it-1)}\boldsymbol{\alpha}^{d,n}||_\infty}{||{}^{(it)}\boldsymbol{\alpha}^{d,n}||_\infty} < \text{tol} \tag{20} $$

where tol is a tolerance provided by the user and the infinity norm is defined as

$$ ||\boldsymbol{x}||_\infty = \max_j x_j \,. \tag{21} $$

In this work we use just one iteration and the results from the DDNIROM were found to be close to the full model (see figure 6 and table 1). The reason why we found a single iteration was adequate, for this example, was that these subdomains have been defined so that there is relatively little activity between subdomains. This enables the iterative domain decomposition method to converge quickly. In addition, even a single iteration has a level of associated implicitness, that is, the solution is updated by using information from neighbouring subdomains as one sweeps across the subdomains.

Algorithm 1: Offline: constructing a set of hypersurfaces for DDNIROM

(1) Generate $\mathcal{N}_s$ snapshots over the time period $[0, T]$ by solving the governing equations

(2) Divide the computational domain Ω into S subdomains

(3) Generate POD basis functions and coefficients

for $d = 1$ *to* S **do**

(a) Generate the POD basis functions $\mathbf{\Phi}^d$ by performing an SVD on the snapshots matrix of subdomain d;

(b) Calculate a set of POD coefficients $(\alpha_1^{d,n}, \alpha_2^{d,n}, \cdots, \alpha_{m^d}^{d,n})$ for each subdomain by projecting the snapshots into a reduced space (m^d is the number of POD coefficients in subdomain d);

endfor

(4) Obtain a set of hypersurfaces

for $d = 1$ *to* S **do**

for $i = 1$ *to* m^d **do**

(i) Set the values $\boldsymbol{y}_i^d = \left\{\alpha_i^{d,1}, \alpha_i^{d,2}, \cdots, \alpha_i^{d,N_t}\right\}$ for the ith POD coefficient

(ii) Obtain the hypersurface f_i^d, for subdomain d and POD coefficient i, by training the GPR network;

endfor

endfor

Algorithm 2: Online DDNIROM calculation for the fluid problem

for $n = 1$ *to* N_t **do**
!! *time loop*

!! *calculate the POD coefficients at the current time step:*
for $it = 1$ *to* $N_{iteration}$ **do**
!! *iteration loop*

for $d = 1$ *to* S **do**
!! *subdomain loop*

if $it == 1$ Initialise POD coefficients for subdomain d ($\boldsymbol{\alpha}^{d,0}$)

for $i = 1$ *to* m^d **do**
!! *POD coefficient loop*

(i) Evaluate the hypersurface f_i^d by using the complete set of POD coefficients $\boldsymbol{\alpha}^{d,n-1}$ and, if available, $\boldsymbol{\alpha}^{sd,n}$, which gives the POD coefficient $\alpha_i^{d,n}$ at the current time level n:

$$\alpha_i^{d,n} = f_i^d(\boldsymbol{\alpha}^{d,n-1}, \boldsymbol{\alpha}^{sd,n}) \tag{22}$$

(ii) The steps inside this loop should be carried out for the POD coefficients of every variable, i.e. all the velocity components

endfor
endfor
When $it > 1$ check convergence:

$$^{(it)}\text{error}^{d,n} = \frac{||^{(it)}\boldsymbol{\alpha}^{d,n} - {}^{(it-1)}\boldsymbol{\alpha}^{d,n}||_\infty}{||^{(it)}\boldsymbol{\alpha}^{d,n}||_\infty} < \text{tol} \tag{23}$$

!! *project the POD coefficients to the full space*
for $d = 1$ *to* S **do**
!! *Calculate the solution* $\boldsymbol{u}^{d,n}$ *for each subdomain d for each time step n by projecting* $\alpha_j^{d,n}$ *onto the full space.*

$$\boldsymbol{u}^{d,n} = \bar{\boldsymbol{u}}^d + \sum_{j=1}^{m^d} \alpha_j^{d,n} \boldsymbol{\phi}_j^d \tag{24}$$

endfor
endfor

5. Numerical Example: Modelling air flow around London South Bank University

This section demonstrates the ability of DDNIROM using a test case in the London South Bank University (LSBU) area, shown in Figure 3. The computational domain has a size of $[0, 2041] \times [0, 2288] \times [0, 250]$ (metres). A synthetic incoming-eddy method is used at the inlet [52]. The mean velocity profile is prescribed as in equation (25)

$$(u, v, w) = \left(0.97561 \ln\left(\frac{z}{0.01}\right), 0, 0\right) \tag{25}$$

where z denotes the height (in m). The Reynolds stresses $\boldsymbol{Re}$ (equation (26)) and the length-scale $\boldsymbol{L}$ (equation (27)) are prescribed constant and equal to 0.8 and 100 m respectively, for the diagonal components, and zero elsewhere.

$$\boldsymbol{Re} = \begin{pmatrix} 0.8 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 0.8 \end{pmatrix}, \tag{26}$$

$$\boldsymbol{L} = \begin{pmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{pmatrix}. \tag{27}$$

Zero velocity is prescribed on the wall boundaries and bottom. A perfect slip condition is specified on the vertical lateral boundaries and zero stress conditions is set to be $p = 0$ at the outlet boundary. In this case, the streamwise direction corresponds to a westerly wind direction. The high-fidelity solutions were obtained by running in parallel the open-source finite element fluid dynamics code

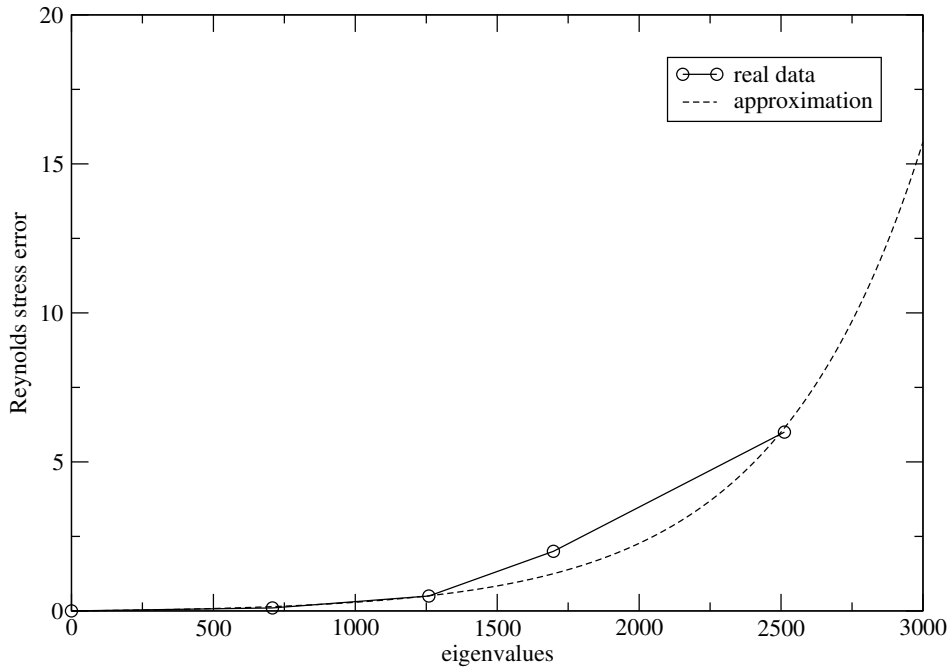
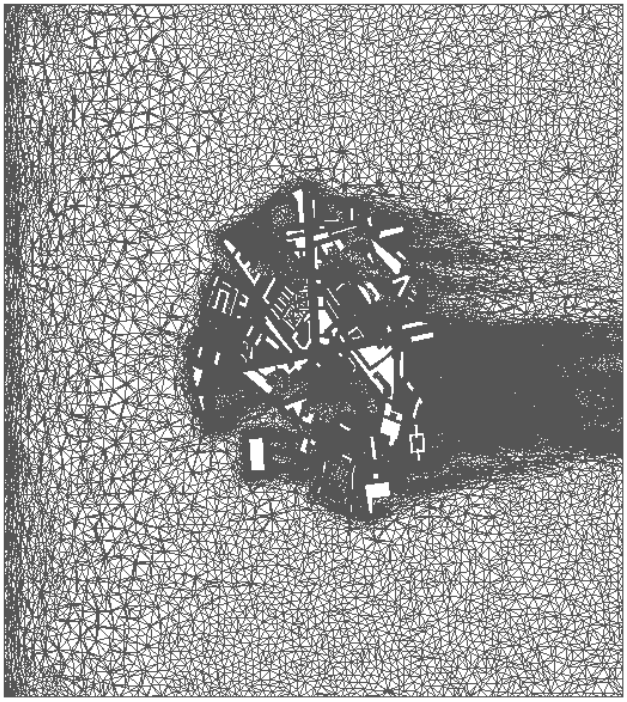


