INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2006; **52**:1–29
Published online 10 February 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1082

A filtering technique for system of reaction–diffusion equations

F. Dupros[1], M. Garbey[1,*,†] and W. E. Fitzgibbon[2]

[1]*Department of Computer Science, University of Houston, Houston, TX, U.S.A.*
[2]*Department of Mathematics, University of Houston, Houston, TX, U.S.A.*

SUMMARY

In this paper, we present a fast parallel solver designed for a system of reaction–convection–diffusion equations. A typical application is the large-scale computing of air quality models for which the main solver corresponds to reaction–diffusion–convection equations. Another potential application is the numerical simulation of population models where several colonies compete. Reaction–diffusion systems can be integrated in time by pointwise Newton iteration when all space-dependent terms are explicit in the time integration. A Newton–Jacobi iteration makes this scheme implicit. Such methods are easy to code and have scalable parallelism. However, they are numerically inefficient. An alternative method is to use operator splitting, decoupling the time integration of the reaction from the time integration of convection–diffusion. However, such methods may not be time accurate thanks to the stiffness of the reaction term and are complex to parallelize with good scalability. A second alternative is to use matrix-free Newton–Krylov methods. These techniques are particularly efficient provided that a good parallel preconditioner is customized to the application. The method is then not trivial to implement. We propose here a new family of fast, easy to code and numerically efficient reaction–diffusion solvers based on a filtering technique that stabilizes the explicit treatment of the diffusion terms. The scheme is completely explicit with respect to space, and the postprocessing to stabilize time stepping uses a simple FFT. We demonstrate the potential of this numerical scheme with two examples in air quality models that usually require the implicit treatment of diffusion terms and have compared our solution to classical schemes for two nonlinear reaction–diffusion problems. For general reaction–diffusion problems on tensorial product grids with regular space step, the filtering process can be applied as a black box postprocessing procedure. Further, we demonstrate on critical components of the algorithm the high potential of parallelism of our method on medium-scale parallel computers. Copyright © 2006 John Wiley & Sons, Ltd.

KEY WORDS: ODE; filtering; reaction–diffusion; air quality

*Correspondence to: M. Garbey, Department of Computer Science, University of Houston, Houston, TX, U.S.A.
†E-mail: garbey@cs.uh.edu

Contract/grant sponsor: NSF; contract/grant number: ACI-0305405
Contract/grant sponsor: US EPA; contract/grant number: CR829068-01

Received 23 March 2005
Revised 14 July 2005
Accepted 18 July 2005

1. INTRODUCTION

We consider application with the main solver corresponding to reaction–diffusion–convection system:

$$\frac{\partial C}{\partial t} = \nabla \cdot (K \nabla C) + (\mathbf{a} \cdot \nabla) C + F(t, x, C) \tag{1}$$

with $C \equiv C(x,t) \in R^m$, $x \in \Omega \subset \mathbf{R}^3, t > 0$. A typical example is an air pollution model where $\mathbf{a}$ is the given wind field, and F is the reaction term combined with source/sink terms. For such a model m is usually very large, and the corresponding ODE system

$$\frac{\partial C}{\partial t} = F(t, x, C) \tag{2}$$

is stiff [1–4]. A second class of examples is the modelling of cancer tumor growth [5, 6].

Equation (1) can be rewritten as

$$\frac{\mathrm{D}C}{\mathrm{D}t} = \nabla \cdot (K \nabla C) + F(t, x, C) \tag{3}$$

where $\mathrm{D}/\mathrm{D}t$ represents the total derivative.

For the time integration of reaction–diffusion–convection, one can distinguish three different time scales. Let us denote $\mathrm{d}t$ the time step and h the minimum size of the mesh in all space directions. Let us assume $\|\mathbf{a}\| = O(1)$ and $\|K\| = O(1)$. First, the Courant–Friedrichs–Lewy (CFL) condition imposes $\mathrm{d}t = O(h)$ with the explicit treatment of the convective term. Second, the stability condition for the explicit treatment of the diffusion term gives $\mathrm{d}t = O(h^2)$. It is standard for reaction–diffusion–convection solver used in air quality to have a second-order scheme in space combined to a second-order scheme in time. We have then to look for an implicit scheme such that $\mathrm{d}t = O(h)$. It is critical to make the treatment of the diffusion term implicit, while it is not so critical for the convective terms. The discretization of the convective term might be done in a number of ways. We refer to Reference [4] for a review of these methods. The Ellam scheme is probably one of the best convective schemes [7]. In our work, we use the method of characteristics that provides a good combination of space–time accuracy, while it is fairly simple to code [8], straightforward to parallelize and implicit in time with no global data dependency in space. The third time scale in the reaction–diffusion–convection equation comes from the reactive source term that is usually stiff in air quality application, but not so stiff in bio-applications as in References [5, 6] and its references. One typically applies nonlinear ODE implicit schemes derived from the theory of (2).

The main problem we address in this paper is the design of a fast solver for reaction–diffusion that has good stability properties with respect to the time step but avoids the computation of the full Jacobian matrix. One classical scheme of work follows the so-called matrix-free Newton–Krylov methods. We refer to Reference [9] for a recent review of these methods. While these techniques can be particularly efficient, it is encouraged to use them in the framework of a package like PETSc to adapt the method to the reaction–diffusion application and optimize each parameter of the method. An alternative is to introduce an operator splitting that integrates a fast nonlinear ODE solver with an efficient linear solver for the diffusion(–convection) operator. However, the stiffness of the reaction terms induces some unusual misperformance problems for high-order operator splitting. In fact, the classical

splitting of Strang might perform less well than a first-order source splitting [10]. As reported recently [11], second-order splitting can also give spurious solution for reaction–diffusion systems that are not even particularly stiff.

We explore some alternative methodology in this paper that consists of stabilizing with *a posteriori* filtering, the explicit treatment of the diffusion term. The diffusion term is then an additional term in the fast ODE solver and the problem is completely parametrized by space dependency. The coding of the algorithm is particularly simple and requires only a postprocessing subroutine that relies on an FFT. The mathematical idea of our method was first introduced in Reference [12]. We will show that it can produce an efficient parallel algorithm due to the intense pointwise computation dominated by the time integration of the large system of ODE's. However, we will not address the issue of load balancing that is dictated by the integration of the chemistry [2, 13], and therefore application dependent. The stabilizing technique based on filtering presented in this paper will be limited to grids that can be mapped to regular space discretization or grids that can be decomposed into subdomains with regular space discretization. We should point out that an alternative and possibly complementary methodology to our approach is the so-called Tchebycheff acceleration [14, 15] and its references, that allows so-called super time steps that decompose into appropriate irregular time stepping.

