COMMUNICATIONS IN NUMERICAL METHODS IN ENGINEERING
Commun. Numer. Meth. Engng 2004; **20**:639–646 (DOI: 10.1002/cnm.695)

Dynamic deactivation for advection-dominated contaminant transport

Rainald Löhner*,† and Fernando Camelli

School of Computational Sciences, MS 4C7, George Mason University, Fairfax, VA 22030-4444, U.S.A.

SUMMARY

A simple dynamic deactivation procedure for advection-dominated contaminant transport is presented. The key idea is to avoid any work in regions where the solution cannot change, i.e. where sources vanish and unknowns do not exhibit any spatial change. Saving factors in CPU of 1:3–1:10 are commonly achieved. The procedure is general and simple to implement, and should be applicable to a large number of codes and transport/dispersion problems. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: contaminant transport; advection–diffusion; finite elements; CFD

1. INTRODUCTION

The intentional or unintentional release of hazardous materials can lead to devastating consequences. In order to prepare effective countermeasures, place sensors or legislate, it is imperative to know the maximum possible damage a release in a certain region can have. Given the many possible boundary conditions (wind direction, atmospheric conditions, humidity, temperature of building walls, emissions of heat exchangers, etc.), a large number of simulations are required [1–6]. Typically, flowfields are precomputed and stored. Different release scenarios (locations, type of release, etc.) are then simulated for these pre-stored flowfields. While the calculation of the (incompressible) flowfields can consume days or weeks of supercomputer time, each one of the release scenarios can be computed in a matter of minutes or hours on present-day PC platforms. This should come as no surprise, given that flowfields require five unknowns at each gridpoint (pressure, velocities, temperature), as well as the solution of the pressure-Poisson system, whereas the release scenario only requires one unknown at each gridpoint (concentration), and no stiff implicit system needs to be solved each time step.

*Correspondence to: R. Löhner, School of Computational Sciences, MS 4C7, George Mason University, Fairfax, VA 22030-4444, U.S.A.
†E-mail: rlohner@gmu.edu

Contract/grant sponsor: DTRA

Received 22 August 2003
Accepted 11 February 2004

Taking into account that a change in concentration can only occur in regions where concentration changes spatially and sources are present opens the possibility of speeding up the release calculations. The key idea is to update only those regions where concentration changes spatially and sources are present, avoiding unnecessary work in the rest of the field. This last possibility has been explored here, achieving speed-ups of 1:3–1:10 for realistic cases.

The remainder of the paper is organized as follows: after describing the partial differential equations (PDEs) used, the basic elements of the solver are given. The deactivation criteria are then defined. Numerical examples and timings follow. Finally, some conclusions are drawn and possible future work is outlined.

2. BASIC ELEMENTS OF SOLVER

Any contaminant transport or release simulation solves the classic advection–diffusion equation

$$c_{,t} + \mathbf{v} \cdot \nabla c = \nabla k \nabla c + S \tag{1}$$

Here $c, \mathbf{v}, k$ denote the concentration, velocity and diffusivity of the medium, S the source-term, and $c_{,t}$ the derivative of c with respect to time. The (in most cases unsteady) velocity field $\mathbf{v}$ is assumed to be divergence free ($\nabla \cdot \mathbf{v} = 0$) and precomputed, i.e. given as an input. The spatial discretization of the computational domain is performed with tetrahedral elements. Denoting by N^i the shape-function of point i, the Galerkin weighted residual method for Equation (1) results in

$$\mathbf{M} \cdot \mathbf{c}_{,t} + \mathbf{A} \cdot \mathbf{c} = -\mathbf{K} \cdot \mathbf{c} + \mathbf{S} \tag{2}$$

with

$$\mathbf{M} = \int N^i N^j \, \mathrm{d}\Omega, \ \mathbf{A} = \int N^i \mathbf{v} \cdot \nabla N^j \, \mathrm{d}\Omega, \ \mathbf{K} = \int \nabla N^i k \nabla N^j \, \mathrm{d}\Omega, \ \mathbf{S} = \int N^i S \, \mathrm{d}\Omega \tag{3}$$