Figure 2: Graph of the relationship between the maximum Reynolds stress error, $\tau_i^{\max}$, for global NIROM and the maximum singular value $\lambda_i^{\max}$ that is neglected after truncating the SVD at a certain level. The graph also shows a fit to this data, given in Equation (15).

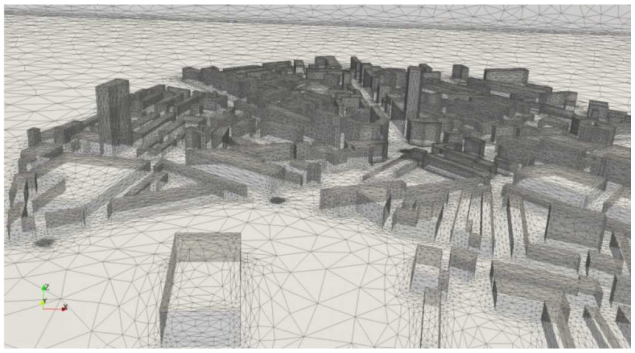
called Fluidity [53, 54]. Fluidity has been parallelised using standard domain decomposition techniques, see [55], where the computational domain is partitioned and each processor is assigned a partition. The Message Passing Interface library is used to communicate between processors. The velocity equations were solved with the Generalised Minimal Residual method and a symmetric successive over-relaxation preconditioner, and the pressure equations were solved with the conjugate gradient method and the BoomerAMG preconditioner. More information about these solvers and preconditioners can be found in the PETSc Users' Manual [55] and technical report [56].

To initialise the problem, the LES simulation was performed for one hour (in real time). The mesh was adapted every three time steps, specifying a maximum edge length of 50 m and a minimum edge length of 0.3 m. This time interval (one hour real time) is sufficient for the flow statistics to reach a quasi-steady state. From this point onwards, the mesh is fixed, as shown in Figure 3.

After the initialisation stage, the simulation is continued for 2000 seconds with a time-step size



(a) A horizontal slice at a height of 15 m



(b) Surface mesh of LSBU test site



(c) Vertical slice

Figure 3: The plots show (a) a horizontal slice at a height of 15 m above the ground; (b) the surface mesh of the test site; (c) the mesh on a vertical slice through the centre-line of the tallest building and parallel to the streamwise direction.

of $4/3$ s on the fixed unstructured mesh with $767,559$ nodes. During the 2000 seconds, 500 snapshots were taken every 4 seconds from the high-fidelity LES simulation results. The DDNIROM was trained with these 500 snapshots.

Figure 4 shows the x component of the Reynolds stress ($u'u'$) on a vertical plane through the largest building in the domain and this plane is aligned with the streamwise direction. In addition, in the same figure, we show the same Reynolds stresses at a height of 1m above the ground and viewed from underneath the domain. We also show the some of the subdomains viewed from underneath the domain at this height. What should be noticed is that the subdomain boundaries tend to avoid the areas with large Reynolds stresses. These results and, unless stated, all that follow, use the maximum Reynolds stresses to determine the nodal weights in the decomposition.

The singular values associated with the SVD of the global NIROM are shown in figure 5 and are slow to decrease after the 30th singular value. Also shown in this figure are the singular values associated with the subdomains following the DDNIROM approaches, with vertex weights determined by equations (18), (16) and (17). Notice that the singular values associated with the subdomains are always less than the global NIROM, but for the uniform weight case $w_i = 1$ some of the subdomains have singular values close to the global NIROM. This suggests that in parts of the domain the solution will not be accurate for this DDNIROM. However, if one uses the Reynolds stresses or approximate singular values as weighting parameters (Figure 5 (c) and (d)) then the singular values associated with all the subdomains are substantially lower than those associated with the global NIROM with the weights defined by the singular values, equation (17), being slightly better (that is, decaying more rapidly). This is the reason why the resulting DDNIROMs are more accurate, see figures 6, 7 obtained with only 24 basis functions in each subdomain. Notice in these images that the DDNIROM with 24 basis functions is considerably more accurate than the global

NIROM with 48 basis functions at the time levels shown and, in fact, at all times. The L_2 error norm of the streamwise velocity is presented in table 1 for both NIROM and DDNIROM, in which the high-fidelity model is taken as the reference solution. The norm is defined as follows:

$$e_{L_2}^{\mathrm{NIROM}}(t) = \frac{||\boldsymbol{u}^{\mathrm{HFM}}(t) - \boldsymbol{u}^{\mathrm{NIROM}}(t)||_2}{||\boldsymbol{u}^{\mathrm{HFM}}(t)||_2} \tag{28}$$

where $\boldsymbol{u}^{\mathrm{NIROM}}(t)$ represents the streamwise nodal velocities from NIROM at time t and $\boldsymbol{u}^{\mathrm{HFM}}(t)$ represents the streamwise nodal velocities from the high-fidelity model at time t. The error norm for DDNIROM is similarly defined.

	$t = 320\,\mathrm{s}$	$t = 1400\,\mathrm{s}$
$e_{L_2}^{\mathrm{NIROM}}$	24.78%	23.61%
$e_{L_2}^{\mathrm{DDNIROM}}$	10.98%	15.23%

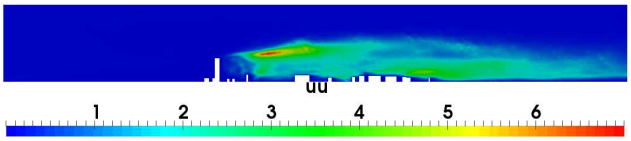
Table 1: L_2 Error norms for NIROM (48 basis functions) and DDNIROM (24 basis functions) calculated for the streamwise velocity at 320 s using the high-fidelity model as a reference solution. (See equation (28) for the definition of the error norms.)

In figure 7 we compare the flow speed of the high-fidelity model, the NIROM with 48 POD bases and the DDNIROM with 24 basis functions at time instances $t = 320\,\mathrm{s}$ (left panel) and $t = 1400\,\mathrm{s}$ (right panel). The results are shown on a plane through the centre-line of the tallest building and parallel to the streamwise direction. This figure shows the superior performance of DDNIROM over NIROM with 48 basis functions as well as the ability of DDNIROM to reproduce closely the high-fidelity model result. In figure 8 we see a similar plot but of DDNIROM at a number of consecutive time levels. In this plot can we can see that the DDNIROM is able to capture eddies moving downstream of the tall building.