We will demonstrate the potential of our numerical scheme with two examples in air quality models that usually require the implicit treatment of diffusion terms, that is local refinement in space via domain decomposition to improve accuracy around source points of pollution or stretched vertical space coordinate to capture better ground effect. For general reaction–diffusion problems on tensorial product grids with regular space step, the filtering process can be applied as a black box postprocessing procedure. Further, we demonstrate on critical components of the algorithm the high potential of parallelism of our method on medium-scale parallel computers.

The plan of this article is as follows. Section 2 presents the methodology for reaction–diffusion problem first in one space dimension and second its generalization to multidimensional problems along with some evaluation of the method using the benchmark problems in Reference [11]. Section 3 gives a two-dimensional example of a computation of a simplified ozone model with local grid refinement and basic diffusion. Section 4 gives a one-dimensional example of the same model with vertical irregular diffusion and strongly varying space step. Section 5 comments on the parallel implementation of the method and discusses the parallel efficiency following the preliminary results presented in Reference [16]. In Section 6, we discuss the potential of this method for grid computing and conclude.

2. METHODOLOGY

2.1. Fundamental observations on the stabilization of explicit scheme

In this section, we restrict ourselves to the scalar equation

$$\partial_t u = \partial_x^2 u + f(u), \quad x \in (0, L), \ \ t \in (0, T) \tag{4}$$

with boundary conditions to be specified later. We assume that the problem is well posed and has a unique solution.

We will consider the first-order semi-implicit Euler scheme:

$$\frac{u^{n+1}-u^n}{\mathrm{d}t} = D_{xx}u^n + f(u^{n+1}) \tag{5}$$

as well as the following second-order scheme in space and in time:

$$\frac{3u^{n+1}-4u^n+u^{n-1}}{2\,\mathrm{d}t} = 2\,D_{xx}u^n - D_{xx}u^{n-1} + f(u^{n+1}) \tag{6}$$

that is a combination of backward second-order Euler (BDF) for the time derivative and second-order extrapolation in time for the diffusion term. We recall that BDF is a standard scheme used for stiff ODEs [4, 17, 18]. We will also use the following second-order scheme:

$$\frac{u^{n+1}-u^n}{\mathrm{d}t} = \tfrac{3}{2}\,D_{xx}u^n - D_{xx}u^{n-1} + \tfrac{1}{2}\,(f(u^n)+f(u^{n+1})) \tag{7}$$

that uses the mid-point rule for the time derivative, and second-order extrapolation in time for the diffusion term.

We restrict ourselves to finite difference discretization with second-order approximation of the diffusion term. The Fourier transform of (6), for instance, when neglecting the nonlinear term has the form

$$\frac{3\hat{u}^{n+1}-4\hat{u}^n+\hat{u}^{n-1}}{2\,\mathrm{d}t} = \Lambda_k(2\hat{u}^n-\hat{u}^{n-1}) \tag{8}$$

where $\Lambda_k = 2/h^2(\cos(hk)-1)$. The stability condition for wave number k has the form

$$2\,\frac{\mathrm{d}t}{h^2}\left|\cos\left(\frac{k\pi}{N}\right)-1\right| < \tfrac{4}{3} \tag{9}$$

with $h=\pi/N$. The maximum time step allowed is then

$$\mathrm{d}t < \tfrac{1}{3}\,h^2 \tag{10}$$

However, it is only the high-frequency components of the solution that are responsible for such a time step constraint, and they are poorly handled by second-order finite differences. We observe numerically that the relative error on large frequencies for second-order derivatives of large frequencies waves $\cos(kx)$, $k \approx N$ with central differences grows up to 9%. Therefore, the main idea is to apply a filter that can remove the high frequencies in order to relax the constraint on the time step while keeping second-order accuracy in space. The main difficulty is then to cope with nonperiodic boundary conditions. With arbitrary Dirichlet boundary conditions, for example, the Fourier expansion of $u(.,t)$ has very bad convergence properties thanks to the Gibbs phenomenon in the neighbourhood of the end points of the interval $(0,L)$.

We recall that a real and even function $\sigma(\eta)$ is called a filter of order p if,

- $\sigma(0)=1, \quad \sigma^{(l)}(0)=0, \quad 1 \leqslant l \leqslant p-1,$
- $\sigma(\eta)=0 \quad \text{for } |\eta| \geqslant 1,$
- $\sigma(\eta) \in C^{p-1} \quad \text{for } \eta \in (-\infty,\infty).$

Let us denote $y=(2\pi/L)x$, and $v(y)=u(x)$. If

$$u(x,t)=\sum_{k=-\infty}^{\infty}\hat{v}_k(t)\mathrm{e}^{\mathrm{i}ky}$$

denotes the Fourier expansion of $u(x,t)$ at time t, then the filtered function u_N^σ is

$$u^\sigma(x,t)=\sum_{k=-\infty}^{\infty}\sigma\left(\frac{k\kappa}{N}\right)\hat{v}_k(t)\,\mathrm{e}^{\mathrm{i}ky}$$

and has nonzero Fourier components only for $|k|\leqslant N/\kappa$. The parameter $\kappa\geqslant 0$ will set the level of cut in frequency space. We will denote T the transform

$$u \overset{\mathscr{T}}{\to} u_N^\sigma$$

For a general time-integration scheme for (4), one can filter the solution after each time step and adjust the frequency cut, i.e. set κ such that the von Neumann necessary condition for stability [19] is satisfied. We first need to show that it is sufficient to ensure the stability of the scheme, and second that the high-frequency components left out from the Fourier expansion of the solution at each time step do not affect the numerical accuracy of the finite difference scheme.

We recall the following result [20] that will be useful to address the issue of the accuracy of our stabilized scheme.

Theorem 1
Let u be a piecewise C^p function with one point of discontinuity, ξ, and let σ be a filter of order p. For any point $y\in[0,2\pi]$, let $d(y)=\min\{|y-\xi+2k\pi| : k=-1,0,1\}$. If $u_N^\sigma=\sum_{k=-\infty}^{\infty}\hat{u}_k\sigma(k/N)\mathrm{e}^{\mathrm{i}ky}$,
then

$$|u(y)-u_N^\sigma|\leqslant CN^{1-p}(d(y))^{1-p}K(f)+CN^{1/2-p}\|u(p)\|_{L^2}$$

where

$$K(f)=\sum_{l=0}^{p-1}(d(y))^l|u^{(l)}(\xi^+)-u^{(l)}(\xi^-)|\int_{-\infty}^{\infty}|G_l^{(p-l)}(\eta)|\,\mathrm{d}\eta$$

and $G_l(\eta)=\frac{\sigma(\eta)-1}{\eta^l}$.

