Adaptive Domain Decomposition for Effective Data Assimilation

Rossella Arcucci[1], Laetitia Mottet[2], César A. Quilodrán Casas[1], Florian Guitton[1], Christopher Pain[2], and Yi-Ke Guo[1]

[1] Data Science Institute, Department of Computing, Imperial College London, UK
[2] Department of Earth Science & Engineering, Imperial College London, UK

Abstract. We present a parallel Data Assimilation model based on an Adaptive Domain Decomposition (ADD-DA) coupled with the open-source, finite-element, fluid dynamics model Fluidity. The model we present is defined on a partition of the domain in sub-domains without overlapping regions. This choice allows to avoid communications among the processes during the Data Assimilation phase. However, during the balance phase, the model exploits the domain decomposition implemented in Fluidity which balances the results among the processes exploiting overlapping regions. Also, the model exploits the technology provided by the mesh adaptivity to generate an optimal mesh we name *supermesh*. The *supermesh* is the one used in ADD-DA process. We prove that the ADD-DA model provides the same numerical solution of the corresponding sequential DA model. We also show that the ADD approach reduces the execution time even when the implementation is not on a parallel computing environment. Experimental results are provided for pollutant dispersion within an urban environment.

Keywords: Data Assimilation · Fluidity · Domain Decomposition · Adaptive mesh · Big Data.

1 Introduction and Motivation

Numerical simulations are widely used as a predictive tool to better understand complex air flows and pollution transport at the scale of individual buildings, city blocks and entire cities. The strongly nonlinear character of many physical processes results in the dramatic amplification of even small input uncertainties producing large uncertainties in the system behavior [7]. To reduce these uncertainties and increase the accuracy of predictions, Data Assimilation (DA) techniques are used [15]. Data Assimilation (DA) is the approximation of the true state of some physical system at a given time by combining time-distributed observations with a dynamic model in an optimal way. DA can be classically approached in two ways: as variational DA [6] and as filtering, such that Kalman Filter (KF) [14]. In both cases, the methods are computed as an optimal solution: statistically, KF methods try to find a solution with minimum variance,

while variational methods compute a solution that minimises a suitable cost function. In certain cases, the two approaches are identical and provide exactly the same solution [15]. While the statistical approach it is often complex and time-consuming, it can provide a richer information structure. Variational approaches are relatively rapid and robust instead [6]. In DA, one makes repeated corrections to data during a single run, to bring the code output into agreement with the latest observed data. In operational forecasting there is insufficient time to restart a run from the beginning with new data then, DA should enable real-time utilisation of data to improve predictions. This mandates the choice of efficient methods to opportunely develop and implement DA models.

Due to the necessity to have DA in real time, we introduce in this paper an efficient Adaptive Domain Decomposition approach for variational Data Assimilation (ADD-DA). The ADD-DA model is presented to assimilate data from sensors into the open-source, parallelised fluid dynamics model Fluidity (`http://fluidityproject.github.io/`). It is estimated that by 2050, around four-million deaths per year will be attributable to outdoor air pollution (twice the current mortality rate) [16]. This mandates the development of techniques that can be used for emergency response, real-time operational prediction and management. A variational DA model (VarDA) to assimilate air pollution data has been introduced in [4] in which it has been shown that the use of an optimal space for solving DA can reduce the execution-time. However, the interface between this VarDA and Fluidity present a big bottleneck due to the gathering of the data after Fluidity to run DA (as shown in Fig. 1-left).