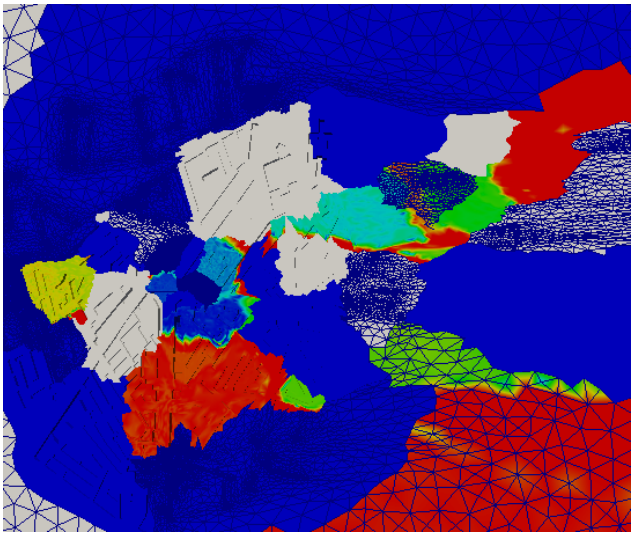
In figure 9 we show the number of nodes within each subdomain for the three methods of determining weights for the decomposition, given by equations (16), (17) and (18). In this figure

the subdomains are ordered in increasing number of nodes belonging to them. Notice that the method with equal weights (equation 18) has approximately equal number of nodes within each subdomain. This is approximate because the ParMETIS partitioning method also tries to minimise the communication between subdomains as defined by section 4 and thus balancing the weight sum in each subdomain is only approximate. Also notice that the weights based on the maximum stress, equation (16), result in the biggest variation of the the node numbers in each subdomain simply because it has the biggest variation between equations (16) and (17). This increased variation can be seen in figure 2.

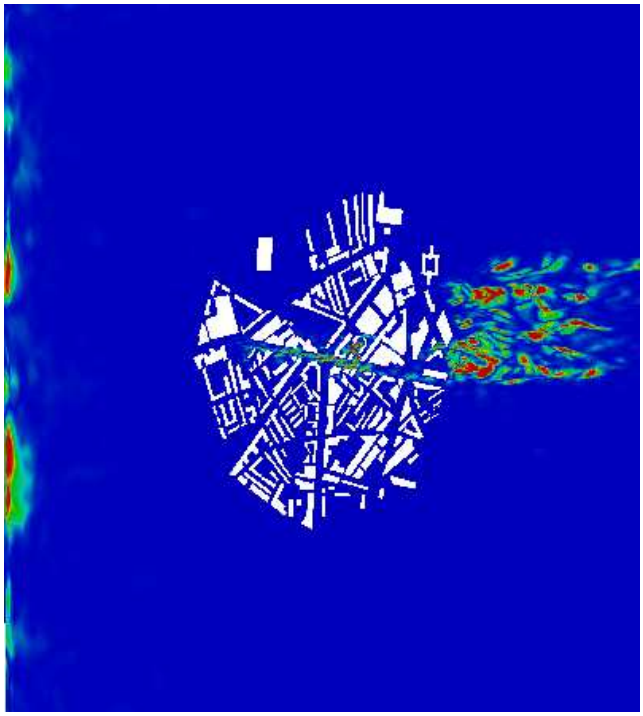
In figure 11 we show the time series of two points in the domain (near the busy intersection, see figure 10). Here we show on the same graph the first 2000 seconds obtained from the LES model and results from DDNIROM from the range $[2000, 4000]$. Visually the LES and DDNIROM models seem to reproduce similar frequencies and amplitudes of the x-component of velocity (streamwise component). Moreover, they also produce similar probability density functions, see figure 12. This helps to show that the DDNIROM model is an accurate model in itself with similar dynamics to the high fidelity model.



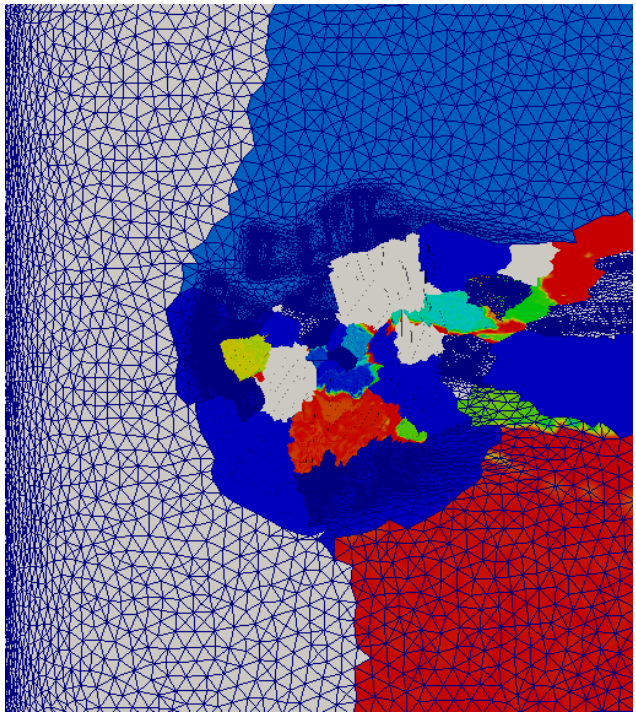
(a) Vertical slice of Reynolds stress.



(b) Central part of 32 subdomains.



(c) Horizontal slice of Reynolds stress.



(d) 32 subdomains in different colours.

Figure 4: Plots of Reynolds stresses and the resulting domain decomposition. (a) and (c) show the Reynolds stresses on vertical and horizontal slices through the domain. In (a) the plane passes through the tallest building and is aligned with the streamwise direction. In (c) the plane lies 1 m above ground level. Plots (b) and (d) give an indication of some of the 32 different subdomains on a horizontal slice through the domain. Plot (b) zooms in on the central part.