In most instances, the Galerkin weighted residual method will not yield (physically correct) monotone results for advection-dominated cases. A number of schemes have been devised to replace the 'Galerkin fluxes' by the so-called 'numerically consistent fluxes'. Upwinding [7–9], anisotropic balancing dissipation [10, 11] and flux-corrected transport (FCT) [12, 13] are examples of such schemes. In what follows, it is irrelevant which one of these schemes is chosen.

Given that one is interested in high temporal fidelity, and that the diffusive terms are typically very small, explicit time-integration schemes are used for Equation (2). Without loss of generality, consider the following m-stage Runge–Kutta scheme to go from time step n to $n+1$:

$$\mathbf{M} \cdot \mathbf{c}^i = \mathbf{r}^i = \mathbf{M} \cdot \mathbf{c}^0 + \alpha^i \Delta t (\mathbf{S} - (\mathbf{A} + \mathbf{K}) \cdot \mathbf{c}^{i-1}), \quad i = 1, m \tag{4}$$

where, $\mathbf{c}^0 = \mathbf{c}^n$, $\alpha^i = 1/(m + 1 - i)$ and $\mathbf{c}^{n+1} = \mathbf{c}^m$. The bulk of the CPU requirements for a scheme of this kind is in the evaluation of the right-hand sides $\mathbf{r}^i$. These required matrix–vector multiplications can be performed in a variety of ways. The simplest is by performing loops over the elements, evaluating all matrix–vector products at the element level [14–16].

A more efficient way to accomplish this for low-order elements can be achieved by switching to an edge-based data structure [16]. Finally, one can also store the matrices in some optimized way (e.g. via sparse matrix format), and perform the matrix–vector products directly [14, 15].

3. DEACTIVATION

Consider again the advection–diffusion equation given by Equation (1). A change in c can only occur in those regions where

$$|S| > 0, \quad |\nabla c| > 0 \tag{5}$$

For the typical contaminant transport problem, the extent of the regions where $|S| > 0$ is very small. In most of the regions that lie upwind of a source, $|\nabla c| = 0$. This implies that in a considerable portion of the computational domain no contaminant will be present, i.e. $c = 0$. The basic idea of deactivation is to identify the regions where no change in c can occur, and to avoid unnecessary work in them.

The marking of deactive regions is accomplished in two loops over the elements. The first loop identifies in which elements sources are active, i.e. where $|S| > 0$. The second loop identifies in which elements a change in the values of the unknowns occurs, i.e. where $\max(c_{\mathrm{el}}) - \min(c_{\mathrm{el}}) > \varepsilon_u$, with ε_u a preset, very small tolerance. Once these active elements have been identified, they are surrounded by additional layers of elements which are also marked as active. This 'safety' ring is added so that changes in neighbouring elements can occur, and so that the test for deactivation does not have to be performed at every time step. Typically, 4–5 layers of elements are added. From the list of active elements, the list of active points is obtained. The addition of elements to form the 'safety' ring can be done in a variety of ways. If the list of elements surrounding elements or elements surrounding points is available, only local operations are required to add new elements. If these lists are not present, one can simply perform loops over the elements, marking points, until the number of 'safety layers' has been reached. In either case, it is found that the cost of these marking operations is small compared to the advancement of the transport equation. Depending on how one performs the evaluation of right-hand sides, the lists of active elements and points are used as follows:

- If the matrix–vector products are evaluated at the element level, computation is only performed for the active elements, and the RHS of the inactive points is set to zero.
- If the matrix–vector products are evaluated at the edge level, computation is only performed for the active edges, and the RHS of the inactive points is set to zero.
- If the matrix–vector products are evaluated using sparse matrix–vector multiplication, computation is only performed for the rows and columns of active points, and the RHS of the inactive points is set to zero.