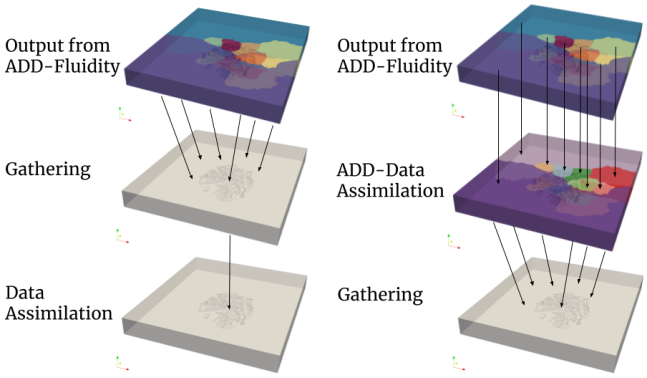


Fig. 1: Comparison of the sequential VarDA coupled with Fluidity (left) and ADD-DA coupled with Fluidity (right)

Parallelisation of data assimilation has been previously used in oceanography, however its application for urban air pollution is novel. Previous works on parallelised data assimilation include domain decomposition (DD) in a regional ocean model of Australia [20] and a parallel implementation of data assimilation for operational oceanography by [22,9]. A different approach which introduce a reduc-

tion on the execution time is presented in [10] where the authors use a recursive filter (RF) with the third order of accuracy (3rd-RF) to approximate horizontal Gaussian covariances. A domain decomposition approach for data assimilation has been presented in [2] where the authors implemented a geographical decomposition made of sub-domains (of fixed same sizes) with overlapping.

In this paper, we developed a DA model based on the same domain decomposition (DD) implemented in Fluidity such that the DA process can be coupled with Fluidity in a straight forward way (Fig. 1-right). We propose a DD approach which does not include any interaction/communication among the sub-domains, thus removing the interaction/communication overhead. We prove that the solution of the ADD-DA model is the same of the VarDA model without any decomposition introduced in [4]. We also show that the approach reduces the execution time even when the implementation is not on a parallel computing environment.

This paper is structured as follows. The ADD-DA model is presented in Section 2. The set-up of the test case is detailed in Section 3 where results using ADD-DA to improve the results of the pollutant concentration are presented. The scalability of the ADD-DA approach is also discussed in Section 3. Finally, conclusions and future work are provided in Section 4.

2 The ADD-DA model

In this section we present a Data Assimilation model based on an Adaptive Domain Decomposition to be coupled with Fluidity.

One of the key and innovative aspects of Fluidity is its mesh-adaptivity capability on unstructured meshes. The use of the mesh-adaptivity allows to have fine mesh in regions where small-scale and important physical processes occur, while keeping a coarser mesh elsewhere, and then allowing to considerably reduce the total computation time [21]. Fluidity was running with mesh adaptivity for an enough long time for the flow statistics to reach a quasi-steady state. From this point onwards, the mesh is fixed and considered as the optimal mesh. This mesh will be referred as the *supermesh* in the following.

Let $\Omega = \{x_j\}_{j=1,\dots,n}$ be the discrete spatial domain representing the *supermesh* and let $\mathcal{P}(\Omega) = \{\Omega_i\}_{i=1,\dots,s}$ be a partition of Ω in subdomains as implemented in Fludity [1]. In Fluidity, the decomposition of the domain into sub-domains is based on the number of nodes x_j. The number of nodes assigned to a partition/processor is assumed to be a good proxy for the expected computational load on that processor and, hence, one aim of the algorithm is to equidistribute the nodes. Other requirements include a minimization of edge cut and data migration. The number of nodes is balanced to be more or less the same on each processor, even if it is not a strict constraint.

For a fixed time t, according to this decomposition, let

$$u_i^{\mathcal{M}} \equiv u_i(t) \tag{1}$$

be the vector denoting the state of the dynamical system. At time t, we get $u(t) = \mathcal{M}\left(u(t-1)\right)$ where $\mathcal{M}$ is the forecasting model , in our case represented by Fluidity. Let be

$$v_i = H_i(u_i) \tag{2}$$

the vector of observations where H_i is an interpolation operator collecting the observations at time t. The aim of DA problem is to find an optimal trade-off between the current estimate of the system state (the background) in (eq.1) and the available observations v_k in (eq.2).