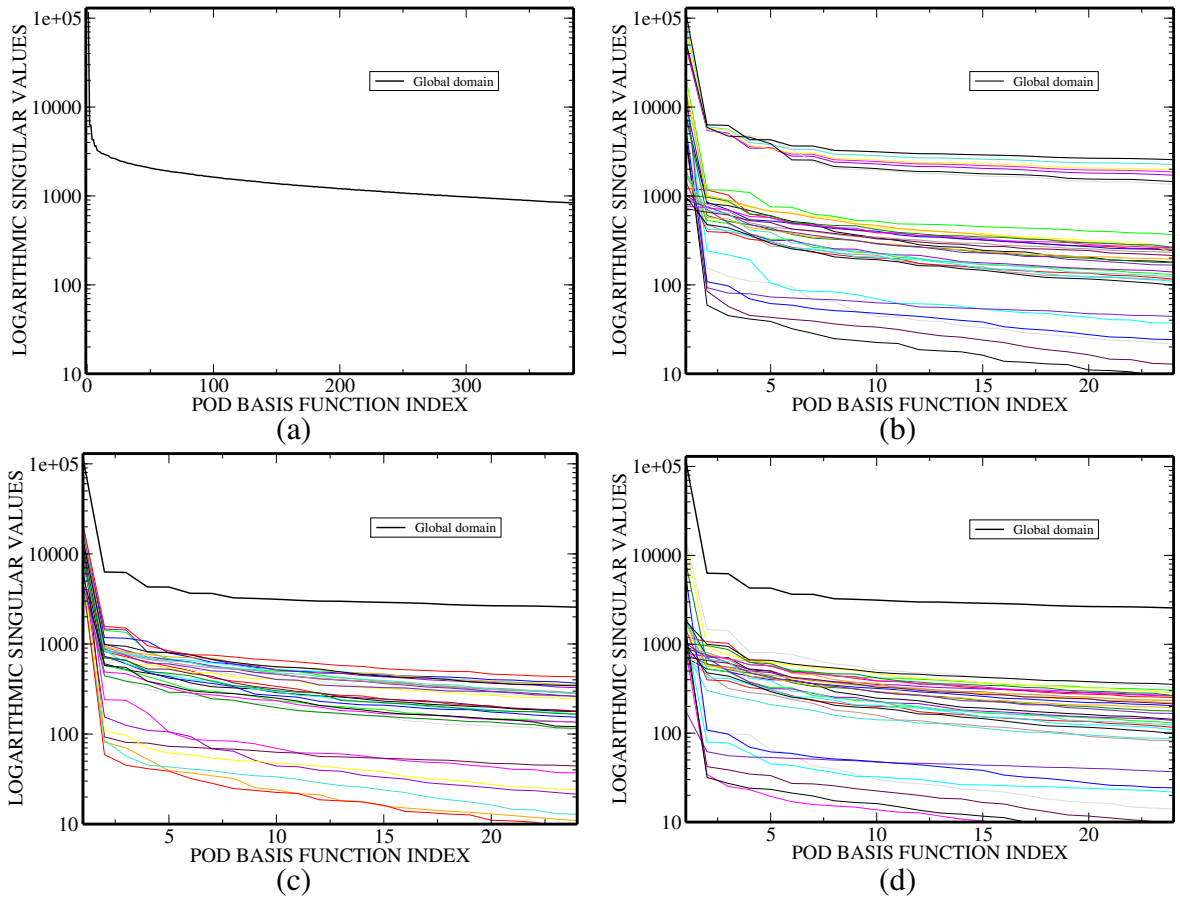
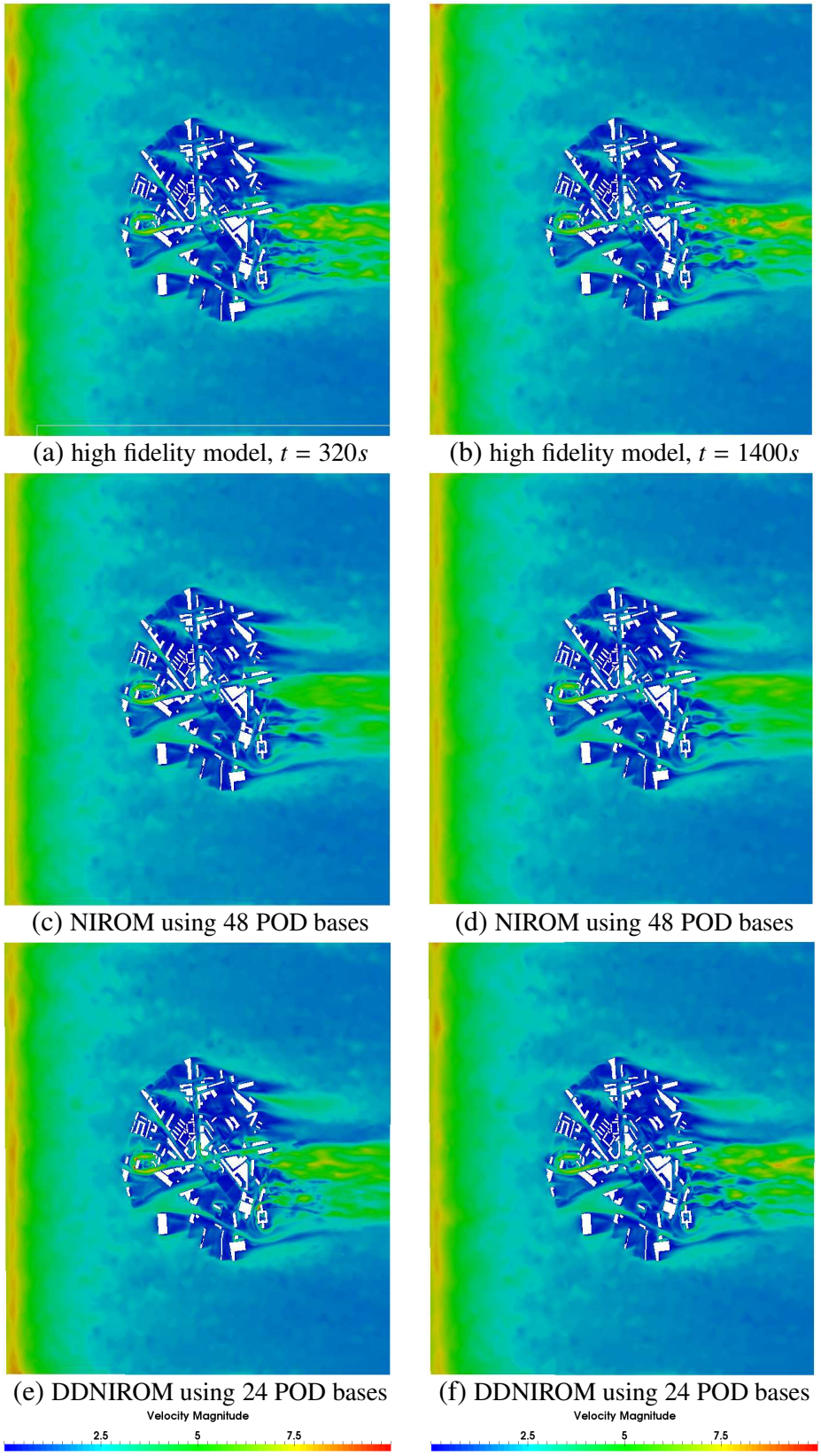


Figure 5: Plots showing the singular values versus number of POD basis functions with a logarithmic scale for: (a) global NIROM; (b) DDNIROM where the decomposition has been performed with a uniform nodal weighting, see equation (18); (c) DDNIROM a decomposition formed with weights based on the Reynolds stresses as defined in equation (16) (d) DDNIROM with the decomposition formed by weights based on an estimation of the largest singular value associated with the neglected modes in the SVD, see equation (15) . The original NIROM result is shown in black on every plot and the other 32 subdomain results are shown in different colours.



(a) high fidelity model, $t = 320s$

(b) high fidelity model, $t = 1400s$

(c) NIROM using 48 POD bases

(d) NIROM using 48 POD bases

(e) DDNIROM using 24 POD bases

(f) DDNIROM using 24 POD bases

Figure 6: A comparison of velocity solutions between the high-fidelity model, NIROM with 48 POD bases at time instances $t = 320$ s (left panel) and $t = 1400$ s (right panel) on a horizontal plane at a height of 15 m above the ground.