Further gains in speed can be obtained by renumbering points and elements/edges in the predominant direction of the flow. The active points/elements/edges will then be grouped in a more compact form. This avoids a large portion of the unnecessary IF-tests for inactive points/elements/edges. In the present study this form of renumbering was not invoked.

4. EXAMPLES AND TIMINGS

The deactivation option was tested on several examples, of which two are included here. The results for the runs with and without deactivation are indistinguishable, so that no degradation in fidelity takes place. The examples are meant to illustrate the gains in performance one can realistically achieve. The particular code used (FEFLO), has repeatedly been benchmarked and compared to experiments [5, 6], and is routinely used for production runs. The (incompressible) flowfields were precalculated using a projection scheme with implicit treatment of viscosity and pressure, and an explicit multistage prediction of the advective terms [17]. These flowfields were stored and read in for the dispersion calculation.

4.1. Building complex

The first example considers dispersion of pollutants from a continuous source around a series of buildings. The geometry, as well as the boundary conditions, are shown in Figure 1(a). A logarithmic wind profile with mean velocity of $v = 2$ m/s at a height of $h = 10$ m was assumed, the vertical distribution of the velocity given by $u = (u_*/\kappa)\ln(z/z_0)$, where u_* is the friction velocity, $\kappa = 0.4$ is the von Karman constant, z is the vertical height and $z_0 = 0.001$ is the roughness coefficient. For turbulence, the Smagorinsky model with Law of the Wall was employed.

The spatial discretization of the domain consisted of approximately 160 Kpts and 835 Kels. A detail of the surface mesh is shown in Figure 1(b). The resulting velocity field at the surface is shown in Figure 1(c). For the dispersion simulation, the flow was assumed steady, and a 5-stage Runge–Kutta scheme was used with a Courant-nr. of $C = 1.0$. The dispersion calculation was run for 200 s of real time on a PC with the following characteristics: Intel-P4 chip running at 2.2 GHz, 1 Gbyte of RAM, Linux operating system, Intel compiler. Figure 1(d) shows the resulting iso-surface of concentration for time $t = 200$ s at a level of $u = 0.01$. Note that at this time the cloud of dispersed material has lifted from the ground, greatly reducing the possible lethality. It is for this reason that a time of $t = 200$ s was deemed sufficient for the simulation.

Deactivation checks were performed every five time steps. The tolerance for deactivation was set to $\varepsilon_u = 10^{-3}$. With this tolerance, towards the end of the run approximately a third of the elements were active. This is reflected in the timings: the usual run (i.e. without deactivation) took $T = 808$ s, whereas the run with deactivation took $T = 275$ s, i.e. a saving of 1:2.94.

4.2. Subway station

The second example considers the dispersion of an instantaneous release in the side platform of a generic subway station. The geometry is shown in Figure 2(a).

A time-dependent inflow is applied on one of the end sides:

$$v(t) = b(t - 60)^3 \mathrm{e}^{-a(t-60)} + v_0$$

where $b = 0.46$ m/s, $a = 0.5\ \mathrm{s}^{-1}$, and $v_0 = 0.4$ m/s. This inflow velocity corresponds approximately to the velocities measured at a New York City subway station [18]. As before, the Smagorinsky model with Law of the Wall was used for this example. The volume grid had