ADD-DA computational model is a system of s non-linear least square problems:

$$u_i^{ADD-DA} = argmin_{u_i} \|u_i - u_i^{\mathcal{M}}\|^2_{\mathbf{B}_i} + \|H_i(u_i) - v_i\|^2_{\mathbf{R}_i} \tag{3}$$

where $\mathbf{R_i}$ and $\mathbf{B_i}$ are the covariance matrices providing the estimate of the errors on v_i and on $u_i^{\mathcal{M}}$, respectively.

As the background error covariance matrix $\mathbf{B_i}$ is ill-conditioned [19], in order to improve the conditioning, only Empirical Orthogonal Functions (EOFs) of the first largest eigenvalues of the error covariance matrix are considered. Since its introduction to meteorology [18], EOFs analysis has become a fundamental tool in atmosphere, ocean, and climate science for data diagnostics and dynamical mode reduction. Each of these applications basically exploits the fact that EOFs allow a decomposition of a data function into a set of orthogonal functions, which are designed in such a way that only a few of these functions are needed in lower-dimensional approximations [13]. Furthermore, since EOFs are the eigenvectors of the error covariance matrix [12], its condition number is reduced as well. Nevertheless, the accuracy of the solution obtained by truncating EOFs exhibits a severe sensibility to the variation of the value of the truncation parameter, so that a suitably choice of the number of EOFs is strongly recommended. This issue introduces a severe drawback to the reliability of EOFs truncation, hence to the usability of the operative software in different scenarios [12,3]. In this paper, we set the optimal choice of the truncation parameter as a trade-off between efficiency and accuracy of the DA algorithm as introduced in [4]. The Optimal ADD-DA model as implemented in this paper is summarised in Algorithm 1. Eq. (3) is linearised around the background state [17]:

$$u_i = u_i^{\mathcal{M}} + \delta u_i \tag{4}$$

where $\delta u_i = u_i - u_i^{\mathcal{M}}$ denotes the increments. The ADD-DA problem can then be re-formulated by the following form:

$$\delta u_i^{ADD-DA} = argmin_{\delta u_i} \left\{ \frac{1}{2} \delta u_i^T \mathbf{B}_i^{-1} \delta u_i + \frac{1}{2} (\mathbf{H}\delta u_i - d_i)^T \mathbf{R}_i^{-1} (\mathbf{H}\delta u_i - d_i) \right\} \tag{5}$$

where

$$d_i = [v_i - H\left(u_i^{\mathcal{M}}\right)] \tag{6}$$

is the misfit between the observation and the solution computed by Fluidity and

$$H\left(u_i\right) \simeq H\left(u_i^{\mathcal{M}}\right) + \mathbf{H}\delta u_i \tag{7}$$

denotes the linearised observational and model operators evaluated at $u_i = u_i^{\mathcal{M}}$ where $\mathbf{H}$ is the Hessian of H.

In eq. (5), the minimisation problem is defined on the field of increments [8]. In order to avoid the inversion of $\mathbf{B}_i$, as $\mathbf{B}_i = \mathbf{V}_i\mathbf{V}_i^T$, the minimisation is computed with respect to a new variable [17] $w_i = \mathbf{V}_i^+\delta u_i$, where $\mathbf{V}_i^+$ denotes the generalised inverse of $\mathbf{V}_i$, yielding to:

$$\begin{aligned} w_i^{ADD-DA} &= argmin_{w_i} J_i(w_i) = \\ &= argmin_{w_i}\left\{\frac{1}{2}w_i^T w_i + \frac{1}{2}(\mathbf{H}\mathbf{V}_i w_i - d_i)^T\mathbf{R}_i^{-1}(\mathbf{H}\mathbf{V}_i w_i - d_i)\right\} \end{aligned} \tag{8}$$

The ADD-DA process, on each sub-domain of the partition, is described in Algorithm 1.