Thus, a discontinuity of $u(x)$ leads to a Fourier expansion with error $O(1)$ near the discontinuity and $O(1/N)$ away from the discontinuity.

We first present the stabilization of scheme (5) with periodic boundary conditions for the heat equation

$$\partial_t u=\partial_x^2 u+f(x,t),\quad x\in(0,L),\ \ t\in(0,T) \tag{11}$$

with periodic boundary conditions

$$u(0,t)=u(L,t),\quad \partial_x u(0,t)=\partial_x u(L,t), t\in(0,T) \tag{12}$$

and the initial condition

$$u(x,0)=u^o(x),\quad x\in(0,L) \tag{13}$$

2.2. The periodic case

Let us consider, for simplicity, a one-step scheme that can be written as

$$u_h^{n+1} = Q\, u_h^n + \mathrm{d}t\, f_h^n \tag{14}$$

where u_h^n is the grid solution at points x_j, $j = 1, \ldots, N$, and time t^n. The scheme is stable for the l^2 discrete norm, iff $\|Q\|_{l^2,h} < 1$.

For time steps $\mathrm{d}t$ such that $\|Q\|_{l^2,h} \geqslant 1$, we are going to filter out the high-frequency components of the solution u^{n+1} that are unstable.

Let P be the matrix of the discrete Fourier transform and P^{-1} its inverse. Let $\mathscr{D}_\sigma$ denotes the diagonal matrix of the transform T.

In Fourier space, the Euler scheme can be written as

$$\hat{u}_N^{n+1} = \hat{Q}\, \hat{u}_N^n + \mathrm{d}t\, \hat{f}_N^n$$

For the stability analysis, we can suppose that $f_h^n \equiv 0$. Stability in the discrete l^2 norm is equivalent to stability in Fourier space, i.e. $\|\hat{Q}\|_{l^2} < 1$.

Our stabilized scheme writes

$$u_h^n \overset{\mathscr{Q}}{\to} u_h^{*,n+1} \overset{\mathscr{P}}{\to} \hat{u}^{n+1} \overset{\mathscr{D}_\delta}{\to} \hat{u}^{n+1} \overset{\mathscr{P}^{-1}}{\to} u^{n+1}$$

The filter $\mathscr{D}_\sigma$ cuts out all frequencies that do not satisfy the von Neumann criterion of stability: we have then $\|\mathscr{D}_\sigma \hat{Q}\|_{l^2} < 1$. From the identity $PQ = \hat{Q}P$, we observe that the stabilized scheme corresponds to the operator: $Q_\sigma = P^{-1} D_\sigma \hat{Q} P$. Since P is an isomorphism, (i.e. $\|u\|_{l^2,h} = \|\hat{u}\|_{l^2}$), we have $\|Q_\sigma\|_{l^2,h} < 1$.

The explicit Euler stabilized scheme can be written now as

$$u_h^{n+1} = Q_\sigma\, u_h^n + \mathrm{d}t\, f_h{}^n + \mathrm{d}t\, r_h^n$$

where r_h^n is the discrete error added by the filtering technique at the end of each time step. This additional consistency error r_h^n is given by Theorem 1 and depends on the smoothness of the solution only.

The convergence of the scheme is then straightforward. In particular, for consistency error r_h^n brought by the filter of second or higher order and a finite difference scheme (14) that is second order, the overall scheme stays second order.

2.3. The Dirichlet boundary condition case

Let us consider first a problem with homogeneous Dirichlet boundary conditions. To apply the filter transform as defined above we construct the periodic extension v^n on the real line of the function $u^n(x)$ as follows. First, we apply the symmetry:

$$\forall x \in (0, L), \quad v^n(2L - x) = -\, v^n(x)$$

and second, the periodic extension:

$$\forall x \in (0, 2L)\ \forall k \in Z, \quad v^n(x + k2L) = v^n(x)$$

An entirely similar stability analysis to the one above can be done by using the basis of trigonometric polynomials $\{\sin(k\pi(x/L)),\ k = 1, \ldots, N\}$. One uses the matching between the

sinus expansion of u^n and the Fourier expansion of v^n. We will denote $\mathscr{P}_{\sin}$ the matrix of the sinus transform.

Let us suppose that $u^n \in C^2(0,L)$. The v^n is only C^1 in the neighbourhood of the end points $x=0$ or $x=L$. Following the result of Theorem 1, the consistency error r^n introduced by the filter is of order $1/N^2$ in the neighbourhood of $x=0$ and $x=L$, and of order $1/N^3$ away of this neighbourhood.

Our stabilized scheme writes

$$u_h^n \overset{\mathscr{Q}}{\to} u_h^{*,n+1} \overset{\mathscr{P}_{\sin}}{\to} \hat{u}^{n+1} \overset{\mathscr{D}_\sigma}{\to} \hat{u}_\sigma^{n+1} \overset{\mathscr{P}_{\sin}^{-1}}{\to} u^{n+1}$$

where $\mathscr{Q}$ is the finite difference scheme with homogeneous Dirichlet boundary conditions, and $\mathscr{P}_{\sin}$ is the matrix of the sinus transform.

The transform T removes all unstable sinus waves, i.e. it satisfies

$$\|\mathscr{D}_\sigma \hat{Q}\|_{l^2} < 1$$

Then we have, in a way similar to the periodic case, $Q_\sigma = \mathscr{P}_{\sin}^{-1} \mathscr{D}_\sigma \hat{Q} \mathscr{P}_{\sin}$ and $\|Q_\sigma\|_{l^2,h} < 1$.

Finally, let us consider the heat equation problem with nonhomogeneous Dirichlet boundary conditions:

$$\partial_t u = \partial_x^2 u + f(x,t), \quad x \in (0,L), \ t \in (0,T), \quad u(0,t) = a(t), \quad u(L,t) = b(t) \tag{15}$$

with the initial condition (13).