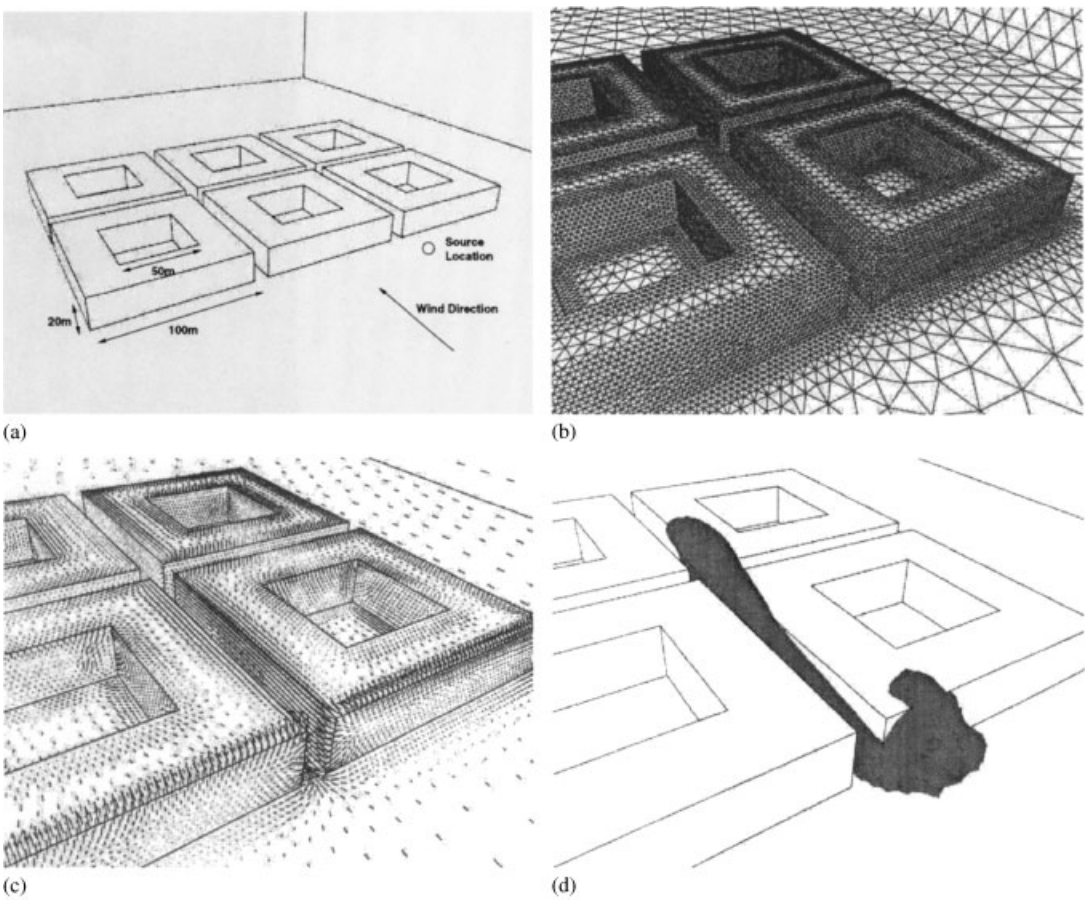
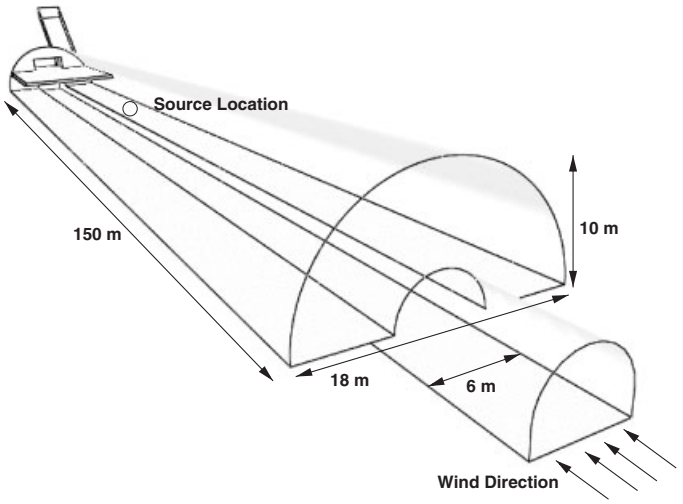