Algorithm 1: ADD-DA

Input: v_i and $u_i^{\mathcal{M}}$

Define H_i ▷ interpolation operator

Compute $d_i \leftarrow v_i - H_i u_i^{\mathcal{M}}$ ▷ compute the misfit

Define R_i ▷ covariance matrix of the observed data v_i

Define V_i ▷ deviance matrix of background data

Define the initial value of δu_i^{ADD-DA}

Compute $V_i \leftarrow TSVD(V_i, m)$ ▷ m is the truncation parameter

Compute $w_i \leftarrow V_i^T \delta u_i^{ADD-DA}$

while *Convergence on* w_i *is obtained* **do**

  Compute $J_i \leftarrow J_i(w_i)$

  Compute $gradJ_i \leftarrow \nabla J_i(w_i)$

  Compute new values for w_i ▷ L-BFGS step

end

Compute $u_i^{ADD-DA} \leftarrow u_i^{\mathcal{M}} + V_i w_i$

Output: u^{ADD-DA}

We prove that the solution computed on this partitioning does not affect the accuracy of the DA process as the solution of the ADD-DA problem is the same than the DA algorithm coupled with Fluidity in [4]. In the following, w^{DA} denotes the solution of the DA process computed without any decomposition [4], i.e. defined on the whole domain Ω and such that:

$$w^{DA} = argmin_w J(w) = argmin_w\left\{\frac{1}{2}w^T w + \frac{1}{2}(\mathbf{H}\mathbf{V}w - d)^T\mathbf{R}^{-1}(\mathbf{H}\mathbf{V}w - d)\right\} \tag{9}$$

The functionals $J_i(w_i)$ (for $i = 1, \ldots, s$) defined in eq.(8) are restrictions of the functional $J(w)$ in eq.(9), i.e. $J(w)/\Omega_i = J_i(w_i)$. It has also to be noted that the functionals $J(w)$ and $J_i(w_i)$ are convex [6]. Hence, the following result helds.

Theorem 1. *Let w^{DA} be the solution of the DA process computed without any decomposition and let w_i^{ADD-DA} be the solution of ADD-DA as defined in eq.(3), we have:*

$$w^{DA}/\Omega_i = w_i^{ADD-DA}, \ \forall i = 1, \ldots, s. \tag{10}$$

Proof*: As w^{DA} is the minimum of $J(w)$ as defined in eq.(9), it yields:*

$$\nabla J(w^{DA}) = 0 \Leftrightarrow \nabla J/\Omega_i \left(w^{DA}/\Omega_i\right) = 0, \ \forall i = 1, \ldots, s$$

As $J/\Omega_i = J_i$, we have:

$$\nabla J_i \left(w^{DA}/\Omega_i\right) = 0, \ \forall i = 1, \ldots, s$$

As J_i is a convex function, the minimum is unique. Then the eq.(10) is satisfied.

The eq.(10) ensures that the accuracy obtained by the decomposition is maintained.

In Section 3, we validate the results provided in this section. We also show that the ADD approach reduces the execution time even if the implementation is not on a parallel computing environment.

3 Experimental test case

This work uses the three dimensional incompressible Navier-Stokes equations: continuity of mass (eq. (11)) and momentum equations (eq. (12)) as the full physical system.

$$\nabla \cdot \mathbf{u} = 0, \tag{11}$$

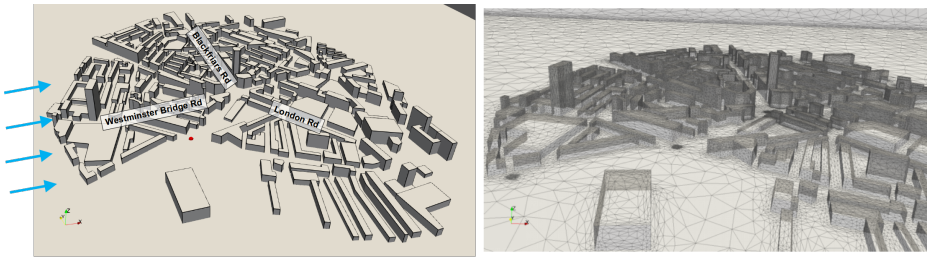
$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla p + \nabla \cdot \tau \tag{12}$$

where $\mathbf{u} \equiv (u, v, w)^T$ is the velocity, $p = \tilde{p}/\rho_0$ is the normalised pressure ($\tilde{p}$ being the pressure and ρ_0 the constant reference density) and τ denotes the stress tensor. Further details of the equations solved and their implementation can be found in [11,5,1]. The dispersion of the pollution is described by the classic advection-diffusion equation such that the concentration of the pollution is seen as a passive scalar (eq. (13)).