We can transform this nonhomogeneous Dirichlet problem into a homogeneous Dirichlet problem using a shift. We will denote by $\mathscr{S}$ the transform:

$$u \overset{\mathscr{S}}{\to} v$$

In order to guarantee that the shift does not affect the stability of the scheme we use a low-order trigonometric polynomial as follows:

$$v^n(x) = u^n(x) - (\alpha \cos(\pi x/L) + \beta) \quad \text{with } \alpha = \tfrac{1}{2}(u^n(0) - u^n(L)), \quad \beta = \tfrac{1}{2}(u^n(0) + u^n(L)) \tag{16}$$

Then, we extend v^n to a $2L$ periodic function as above. Thanks to the superposition principle, we can reuse the stability analysis for the problem with homogeneous boundary conditions.

The stabilized scheme writes now

$$u_h^n \overset{\mathscr{Q}}{\to} u_h^{*,n+1} \overset{\mathscr{S}}{\to} v^{*,n+1} \overset{\mathscr{P}_{\sin}}{\to} \hat{v}^{n+1} \overset{\mathscr{D}_\sigma}{\to} \hat{v}_\sigma^{n+1} \overset{\mathscr{P}_{\sin}^{-1}}{\to} v^{n+1} \overset{\mathscr{S}^{-1}}{\to} u^{n+1}$$

where $\mathscr{S}$ is for the shift (16).

Figure 1 gives a graphic illustration of the method. The top left figure shows the function u^{n+1} and the corresponding first-order trigonometric polynomial used in the $\mathscr{S}$ transform. The top right figure shows the shifted signal. The bottom left figure shows the periodic extension v^{n+1} of the signal. The right bottom picture shows the truncation error after filtering. It illustrates the Gibbs phenomenon located at the end point of the interval. This Gibbs phenomenon is the main limiting factor to the accuracy of the stabilized method.

Remark: The previous analysis does not apply to inhomogeneous Neumann boundary conditions, for example. As a matter of fact, the Neumann stability criteria is no longer a sufficient condition.

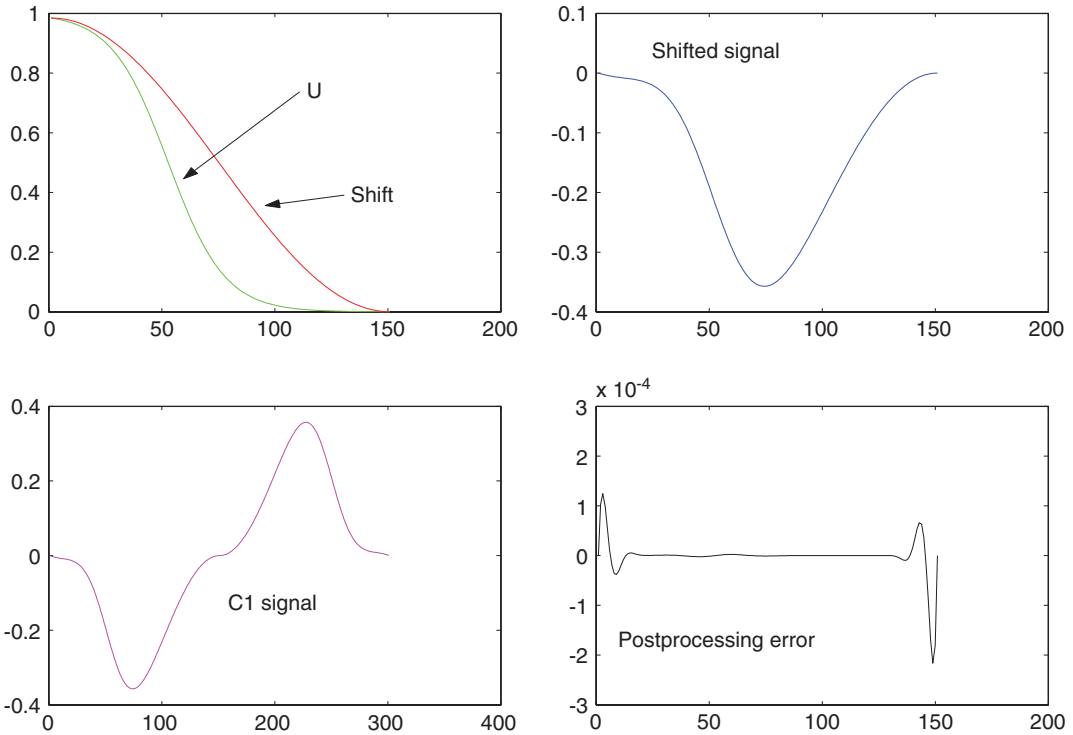

Figure 1. Illustration of the method.

2.4. *Some remarks on the implementation*

In practice we take σ to be the following eighth-order filter [20]:

$$\sigma(\xi) = y^4(35 - 84y + 70y^2 - 20y^3) \quad \text{where } y = \tfrac{1}{2}(1 + \cos(\pi\xi)) \tag{17}$$

The stretching factor $\kappa > 1$ is chosen to remove all unstable wave components. For scheme (6), for example, we obtain

$$\kappa > \kappa_c = \frac{\pi}{a \cos\left(1 - \frac{2}{3}\frac{h^2}{\mathrm{d}t}\right)} \tag{18}$$

Because the function σ is not a step function, but rather a smooth decaying function, the filter damps significantly some of the high frequencies less than N/κ, and it can be suitable to take $\kappa = C_l\kappa_c$ with C_l that is less than 1. One can compute the optimum κ for each time step by monitoring the growth of the highest waves that are not completely filtered out by $\sigma(\kappa(k/N))$.

To run the stabilized algorithm with large time step we need indeed to have κ large. A possible way to preserve the accuracy of the method is then to make the periodic extension

of the signal u^{n+1} smoother than C^1 at the end points. For this purpose, one can use a trigonometric polynomial that matches the second-order derivative of the time step solution.

To be more specific, let us assume that the solution of the parabolic problem at each time level t^n, is in $C^3(0,\pi)$. We define the shift

$$v(x) = u(x) - \sum_{j=1}^{4} \alpha_j \cos((j-1)x) \tag{19}$$

such that the extension of v to a periodic function is in $C^3(0,2L)$. The first- and third-order derivatives of v are zero at the points x_k, and the second-order derivative is approximately given by

$$u_{xx}(x_k) \approx \frac{3u^{n+1}(x_k) - 4u^n(x_k) + u^{n-1}(x_k)}{2\delta t} - f(u^{n+1}(x_k)) \tag{20}$$

The coefficients α_j are found by solving a linear system of equations:

$$\begin{aligned}
\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 &= u(0) \\
\alpha_1 - \alpha_2 + \alpha_3 - \alpha_4 &= u(\pi) \\
-\alpha_2 - 4\alpha_3 - 9\alpha_4 &= u_{xx}(0) \\
\alpha_2 - 4\alpha_3 + 9\alpha_4 &= u_{xx}(\pi)
\end{aligned}$$

This technique does not extend to the two-dimensional space problem, because we do not retrieve an approximation of the second-order normal derivative from the parabolic equation as in the one-dimensional case.