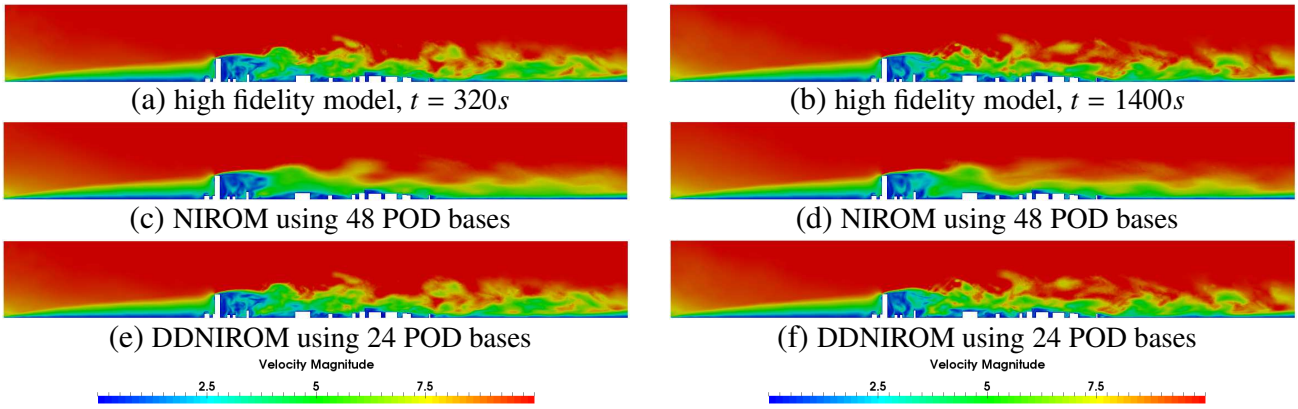


Figure 7: A comparison of velocity solutions between the high-fidelity model, NIROM with 48 POD bases and DDNIROM with 24 basis functions at time instances $t = 320\,\mathrm{s}$ (left panel) and $t = 1400\,\mathrm{s}$ (right panel) on a plane through the centre-line of the tallest building and parallel to the streamwise direction.

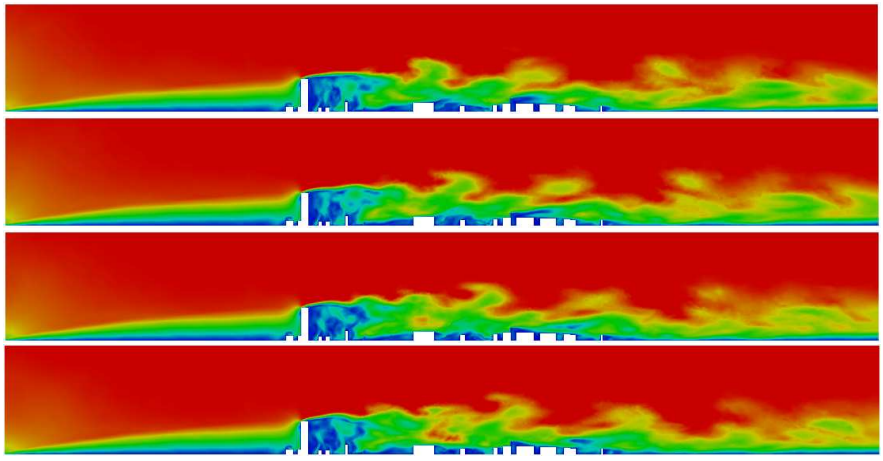


Figure 8: DDNIROM at time levels 2328 s, 2336 s, 2344 s, 2352 s on a plane through the centre-line of the tallest building and parallel to the streamwise direction.

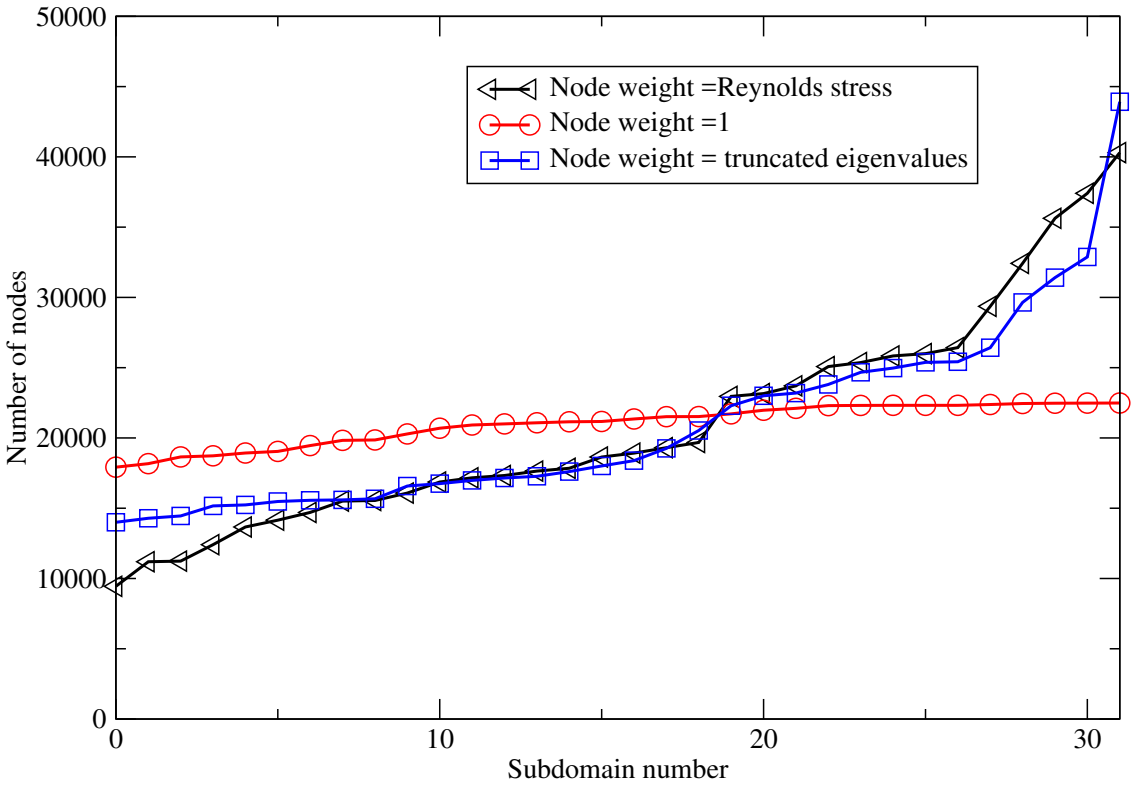


Figure 9: Subdomain number against number of nodes in each subdomain.

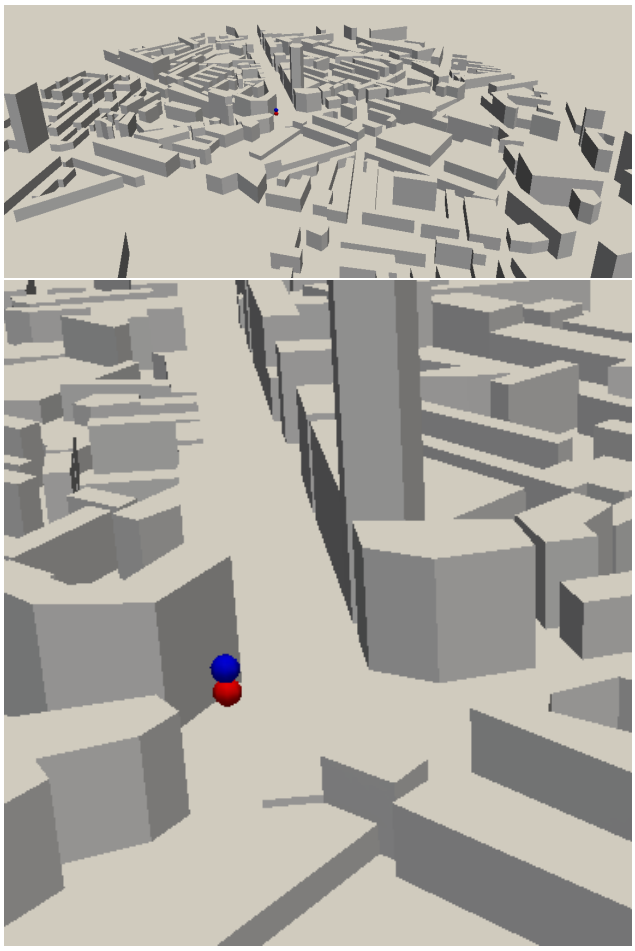


Figure 10: Location of two points: red point 5 m above the ground level and the blue point is 10 m above the ground level.