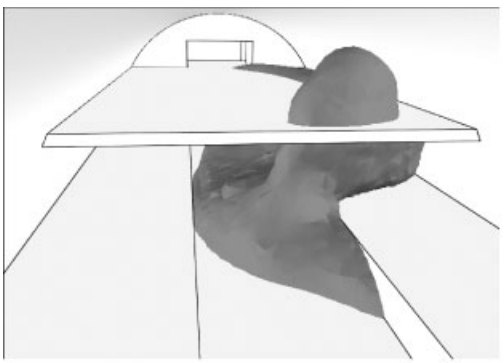
Figure 1. (a, b) Problem definition and detail of surface mesh; and (c, d) surface velocities and iso-surface of concentration.

730 Kels and 144 Kpts. The dispersion simulation was performed using a three-stage Runge–Kutta scheme with a Courant-nr. of $C = 0.6$. The dispersion calculation was run for 485 s of real time (corresponding to the time of a train entering, halting, exiting the station and the time for the next train) on a workstation with the following characteristics: Dec Alpha chip running at 0.67 GHz, 4 Gbyte of RAM, Linux operating system, Compaq compiler. Figures 2(b)–(e) show the resulting iso-surface of concentration level $c = 0.0001$, as well as the surface velocities for time $t = 485$ s. Note the transient nature of the flowfield, which is reflected in the presence of many vortices.

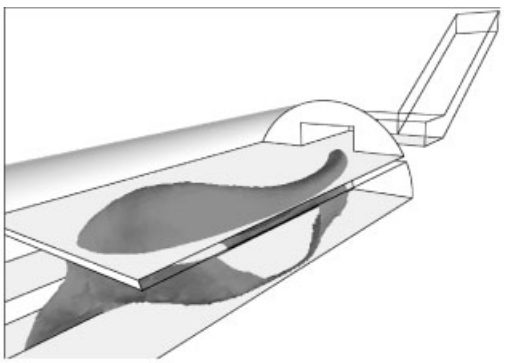
Deactivation checks were performed every five time steps. The tolerance for deactivation was set to $\varepsilon_u = 10^{-3}$. The usual run (i.e. without deactivation) took $T = 5,296$ s, whereas the run with deactivation took $T = 526$ s, i.e. a saving in excess of 1:10. This large reduction in computing time was due to two factors: the elements with the most constraining time step are located at the entry and exit sections, and the concentration cloud only reaches this



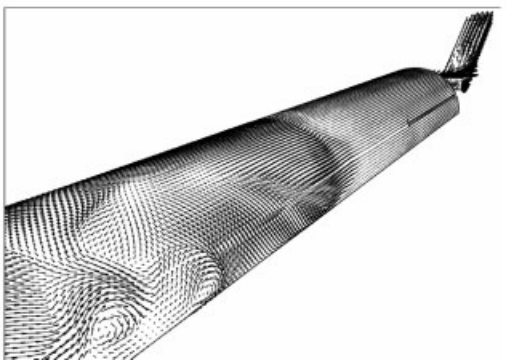
(a)



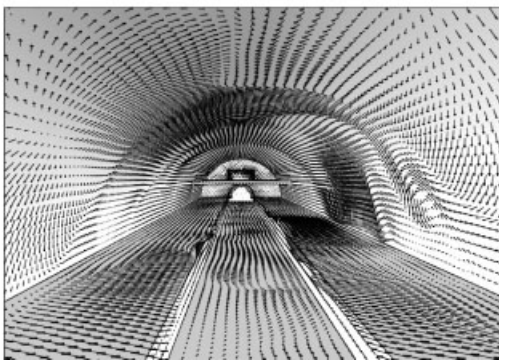
(b)



(c)



(d)



(e)

Figure 2. (a) Problem definition; (b, c) iso-surface of concentration $c = 0.0001$; and (d, e) surface velocities.

zone very late in the run (or not at all); furthermore, as this is an instantaneous release, the region of elements where concentration is present in meaningful values is always very small as compared to the overall domain.