An alternative solution is to filter the discrete time derivative instead of the function u^n itself. As a matter of fact $u^{n+1} - u^n$ is of order dt and we gain automatically the order of accuracy dt by applying the filter to a smaller quantity.

For BDF, the $\mathscr{S}$ transform writes

$$v^{n+1} = u^{n+1} - \tfrac{4}{3}u^n + \tfrac{1}{3}u^{n-1} - (\alpha\cos(x) + \beta)$$

with $\alpha = \frac{1}{2}(w(0) - w(L))$, $\beta = \frac{1}{2}(w(0) + w(L))$, and $w = 3u^{n+1} - 4u^n + u^{n-1}$. We use the fact that unstable wave components have been removed from previous time steps as well. This procedure is completely general and independent of the space dimension.

We have checked on the linear heat equation test case that these different methods perform according to the linear stability analysis and have accuracy sensitive to the smoothness of the data of the problem. Our main goal, however, is to use the method to systems of nonlinear reaction–diffusion equations. We are now going to report on some numerical experience on nonlinear reaction–diffusion equations with our technique based on the postprocessing of the time derivative.

2.5. Numerical experiment with nonlinear equation

Following the interesting work of (see Reference [11] and its references), we have compared our algorithm to a number of classical time-integration schemes. We use the exact same first

two benchmark problems of Reference [11], the thermal wave problem

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + 8\,T^2(1-T)$$

and the Brusselator system

$$\frac{\partial T}{\partial t} = D_1 \frac{\partial^2 T}{\partial x^2} + \alpha - (\beta+1)T + T^2 C$$

$$\frac{\partial C}{\partial t} = D_2 \frac{\partial^2 C}{\partial x^2} + \beta T - T^2 C$$

We refer to Reference [11] for the notation and values of the parameters.

First, we have checked that, thanks to the stabilization of all unstable waves following the Fourier analysis, our scheme is unconditionally stable. The time step can still be limited by nonlinear instabilities but we have not observed negative values of the unknowns that are commonly associated with such instabilities. Second, we have compared the accuracy of our scheme using both stabilized schemes (6) and (7) to first-order Euler implicit, second-order backward Euler (BE), Crank–Nicholson, first- and second-order splitting. For the second-order splitting method we use the Strang splitting with the following order: Reaction–Diffusion–Reaction (RDR).

For the thermal wave problem, we have computed the error against the exact solution

$$u(x,t) = \tfrac{1}{2}(1 - \tanh(x - 2t))$$

The time scale of the problem is

$$\tau = h/2$$

where h is the space step. The solution is computed on the space–time domain $(-10, 10) \times (0, 1.024)$.

In Figure 2, we impose $\mathrm{d}t = h$ and look at the convergence of the solution as both parameters go to zero simultaneously. Figure 2 shows that the stabilized methods based on schemes (6) and (7) have second-order convergence. However, Crank–Nicolson and the Strang splitting have better accuracy. In Figure 3 we plot the number of floating point operations for the exact same runs. The BE scheme uses a Newton scheme and a conjugate gradient method to solve the linear system. The matrix of the linear system is preconditioned by the diagonal. Thanks to the time stepping, this matrix after rescaling is very close to the identity matrix. The BE scheme is the cheapest scheme of all traditional methods that have been tested here and requires very few iterative steps for the Krylov method. However, BE is still far more expensive than both stabilized schemes for $N \leqslant 250$. Further, the Splitting method requires one order of magnitude more flops than both stabilized schemes. To show that this is independent of the choice of the linear solver, Figure 3 shows that Strang splitting is more expansive than the stabilized methods, when one does not even count the number of flops to solve the linear system!

In Figure 4, we look at the convergence properties of the scheme with a fixed small space step, that is $h = 0.04$ and varying time step from $\mathrm{d}t$ of order h to $\mathrm{d}t$ of order one! One can verify that the stabilized scheme exhibits second-order convergence for small enough time

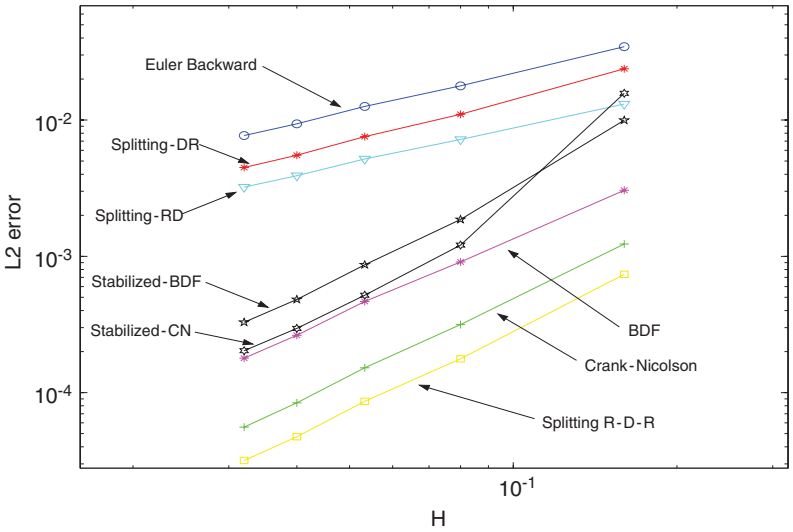

Figure 2. Convergence study for the thermal wave problem, with $dt = h$.