$$\frac{\partial c}{\partial t} + \nabla.(\mathbf{u}c) = \nabla. \left(\overline{\overline{\kappa}} \nabla c\right) + F \tag{13}$$

where $\overline{\overline{\kappa}}$ is the diffusivity tensor (m^2/s) and F represents the source terms ($\mathrm{kg}/\mathrm{m}^3/\mathrm{s}$), i.e. the pollution generated by a source point for example.

The capability of ADD-DA has been estimated using a realistic case representing a real urban area located in London South Bank University (LSBU) in Elephant and Castle, South London, UK (Fig. 2). The computational domain includes 767,559 nodes (Fig. 2b). In air pollution problems, we are interested in optimising the concentration field of the pollutant as well as the spread of it into the domain. In this work, a point source of pollution, mimicking pollution generated by traffic in a busy intersection, is located into the domain (red sphere in Fig. 2a) with a source term equal to $1kg/m^3/s$ and the dispersion behaviour of it is simulated for a westerly wind (blue arrows in Fig. 2a). Observed values of the state variable are provided by sensors from positions randomly located among the buildings.



(a) (b)

Fig. 2: (a) Computational domain and (b) surface mesh of the test site: the London South Bank University (LSBU), London (UK) area. In (a) the red sphere denotes the location of the source and the blue arrows the wind direction.

ADD-DA combined with Fluidity is a fully-parallel program, using the Message Passing Interface (MPI) library to communicate information between processors. The *supermesh* is decomposed into sub-domains **.vtu* files and **.halo* files, the latest ones containing information on overlapping regions. The **.vtu* files are read in the ADD-DA algorithm. In each of these sub-domains, ADD-DA computes the background covariance matrices (Step 4 in Algorithm 1) and computes the solution of the assimilation process (Steps 9-13 in Algorithm 1). Algorithm 1 has been implemented and tested on 3 high performance nodes equipped with bi-Xeon E5-2650 v3 CPU and 250GB of RAM.

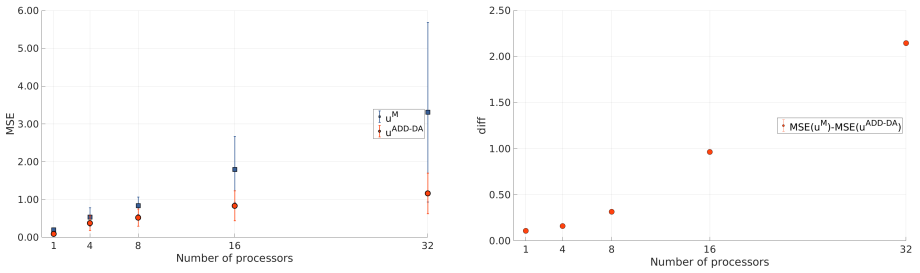
The accuracy of the ADD-DA results is evaluated by the mean squared error on each subdomain:

$$MSE(u_i) = \frac{\|u_i - u_i^C\|_{L^2}}{\|u_i^C\|_{L^2}} \tag{14}$$

computed with respect to a control variable u_i^C, for $i = 1, \ldots, s$ and s still denotes the number of sub-domains. The global mean squared error is then defined by: $MSE(u) = mean_i\{MSE(u_i)\}$.