5. CONCLUSIONS AND OUTLOOK

A simple dynamic deactivation procedure for advection-dominated contaminant transport has been presented. The key idea is to perform computational work only where necessary, avoiding any work in regions where sources vanish and unknowns do not exhibit any spatial change. Saving factors in CPU of 1:3–1:10 are commonly achieved. Two typical examples runs have been included.

The procedure is general and simple to implement, and should be applicable to a large number of codes and and transport/dispersion problems. Dynamic deactivation works particularly well for scalar processors, such as those encountered in PCs or workstations. Given that a large number of parameter scoping and genetic algorithm calculations are currently being offloaded to processors of this kind, reductions in computing time of 1:3–1:10 represent significant savings.

ACKNOWLEDGEMENTS

This work was partially supported by DTRA under the auspices of CAMP.

REFERENCES

1. Hanna SR, Briggs GA, Hosker RP. Handbook on atmospheric diffision. *NOAA DOE/TIC*-11223, 1982.
2. Stern AC, Boudel RW, Turner DB, Fox DL. *Fundamentals of Air Pollution*. Academic Press: New York, 1984.
3. Arya SP. *Introduction to Micrometeorology*. Academic Press: New York, 1998.
4. Arya SP. *Air Pollution Meteorology and Dispersion*. Oxford University Press: Oxford, 1999.
5. Camelli F, Löhner R. Flow and dispersion around buildings: an application with FEFLO. *Proceedings of ECCOMAS 2000 Conference*, Barcelona, Spain, September, 2000.
6. Hanna SR, Tehranian S, Carissimo B, Macdonald RW, Löhner R. Comparisons of model simulations with observations of mean flow and turbulence within simple obstacle arrays. *Atmospheric Environment* 2002; **36**:5067–5079.
7. van Leer B. Towards the ultimate conservative scheme. II. Monotonicity and conservation combined in a second order scheme. *Journal of Computational Physics* 1974; **14**:361–370.
8. Sweby PK. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM Journal on Numerical Analysis* 1984; **21**:995–1011.
9. Hirsch C. *Numerical Computation of Internal and External Flow*. Wiley: New York, 1991.
10. Kelly DW, Nakazawa S, Zienkiewicz OC, Heinrich JC. A note on anisotropic balancing dissipation in finite element approximation to convection diffusion problems. *International Journal for Numerical Methods in Engineering* 1980; **15**:1705–1711.
11. Brooks AN, Hughes TJR. Streamline upwind/petrov Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1982; **32**:199–259.
12. Parrott AK, Christie MA. FCT applied to the 2-D finite element solution of tracer transport by single phase flow in a porous medium. *Proceedings of ICFD-Conference on Numerical Methods in Fluid Dynamics*, Morton KW, Baines (eds). Academic Press: Reading, 1986.
13. Löhner R, Morgan K, Peraire J, Vahdati M. Finite element flux-corrected transport (FEM-FCT) for the Euler and Navier–Stokes equations. *International Journal for Numerical Methods in Fluids* 1987; **7**:1093–1109.
14. Bathe KJ. *Finite Element Procedures*. Prentice-Hall: Englewood Cliffs, NJ, 1995.
15. Zienkiewicz OC, Taylor R. *The Finite Element Method* (5th edn). Elsevier: Amsterdam, 2000.
16. Löhner R. *Applied CFD Techniques*. Wiley: New York, 2001.

17. Löhner R, Chi Yang, Cebral JR, Soto O, Camelli F, Waltz J. Improving the speed and accuracy of projection-type incompressible flow solvers. *AIAA*-03-3991-CP, 2003.
18. Pflistch A, Kleeberger M, Küsel H. On the vertical structure of air flow in the subway New York City and Dortmund (Germany). *Proceedings of 4th Annual George Mason University of Transport and Dispersion Modeling Workshop*, Fairfax, VA, July 2000.