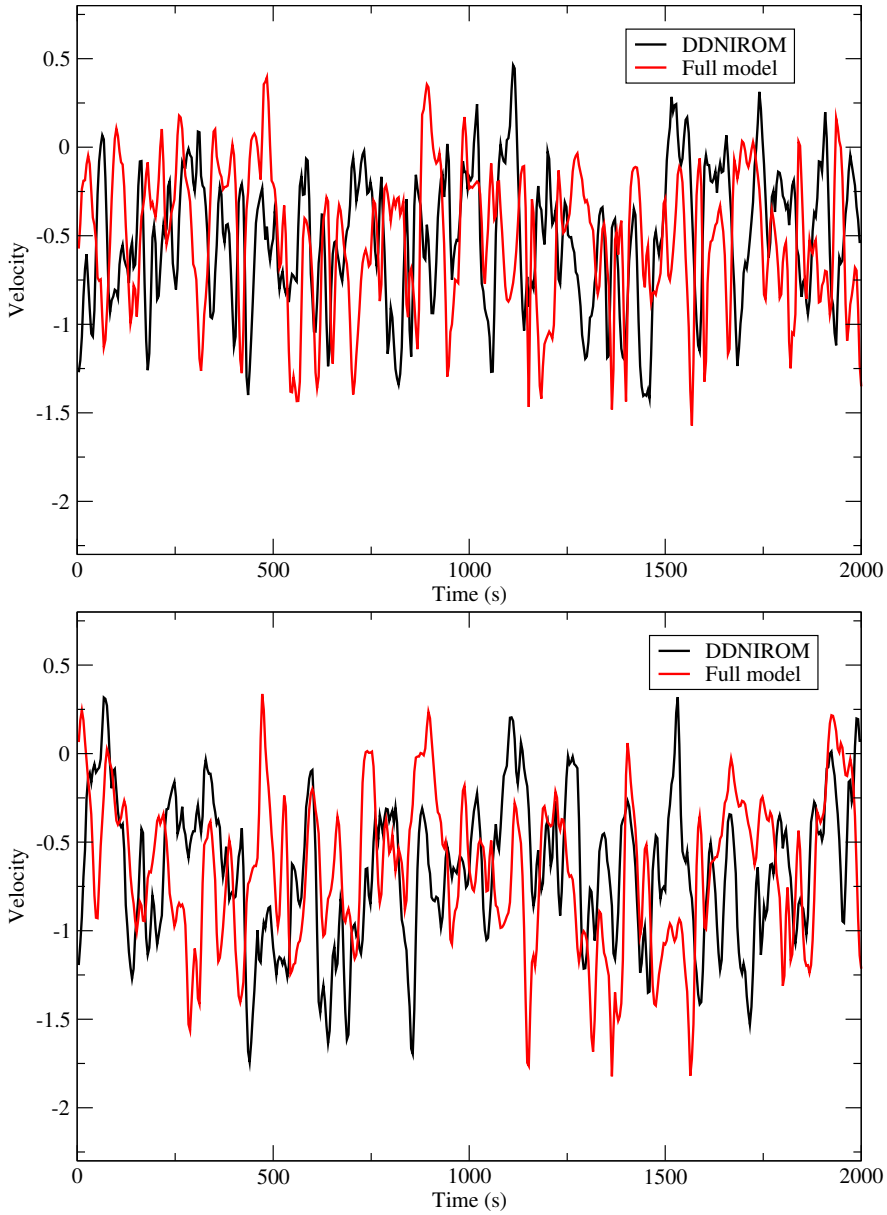
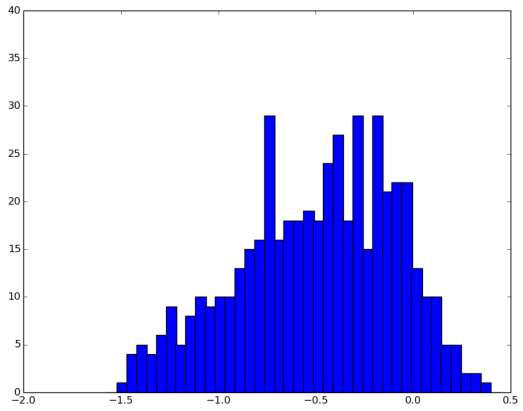
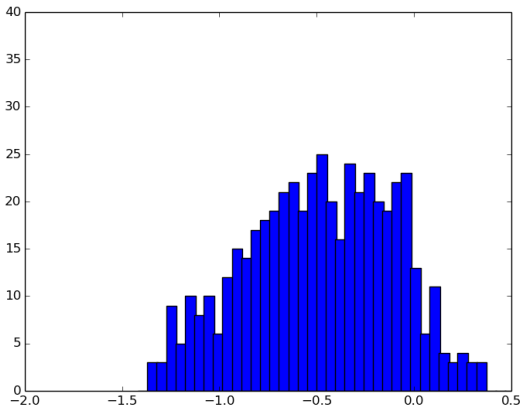


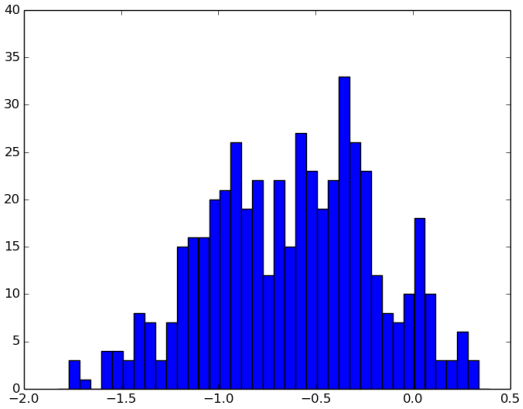
Figure 11: The plots displayed above show the time series of the x-component of the velocity (m/s) at the two locations shown in Figure 10 (the top plot corresponds to the higher (blue) point and the bottom plot to the lower (red) point) from the high-fidelity model and from DDNIROM (in the predicting time interval) with 24 basis functions. The high fidelity model result is shown between 0 and 2000 seconds and the DDNIROM is shown between 2000 seconds and 4000 seconds but on the same axis.



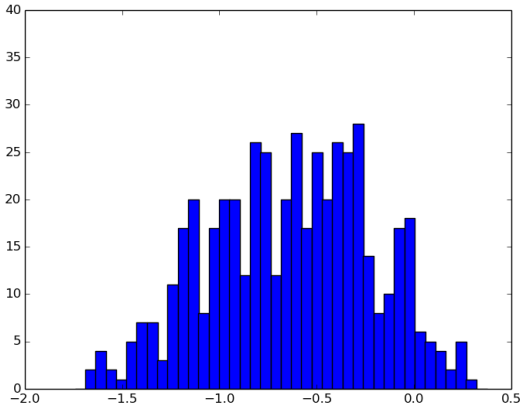
(a) high fidelity model at 10 m above the ground



(b) DDNIROM 10 m above the ground



(c) high fidelity model at 5 m above the ground



(d) DDNIROM 5 m above the ground

Figure 12: Probability density functions of the x-component of velocity of the air at 5 m and 10 m above the ground level and at the two points shown in figure 10. Left are the high fidelity model results and right the DDNIROM results with 24 POD basis functions in each subdomain. The horizontal axis is the interval of velocity (m/s) in the x-direction or streamwise direction.