Fig. 3a shows the values of $MSE(u^{\mathcal{M}})$ and $MSE(u^{ADD-DA})$ as a function of the number of sub-domains which constitute the decomposition. The sub-domains are labeled by the ID of processors p, i.e. $s = p$. For each domain decomposition

made of $p = 4, 8, 16, 32$ sub-domains, the value of $MSE(u^{\mathcal{M}})$ is greater than the $MSE(u^{ADD-DA})$. It has also confirmed in Fig. 3b which shows the values of the difference between $MSE(u^{\mathcal{M}})$ and $MSE(u^{DA})$.



(a) (b)

Fig. 3: (a) MSE of model background ($u^{\mathcal{M}}$) and ADD-DA (u^{ADD-DA}) as a function of the number of processors. The markers depict the MSE, and the error bars their standard deviations, for the assimilation of pollutant concentration, and (b) Error reduction as a function of the number of processors.

We evaluated the execution time needed to compute the solution of the ADD-DA model by Algorithm 1. Let $T_s(n)$ denotes the execution time of the Algorithm 1 for a domain decomposition made of s sub-domains. We still assume that $p = s$, where p denotes the number of processors and we pose

$$T_s(n) = max\{T_{s_i}(n_i)\}_{i=1,\dots,s} \tag{15}$$

where $T_{s_i}(n_i)$ denotes the execution time for each processor on each sub-domain. The total execution time is shown in Table 1. There is a clear decreasing trend in the total execution time with the increase of number of processors. Is is also confirmed by the values of speed-up computed as $S_s = T_1(n)/T_s(n)$.

p	T_p (seconds)	S_s	$\hat{S}_p$
1	2.80×10^3	-	-
4	8.04×10^2	3.48	4
8	3.63×10^2	7.71	8
10	3.06×10^2	9.15	10
16	1.96×10^2	14.3	16
32	3.35×10^2	11.9	32

Table 1: Execution times for $s = p$.

As described in the Fluidity manual [1], it is suggested to have at least $50,000$ nodes per processor in Fluidity to have full advantage of the parallelisation. It is due to the surface-to-volume ratio [2] which becomes too big for small sub-domains. As the number of nodes in our simulation is $n = 767,559$, the optimal number of processors to use to run Fluidity is supposed to be less than

$p = \frac{767{,}559}{50{,}000} \simeq 15$. The optimal number of processors computed and suggested by Fluidity [1], for the computational domain in Fig. 2b, is made of $p = 10$ sub-domains. This constraint affects the ADD-DA execution time as confirmed by the results of speed-up in Table 1. In fact, the total execution time of ADD-DA using 10 processors is 306 seconds, while using one processor is 2800 seconds with a speed-up of 9.15. Increasing the number of processors, we start to lose gain in terms of speed-up compared with the theoretical one $\hat{S}_p = p$. We observed a reduction of the execution time even if ADD-DA implements a decomposition of s sub-domains but Algorithm 1 runs on one processor, i.e. $s \neq p$. In fact, we tested ADD-DA implementing a First In First Out (FIFO) queue processing the sub-domains of the decomposition on $p = 1$ processor and we have seen that the total execution time of ADD-DA for a decomposition of $s = 32$ sub-domains is 9.48×10^2 seconds. Even if the gain in terms of Speed-Up is only 2.95, this result underlines the gain we have in introducing Adaptive Domain Decomposition on top of the math stack, i.e. in the mathematical model. This results is due to the complexity of the numerical model which decreases when we introduce the ADD. In fact, the time complexity of Algorithm 1 is $\tau(n) \simeq m \times n^2$ where m is the truncation parameter for the TSVD (Step 6 of Algorithm 1) and n is the number of points of the *supermesh*. If we assume that the time needed to perform $\tau(n)$ floating point operations is $T(n) = \tau(n) \times t_{flop}$, where t_{flop} denotes the unitary time required for the execution of one floating point operation, we have in our case

$$T(n) = m \times n^2 \times t_{flop} \tag{16}$$

Assumed that $n = \sum_{i=1}^{s} n_i$ and assuming a fixed truncation parameter m, the (16) gives:

$$T(n) = m \times \left(\sum_{i=1}^{s} n_i \right)^2 \times t_{flop} \tag{17}$$

Due to the properties of the square of a polynomial, the (17) gives

$$T(n) = m \times \left(\sum_{i=1}^{s} n_i \right)^2 \times t_{flop} > m \times \sum_{i=1}^{s} n_i^2 \times t_{flop} = \tag{18}$$

$$= \sum_{i=1}^{s} m \times n_i^2 \times t_{flop} = \sum_{i=1}^{s} T_{s_i}(n_i) > max\{T_{s_i}(n_i)\} = T_s(n)$$

which gives $T_s(n) < T(n)$ where $T_s(n)$ still denotes the execution time defined in (15) and $T(n)$ is the execution time when the algorithm does not implement any decomposition.

Fig. 4 shows the impact of ADD-DA on the iso-surface of the pollutant concentration for $5.10^{-1} kg/m^3$ computed in parallel with $p = 10$ processors and generated by a point source. Fig. 4a shows the results predicted by Fluidity, i.e. $u^{\mathcal{M}}$, while Fig. 4b shows the observed data, i.e. v. Values v are assimilated in parallel by ADD-DA to correct the forecasting data $u^{\mathcal{M}}$. The assimilated data after the ADD-DA process, i.e. u^{ADD-DA}, are then obtained (Fig. 4c).

(a) $u^{\mathcal{M}}$: predicted pollutant concentration field

(b) v: observed pollutant concentration field

(c) u^{DA}: assimilated pollutant concentration field

Fig. 4: Iso-surface, in white, of the pollutant concentration for $5.10^{-1} kg/m^3$ computed in parallel with $p = 10$ and generated by a point source.

Conclusions and future works are presented in next section where we propose the next steps towards the development of a scalable data assimilation software for accurate air pollution prediction in big cities.

4 Conclusions and future work

We presented a parallel Data Assimilation model based on Adaptive Domain Decomposition (ADD-DA) coupled with the open-source, finite-element, fluid dynamics model Fluidity. The model is defined on a partition of the domain in sub-domains without overlapping regions. We provided experimental results for pollutant dispersion within an urban environment. We proved that the ADD-DA model provides the same numerical solution of the sequential model. We have also shown that the ADD approach reduces the execution time even when the implementation is not on a parallel computing environment. An implementation of ADD-DA for improving air pollution prediction in big boroughs of London (as the one shown in Fig. 5) has been developed as future work. In that case we are implementing a multi-level parallelism. Starting from the Adaptive Domain Decomposition, the sub-domains are distributed on parallel processing units and, each sub-domain, is decomposed in sub-sub-domains to implement a First In First Out (FIFO) queue as we have seen it provides a further improvements in terms of reduction of the execution time.

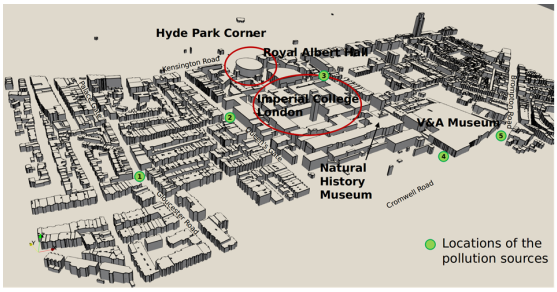


Fig. 5: Computational domain in the South West London area. The corresponding surface mesh is made of 12,9 millions of nodes.

Acknowledgments

This work is supported by the EPSRC Grand Challenge grant "Managing Air for Green Inner Cities" (MAGIC) EP/N010221/1 and by the EPSRC Centre for Mathematics of Precision Healthcare EP/N0145291/1.