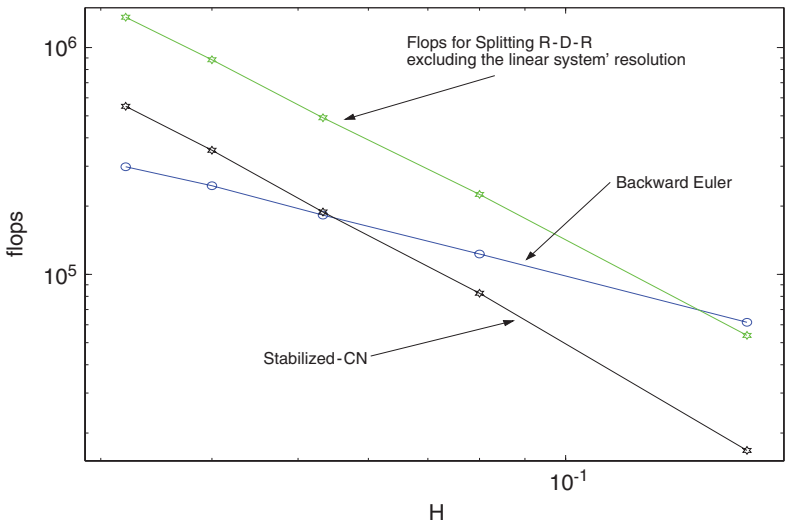

Figure 3. Arithmetic complexity for the thermal wave problem, with $dt = h$.

step and is always stable. The Crank–Nicolson scheme and Strang splitting are more accurate but more expansive also.

The test case with the Brusselator problem is most interesting. The time scale of the problem is $\tau = 12$. We are interested in computation that can handle few cycles. As pointed out in the work of Ropp *et al.* [11], the splitting methods provide peculiar solutions when the time step

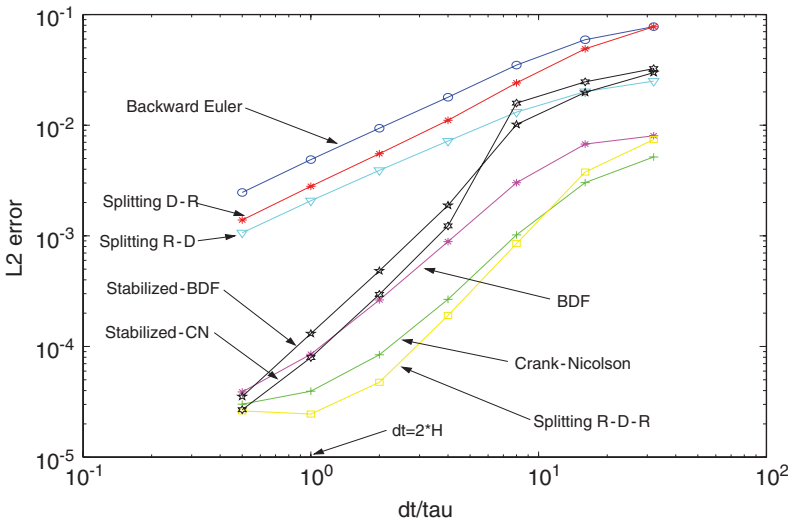

Figure 4. Convergence study for the thermal wave problem, with fixed space step $h = 0.04$.

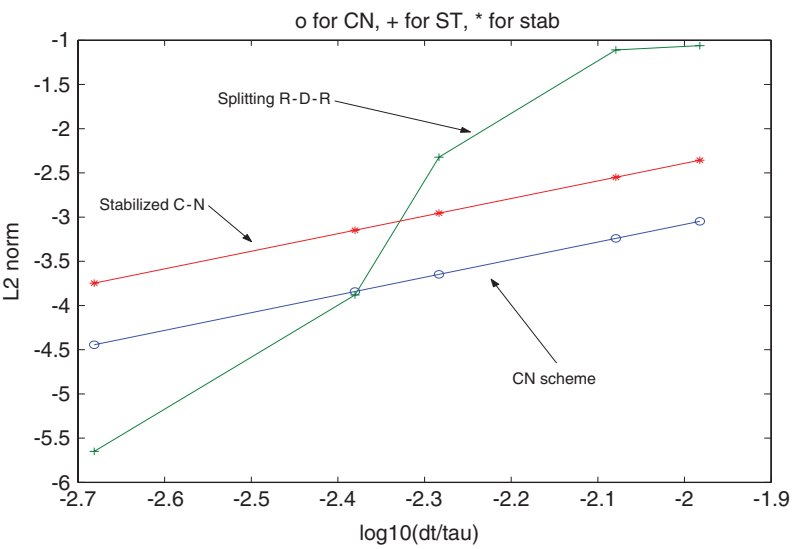

Figure 5. Convergence study for the Brusselator problem, with fixed space step $h = 0.0025$.

is not small enough. We have reproduced this result in Figure 5, for $T = 80$, fixed space step $h = 0.0025$ and varying time steps. Since we are looking at the convergence with respect to the time step, the referenced solution is obtained by using Strang splitting with a time step twice smaller as the smallest time step shown in Figure 4, second-order Richardson extrapolation in time and the same fixed space step h. The error for the Crank–Nicolson and second-order

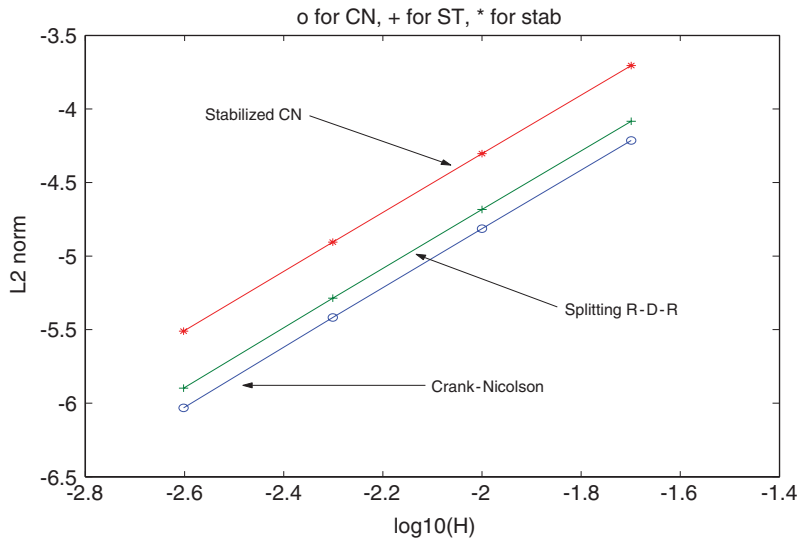

Figure 6. Convergence study for the Brusselator problem, with $\mathrm{d}t = h$.

splitting of Strang methods exhibit a small plateau as the time step $\mathrm{d}t$ get very small that correspond to the spatial error.