Table 2 shows the CPU cost required by both the high-fidelity model and DDNIROM to solve for 4 seconds in real time (which corresponds to three time steps for the high-fidelity model and one time step for the DDNIROM). The high-fidelity model solves for just over 3 million degrees of freedom, whereas the DDNIROM has just 72 degrees of freedom in each subdomain. It is worth noting that the CPU time required to solve the DDNIROM over this time interval is only 0.128 s, whereas the high-fidelity model requires 1555 s running in parallel on a workstation with 10 Intel® Xeon® X5680 CPU processors (each core has a frequency of 3.3 GHz) and 512 GB RAM. As the DDNIROM has so few degrees of freedom there would be no gain from running this model in parallel. The total amount of time required to generate the 500 snapshots for this example was 9 days approximately.

Table 2: Comparison of the CPU cost (in seconds) required to solve the high fidelity model and DDNIROM for 4 seconds in real time. For the DDNIROM, 'solution' corresponds to step (a) in the flow chart, figure 1, and 'projection' corresponds to step (b).

	high-fidelity model	DDNIROM
assembly	1555	n/a
solution		0.128
projection	n/a	0.003
total	1555	0.131

6. Conclusions

In this paper we develop a method of decomposing the computational domain into subdomains in order to enhance the performance of the Domain Decomposition Non-Intrusive Reduced Order Model (DDNIROM). Within each subdomain we use the Gaussian Process Region (GPR) (a machine learning method) to obtain a hypersurface that approximates the discretised governing equations. Each local hypersurface represents, not only the fluid dynamics over the subdomain it

belongs to, but also the interactions with the surrounding subdomains. This implicit coupling between the subdomains provides the global coupling necessary to enforce incompressibility and is a means of providing boundary conditions for each subdomain.

As suggested, choosing the subdomains is the key to the good performance of the method and we have investigated three different choices of weighting (of the finite element nodes) for the domain decomposition algorithm. If we use a node weight that is based on the maximum Reynolds stresses or the largest singular value of the discarded POD modes, then the subdomain approach worked well, because, by design, there is relatively little activity between subdomains. However, if one uses a uniform weight then the accuracy of the approach was not as good, and only marginally better than global NIROM (DDNIROM with one subdomain). In addition, it may not be so important to balance the accuracy associated with a fixed number of POD basis functions in each subdomain. The reason for this is because we can always choose a different number of basis functions associated with each subdomain as demonstrated previously in [41] in order to balance the accuracy within each subdomain.

The major advantage of using domain decomposition methods is that they: (a) may be used for large problems with lots of degrees of freedom; (b) are more accurate for the local hypersurface formation; (c) allow the use of smaller Singular Valued Decompositions within each subdomain when compared to SVDs across the entire domain; (d) lead to lower dimensional surface fits in the GPR than used in NIROM, which may result in a higher accuracy because they surface fit in lower dimensional spaces; (e) result in an ability to form the NIROMs independently by running models across each region independently.

This may make possible large-scale modelling, of, say an entire city, even based on detailed LES turbulent flow models. These advantages have been demonstrated within this paper. In ad-

dition, it is shown that even using relatively small number of POD basis functions within each subdomain then it is possible to form a high fidelity DDNIROM: only 24 basis functions were used in each subdomain in this work. The reason for this is the choice of the subdomains, in particular, minimising the activity between subdomains. This was demonstrated in the complex problem of turbulent flows around buildings. This is a fairly typical turbulent flow problem and thus we expect similar performance in other turbulent flow problems which will be the subject of future investigations.

Acknowledgments

The authors would like to thank the reviewers for their helpful comments which have improved the manuscript. The authors are grateful for the support of the following: Managing Air for Green Inner Cities (MAGIC) (EP/N010221/1); the EPSRC multi-phase flow programme grant MEMPHIS (EP/K003976/1); The IMPACT (Innovative Materials, Processing and Numerical Technologies) at Swansea University; the Innovate UK Smart-GeoWells consortium; Imperial College High Performance Computing Service; funding from the European Union Seventh Frame work Programme (FP7/20072013) under grant agreement No.603663 for the research project PEARL (Preparing for Extreme And Rare events in coastal regions) and the EPSRC MUFFINS project (MUltiphase Flow-induced Fluid-flexible structure InteractioN in Subsea applications EP/P033148/1).

References

[1] B.R. Noack, W. Stankiewicz, M. Morzyński, and P.J. Schmid. Recursive dynamic mode decomposition of transient and post-transient wake flows. *Journal of Fluid Mechanics*, 809:843–872, 2016.

[2] X. Xie, D. Wells, Z. Wang, and T. Iliescu. Approximate deconvolution reduced order modeling. *Computer Methods in Applied Mechanics and Engineering*, 313:512–534, 2017.

[3] S. Lorenzi, A. Cammi, L. Luzzi, and G. Rozza. POD-Galerkin method for finite volume approximation of Navier–Stokes and RANS equations. *Computer Methods in Applied Mechanics and Engineering*, 311:151–179, 2016.

[4] Y. Wang, I.M. Navon, X. Wang, and Y. Cheng. 2D Burgers equation with large Reynolds number using POD/DEIM and calibration. *International Journal for Numerical Methods in Fluids*, 82(12):909–931, 2016.

[5] B.R. Noack, M. Morzynski, and G. Tadmor. *Reduced-Order Modelling for Flow Control*, volume 528. Springer, 2011.

[6] M. Bergmann and L. Cordier. Optimal control of the cylinder wake in the laminar regime by trust-region methods and POD reduced-order models. *Journal of Computational Physics*, 227(16):7813–7840, 2008.

[7] G. Dimitriu, N. Apreutesei, and R. Ştefănescu. Numerical simulations with data assimilation using an adaptive POD procedure. In *Large-Scale Scientific Computing*, pages 165–172. Springer, 2010.

[8] J. Oliver, M. Caicedo, A.E. Huespe, J.A. Hernández, and E. Roubin. Reduced order modeling strategies for computational multiscale fracture. *Computer Methods in Applied Mechanics and Engineering*, 313:560–595, 2017.

[9] G. Dimitriu, R. Stefanescu, and I.M. Navon. POD-DEIM approach on dimension reduction of

a multi-species host-parasitoid system. *Ann. Acad. Rom. Sci. Ser. Math. Appl*, 7(1):173–188, 2015.

[10] F. Fang, T. Zhang, D. Pavlidis, C.C. Pain, A.G. Buchan, and I.M. Navon. Reduced order modelling of an unstructured mesh air pollution model and application in 2D/3D urban street canyons. *Atmospheric Environment*, 96:96–106, 2014.

[11] A. Manzoni, F. Salmoiraghi, and L. Heltai. Reduced Basis Isogeometric Methods (RB-IGA) for the real-time simulation of potential flows about parametrized NACA airfoils. *Computer Methods in Applied Mechanics and Engineering*, 284:1147–1180, 2015.

[12] F. Ballarin, E. Faggiano, S. Ippolito, A. Manzoni, A. Quarteroni, G. Rozza, and R. Scrofani. Fast simulations of patient-specific haemodynamics of coronary artery bypass grafts based on a POD–Galerkin method and a vascular shape parametrization. *Journal of Computational Physics*, 315:609–628, 2016.

[13] M. Schlegel and B.R. Noack. On long-term boundedness of Galerkin models. *Journal of Fluid Mechanics*, 765:325–352, 2015.

[14] J. Osth, B.R. Noack, S. Krajnovi, D. Barros, and J. Bore. On the need for a nonlinear subscale turbulence term in POD models as exemplified for a high-Reynolds-number flow over an Ahmed body. *Journal of Fluid Mechanics*, 747:518–544, 2014.

[15] L.P. Franca and S.L. Frey. Stabilized finite element methods: II. The incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 99(2):209–233, 1992.