References

1. AMCG, I.C.L.: Fluidity manual v4.1.12 (4 2015), `https://figshare.com/articles/Fluidity_Manual/1387713`
2. Arcucci, R., DAmore, L., Carracciuolo, G. Scotti, a.G.L.: A decomposition of the tikhonov regularization functional oriented to exploit hybrid multilevel parallelism. International Journal of Parallel Programming **45(5)**, 1214–1235 (2017)

3. Arcucci, R., D'Amore, L., Pistoia, J., Toumi, R., Murli, A.: On the variational data assimilation problem solving and sensitivity analysis. Journal of Computational Physics **335**, 311–326 (2017)
4. Arcucci, R., Mottet, L., Pain, C., Guo, Y.K.: Optimal reduced space for variational data assimilation. Journal of Computational Physics pp. 51 – 69 (2019)
5. Aristodemou, E., Bentham, T., Pain, C., Robins, A.: A comparison of mesh-adaptive les with wind tunnel data for flow past buildings: mean flows and velocity fluctuations. Atmospheric Environment journal **43**, 6238–6253 (2009)
6. Asch, M., Bocquet, M., Nodet, M.: Data assimilation: methods, algorithms, and applications, vol. 11. SIAM (2016)
7. Christie, M.A., Glimm, J., Grove, J.W., Higdon, D.M., Sharp, D.H., Wood-Schultz, M.M.: Error analysis and simulations of complex phenomena. Los Alamos Science (29) (2005)
8. Courtier, J.: A strategy for operational implementation of 4d-var, using an incremental approach. Q J R Meteorol Soc **120**(519), 1367–1387 (1994)
9. DAmore, L., Arcucci, R., Marcellino, L., Murli, A.: A parallel three-dimensional variational data assimilation scheme. In: AIP Conference Proceedings. vol. 1389, pp. 1829–1831. AIP (2011)
10. Farina, R., Dobricic, S., Storto, A., Masina, S., Cuomo, S.: A revised scheme to compute horizontal covariances in an oceanographic 3d-var assimilation system. Journal of computational physics **284**, 631–647 (2015)
11. Ford, R., Pain, C.C., Goddard, A.J.H., De Oliveira, C.R.E., Umpleby, A.P.: A non-hydrostatic finite-element model for three-dimensional stratified oceanic flows. part I: Model formulation. Monthly Weather Review **132**, 2816–2831 (2004)
12. Hannachi, A.: A primer for eof analysis of climate data. Department of Meteorology, University of Reading, UK (2004)
13. Hannachi, A., Jolliffe, I., Stephenson, D.: Empirical orthogonal functions and related techniques in atmospheric science: A review. International Journal of Climatology: A Journal of the Royal Meteorological Society **27**(9), 1119–1152 (2007)
14. Kalman, R.E.: A new approach to linear filtering and prediction problems. Journal of basic Engineering **82**(1), 35–45 (1960)
15. Kalnay, E.: Atmospheric modeling, data assimilation and predictability. Cambridge (2003)
16. Lelieveld, J., Evans, J.S., Fnais, M., Giannadaki, D., Pozzer, A.: The contribution of outdoor air pollution sources to premature mortality on a global scale. Nature **525**(7569), 367 (2015)
17. Lorenc, A.: Development of an operational variational assimilation scheme. Journal of the Meteorological Society of Japan **75**, 339–346 (1997)
18. Lorenz, E.N.: Empirical orthogonal functions and statistical weather prediction (1956)
19. Nichols, N.: Mathematical concepts in data assimilation. W. Lahoz, et al. (Eds.), Data Assimilation, Springer (2010)
20. Oke, P.R., Brassington, G.B., Griffin, D.A., Schiller, A.: The bluelink ocean data assimilation system (bodas). Ocean Modelling **21**(1-2), 46–70 (2008)
21. Pain, C., Umpleby, A., De Oliveira, C., Goddard, A.: Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations. Computer Methods in Applied Mechanics and Engineering **190**, 3771–3796 (2001)
22. Teruzzi, A., Di Cerbo, P., Cossarini, G., Pascolo, E., Salon, S.: Parallel implementation of a data assimilation scheme for operational oceanography: The case of the medbfm model system. Computers & Geosciences **124**, 103–114 (2019)