Figure 6 shows the convergence of the numerical scheme with $\mathrm{d}t = h$ and both parameters going to zero. The reference solution is obtained with $\mathrm{d}t = 0.002$ and $h = 5 \times 10^{-4}$, and the Strang splitting. Crank–Nicolson and the splitting methods gives similar accuracy. The stabilized Crank–Nicolson scheme (7) provides a slightly less accurate solution with much less flops.

We will apply this technique to a stiff nonlinear problem corresponding to a simplified air pollution model in Sections 3 and 4, but first let us extend the algorithm to multiple-dimension problems.

2.6. *Generalization to two-space dimensions*

For simplicity, we restrict ourselves in this presentation to two space dimensions, but the present method can be extended to three dimensions in a straightforward way (see Section 5.2). Let us consider the problem

$$\partial_t u = \Delta u + f(u), \quad (x, y) \in (0, \pi)^2, \ \ t > 0 \tag{21}$$

in two space dimensions with Dirichlet boundary conditions

$$u(x, 0/\pi) = g_{0/\pi}(y), \quad u(0/\pi, y) = h_{0/\pi}(x), \ \ x, y \in (0, \pi)$$

subject to compatibility conditions:

$$g_{0/\pi}(0) = h_0(0/\pi), \quad g_{0/\pi}(\pi) = h_\pi(0/\pi)$$

Once again, we look at a scheme analogous to (5) with, for example, a five-point scheme for the approximation of the diffusive term. Our extension of the algorithm to a multidimensional

problem is rather straightforward. The algorithm remains essentially the same, except for the fact that one needs to construct an appropriate low frequency shift that allows the application of a filter to a smooth periodic function in *both* space directions. One first employs a shift to obtain homogeneous boundary conditions in x direction

$$v(x, y) = u(x, y) - (\alpha \cos(x) + \beta) \tag{22}$$

with

$$\alpha(y) = \tfrac{1}{2}(g_0 - g_\pi), \quad \beta(y) = \tfrac{1}{2}(g_0 + g_\pi)$$

and then an additional shift in y direction as follows:

$$w(x, y) = v(x, y) - (\gamma \cos(y) + \delta) \tag{23}$$

with

$$\gamma(x) = \tfrac{1}{2}(v(x, 0) - v(x, \pi)), \quad \delta(x) = \tfrac{1}{2}(v(x, 0) + v(x, \pi))$$

In order to guarantee that none of the possibly unstable high frequencies will appear in the reconstruction step:

$$u(x) = \sigma_N w(x) + \alpha \cos(x) + \beta + \gamma \cos(y) + \delta \tag{24}$$

high-frequency components of the boundary conditions g must be filtered out with the procedure described in Section 2.3.

We will present in the next section a domain decomposition motivated by mesh refinement around a point source. Let us mention that for the convective term, we have used either explicit second-order one-side finite differences for the convective terms or the method of characteristic that is *a priori* implicit and can be made second-order (see Reference [8] and its references).

3. ADAPTIVE DOMAIN DECOMPOSITION TO TREAT A LOCALIZED SOURCE IN AIR POLLUTION

We are now going to apply our filtering technique to a simplified air pollution model. We will restrict our attention to situations where diffusion terms usually require an implicit solver in space. The first example will be brought by local refinement configuration when one wants to capture accurately a localized source term such as a chimney. As a simple illustration, we consider the following reactions, which constitute a basic air pollution model taken from Reference [4]:

$$\begin{aligned} NO_2 + h\nu &\longrightarrow^{k_1} NO + O(^3P) \\ O(^3P) + O_2 &\longrightarrow^{k_2} O^3 \\ NO + O_3 &\longrightarrow^{k_3} O_2 + NO_2 \end{aligned}$$

We set $c_1 = [O(^3P)]$, $c_2 = [NO]$, $c_3 = [NO_2]$, $c_4 = [O_3]$. One begins by considering the following corresponding ODE system:

$$\frac{\mathrm{d}c_1}{\mathrm{d}t} = k_1 c_3 - k_2 c_1$$

$$\frac{\mathrm{d}c_2}{\mathrm{d}t} = k_1 c_3 - k_3 c_2 c_4 + s_2$$

$$\frac{\mathrm{d}c_3}{\mathrm{d}t} = k_3 c_2 c_4 - k_1 c_3$$

$$\frac{\mathrm{d}c_4}{\mathrm{d}t} = k_2 c_1 - k_3 c_2 c_4$$

We write the system as

$$\frac{\mathrm{d}c}{\mathrm{d}t} = F(c,t)$$

with initial condition

$$c(t=0) = [0, 13 \times 10^7, 5 \times 10^{11}, 8 \times 10^{11}]^{\mathrm{t}} \tag{25}$$

The chemical parameter values are

$$s_2 = 10^6, \quad k_3 = 10^{-16}, \quad k_2 = 10^5$$

k_1 is time dependent. At night $k_1 = 10^{-40}$. During the day

$$k_1 = 10^{-5} \exp\left(7\left(\sin\left(\frac{\pi}{16}(t_h - 4)\right)\right)^{0.2}\right) \tag{26}$$

for

$$t_h \in (4, 20)$$

with

$$t_h = t/3600 - 24[t/3600/24]$$

It can be shown that this problem is well posed, and that the vector function $c(t)$ is continuous [21]. At transition between day and night the discontinuity of $k_1(t)$ brings a discontinuity of the time derivative $\mathrm{d}c/\mathrm{d}t$. This singularity is typical of air pollution problems. Nevertheless, this test case can be computed with second BE (BDF) and constant time step for about four days, more precisely $t \in (0, 3 \times 10^5)$ with $\mathrm{d}t < 1200$. Larger time steps lead to too inaccurate solution. We use a Newton scheme to solve the nonlinear set of equations provided by BDF at each time step. The norm of the Jacobian of the system is of order 10^8.

We recall that for air pollution, we look for numerically efficient schemes that deliver a solution with a 1% error.

Second, we introduce in the previous model spatial diffusion effect.

$$\frac{\partial C}{\partial t} = \Delta C + F(C, x, y, t), \quad (x, y) \in (-L, L)^2 \tag{27}$$

with homogeneous Newmann boundary conditions.