[16] S. Chaturantabut and D.C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM J. Sci. Comput*, 32:2737–2764, 2010.

[17] D. Xiao, F. Fang, J. Du, C.C. Pain, I.M. Navon, A.G. Buchan, A.H. ElSheikh, and G. Hu. Non-linear Petrov–Galerkin methods for reduced order modelling of the Navier–Stokes equations using a mixed finite element pair. *Computer Methods In Applied Mechanics and Engineering*, 255:147–157, 2013.

[18] D. Xiao, F. Fang, A.G. Buchan, C.C. Pain, I.M. Navon, J. Du, and G. Hu. Non-linear model reduction for the Navier–Stokes equations using residual DEIM method. *Journal of Computational Physics*, 263:1–18, 2014.

[19] D. Xiao, F. Fang, A.G. Buchan, C.C. Pain, I.M. Navon, and A. Muggeridge. Non-intrusive reduced order modelling of the Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 293:522–541, 2015.

[20] D. Xiao, F. Fang, C.C. Pain, and I.M. Navon. A parameterized non-intrusive reduced order model and error analysis for general time-dependent nonlinear partial differential equations and its applications. *Computer Methods in Applied Mechanics and Engineering*, 317:868–889, 2017.

[21] Z. Wang, D. Xiao, F. Fang, R. Govindan, C.C. Pain, and Y. Guo. Model identification of reduced order fluid dynamics systems using deep learning. *International Journal for Numerical Methods in Fluids*, 86(4):255–268, 2018.

[22] D. Xiao, F. Fang, C.C. Pain, and G. Hu. Non-intrusive reduced order modelling of the Navier-

Stokes equations based on RBF interpolation. *International Journal for Numerical Methods in Fluids*, 79(11):580–595, 2015.

[23] D. Xiao, P. Yang, F. Fang, J. Xiang, C.C. Pain, and I.M. Navon. Non-intrusive reduced order modeling of fluid-structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 303:35–54, 2016.

[24] D. Xiao, Z. Lin, F. Fang, C.C. Pain, I.M. Navon, P. Salinas, and A. Muggeridge. Non-intrusive reduced-order modeling for multiphase porous media flows using Smolyak sparse grids. *International Journal for Numerical Methods in Fluids*, 83(2):205–219, 2017.

[25] H.A. Schwarz. Ueber einige abbildungsaufgaben. *Journal für die reine und angewandte Mathematik*, 70:105–120, 1869.

[26] J.S. Przemieniecki. Matrix structural analysis of substructures. *AIAA Journal*, 1(1):138–147, 1963.

[27] I. Schur. Gesammelte Abhandlungen. Band II., 1973.

[28] I. Schur. Vorlesungen über invariantentheorie. 1968.

[29] G. Meurant. Domain decomposition methods for partial differential equations on parallel computers. *The International Journal of Supercomputing Applications*, 2(4):5–12, 1988.

[30] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford Science Publications, 1999.

[31] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[32] C.C. Pain and A.J.H. Goddard. A neural network graph partitioning procedure for grid based domain decomposition. *International Journal of Numerical Methods in Engineering*, 44:593–613, 1999.

[33] D.J. Lucia, P.I. King, and P.S. Beran. Reduced order modeling of a two-dimensional flow with moving shocks. *Computers & Fluids*, 32(7):917–938, 2003.

[34] D.J. Lucia, P.I. King, and P.S. Beran. Domain decomposition for reduced-order modeling of a flow with moving shocks. *AIAA journal*, 40(11):2360–2362, 2002.

[35] J. Baiges, R. Codina, and S. Idelsohn. A domain decomposition strategy for reduced order models. Application to the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 267:23–42, 2013.

[36] P. Kerfriden, O. Goury, T. Rabczuk, and S.P.A. Bordas. A partitioned model order reduction approach to rationalise computational expenses in nonlinear fracture mechanics. *Computer Methods in Applied Mechanics and Engineering*, 256:169–188, 2013.

[37] D. Amsallem, M.J. Zahr, and C. Farhat. Nonlinear model order reduction based on local reduced-order bases. *International Journal for Numerical Methods in Engineering*, 92(10):891–916, 2012.

[38] S. Chaturantabut. Temporal localized nonlinear model reduction with a priori error estimate. *Applied Numerical Mathematics*, 119:225–238, 2017.

[39] H. Antil, M. Heinkenschloss, R.H.W. Hoppe, and D.C. Sorensen. Domain decomposition and model reduction for the numerical solution of PDE constrained optimization problems with

localized optimization variables. *Computing and Visualization in Science*, 13(6):249–264, 2010.

[40] H. Antil, M. Heinkenschloss, and R.H.W. Hoppe. Domain decomposition and balanced truncation model reduction for shape optimization of the Stokes system. *Optimization Methods and Software*, 26(4-5):643–669, 2011.

[41] D. Xiao, F. Fang, C.E. Heaney, I.M. Navon, and C.C. Pain. Domain decomposition for the non-intrusive reduced order modelling of fluid flow. *Computer Methods in Applied Mechanics and Engineering*, submitted.

[42] E. Aristodemou, T. Bentham, C.C. Pain, R. Colvile, A. Robins, and H. ApSimon. A comparison of mesh-adaptive LES with wind tunnel data for flow past buildings: Mean flows and velocity fluctuations. *Atmospheric Environment*, 43(39):6238–6253, 2009.

[43] J. Song, S. Fan, W. Lin, L. Mottet, H. Woodward, M. Davies Wykes, R. Arcucci, D. Xiao, J.-E. Debay, H. ApSimon, E. Aristodemou, D. Birch, M. Carpentieri, F. Fang, M. Herzog, G.R. Hunt, R.L. Jones, C.C. Pain, D. Pavlidis, A.G. Robins, C.A. Short, and P. F. Linden. Natural ventilation in cities: the implications of fluid mechanics. *Building Research & Information*, 46(8):809–828, 2018.

[44] D. Xiao, C.E. Heaney, L. Mottet, F. Fang, W. Lin, I.M. Navon, Y. Guo, O.K. Matar, A.G. Robins, and C.C. Pain. A reduced order model for turbulent urban flows using machine learning. *Building and Environment*, 148:323–337, 2019.

[45] R. Pinnau. *Model Reduction via Proper Orthogonal Decomposition*, pages 95–109. Springer Berlin Heidelberg, 2008.

[46] L. Sirovich. Turbulence and the dynamics of coherent structures, part III: Dynamics and scaling. *Quarterly of Appl.Math.*, XLV:583–590, 1987.

[47] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[48] C.E. Rasmussen. Gaussian Processes in Machine Learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[49] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[50] George Karypis and Vipin Kumar. Multilevel *k*-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing*, 48(1):96–129, 1998.

[51] George Karypis, Kirk Schloegel, and Vipin Kumar. Parmetis: Parallel graph partitioning and sparse matrix ordering library. 1997.

[52] D. Pavlidis, G. J. Gorman, J. L. M. A. Gomes, C. C. Pain, and H. ApSimon. Synthetic-eddy method for urban atmospheric flow modelling. *Boundary-Layer Meteorology*, 136:285–299, 2010.

[53] C.C. Pain, M.D. Piggott, A.J.H. Goddard, F. Fang, G.J. Gorman, D.P. Marshall, M.D. Eaton, P.W. Power, and C.R.E. De Oliveira. Three-dimensional unstructured mesh ocean modelling. *Ocean Modelling*, 10(1-2):5–33, 2005.

[54] *Fluidity version 4.1.12*, 2015.

[55] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.10, Argonne National Laboratory, 2018.

[56] Satish Balay, William D. Gropp, Lois C. McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhauser Press, 1997.