We take a source term $S(x, y)$ that exhibits a sharp peak at the centre $(x, y) = (0, 0)$ of the domain:

$$S(x, y) = s_2 \exp(-40(x^2 + y^2)/L^2)$$

with $L = 400$. Let G be a regular grid of constant space h_1 in every direction of Ω. In order to capture accurately the source term, we decompose Ω into two overlapping subdomains Ω_1 and $\Omega_2 = (-L/q, L/q)^2$, of approximately the same number of grid points. In particular, the space grid G_2 of Ω_2 is embedded in G with a refinement factor $q = 2^k$, $k \in \Omega \subset \mathbf{N}$. G_1 denotes the grid G restricted to Ω_1. The thickness of the overlap between Ω_1 and Ω_2 is of order h_1. We use the same semi-implicit BDF scheme

$$\frac{3C_k^{n+1} - 4C_k^n + C_k^{n-1}}{2\,\mathrm{d}t} = 2\Delta^h C_k^n - \Delta^h C_k^{n-1} + F(C_k^{n+1}), \quad k = 1, 2 \tag{28}$$

Δ^h is for the classical five-point approximation of the Laplacian, i.e.

$$\forall k \in \{1, 2\} : \Delta^h C_k(x, y) = \frac{C_k(x + h, y) + C_k(x - h, y) + C_k(x, y + h) + C_k(x, y - h) - 4C_k(x, y)}{h^2}$$

Let us denote $\mathrm{d}t_0 = h_1^2/6$, the maximum time step allowed with the explicit treatment of the diffusion term in Ω_1.

In principle the explicit time stepping with respect to diffusion should lower the time step to $\mathrm{d}t = \mathrm{d}t_0/q^2$! Our filtering technique allows us to use the *same time step* in Ω_1 and Ω_2. We apply the method described in Section 2.6 to the solution in Ω_2. The implementation is therefore extremely simple. We let C_1^{n+1}, (respectively, C_2^{n+1}) be the solution in Ω_1, (respectively, Ω_2) obtained at time step t^{n+1}.

Then we assemble the global solution on the composite grids $G_1 \cup G_2$ using a cut-off function $\mathbf{H}$, so-called partition of unity. $\mathbf{H}(x, y)$ has to be a smooth bounded function with values between 0 and 1, such that

$$\mathbf{H}(x, y) = 1 \quad \text{for } (x, y) \in \Omega \backslash \Omega_2$$

$$\mathbf{H}(x, y) = 0 \quad \text{for } (x, y) \in \Omega_2 \backslash \Omega_1$$

At each time step t^n, we assemble the solution on the composite grids to be

$$C_1^{n+1} := \mathbf{H} \cdot C_1^{n+1} + (1 - \mathbf{H}) P(C_2^{n+1})$$

$$C_2^{n+1} := \mathbf{H} \cdot I(C_1^{n+1}) + (1 - \mathbf{H}) \cdot C_2^{n+1}$$

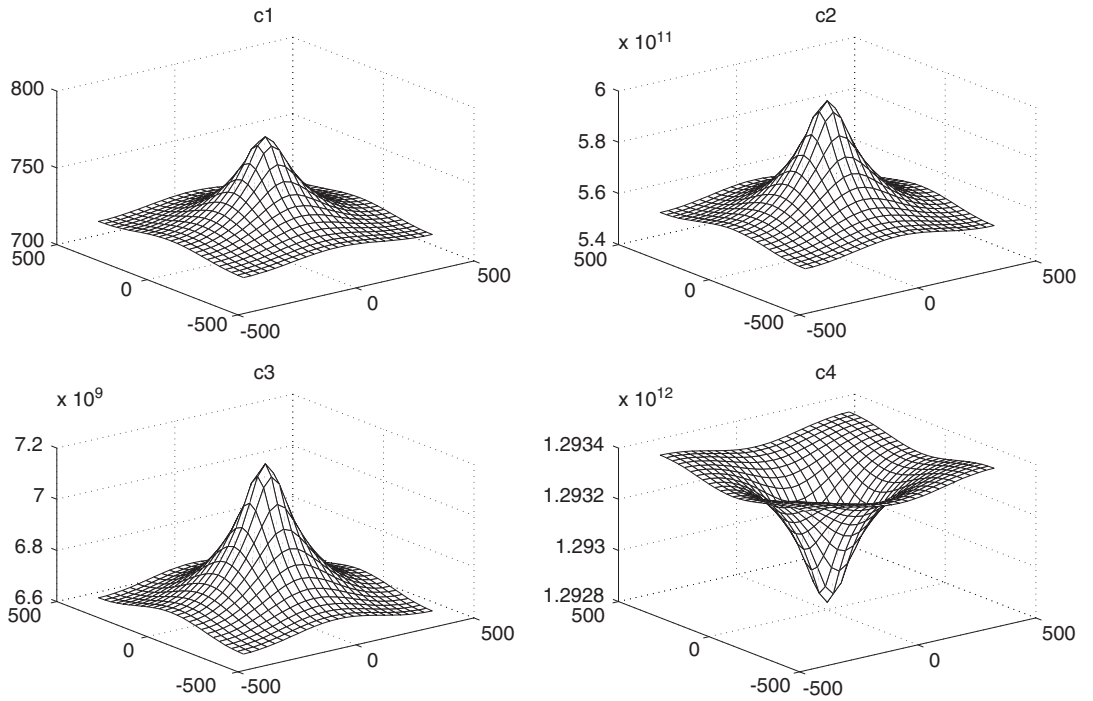

Figure 7. Solution $C=(C_1, C_2, C_3, C_4)$, at final time $t=3\times 10^5$ on grid G_1.

where P denotes the projection of G_2 into G_1, and I denotes the spline interpolation from G_1 to G_2.

We have implemented this domain decomposition on problem (27) for $t\in(0, 10^5)$ with constant state initial condition (25). The time step is the maximum time step allowed by the explicit treatment of the diffusion term in Ω_1. The refinement factor in the internal subdomain Ω_2 is 4, and therefore the time step is 16 *times larger* than the explicit treatment of the diffusion term in Ω_2 should allow. The cut-off function $\mathbf{H}$ reemploys the eighth-order filter profile (17) with a large κ. The number of grid points in G and G_2 is 25 in each directions. Figures 7–9 show the result of this computation with FUcos1 method. We have checked that the main error comes from the integration of the ODE system at each grid point. We have an accurate resolution of the peak at the centre points thanks to the local refinement. The composite solution remains smooth and the time stepping is stable even if the thickness of the overlap is as low as $2h_1$.

Finally, we observe that our algorithmic solution does not require any operator splitting and is obviously rather inexpensive in terms of arithmetic complexity.

4. APPLICATION TO VERTICAL TRANSPORT IN AIR POLLUTION

Our second example will be related to vertical transport of pollutant induced by gravity [1, 3]. In this case the diffusion coefficient is space dependent, and the source term is located near

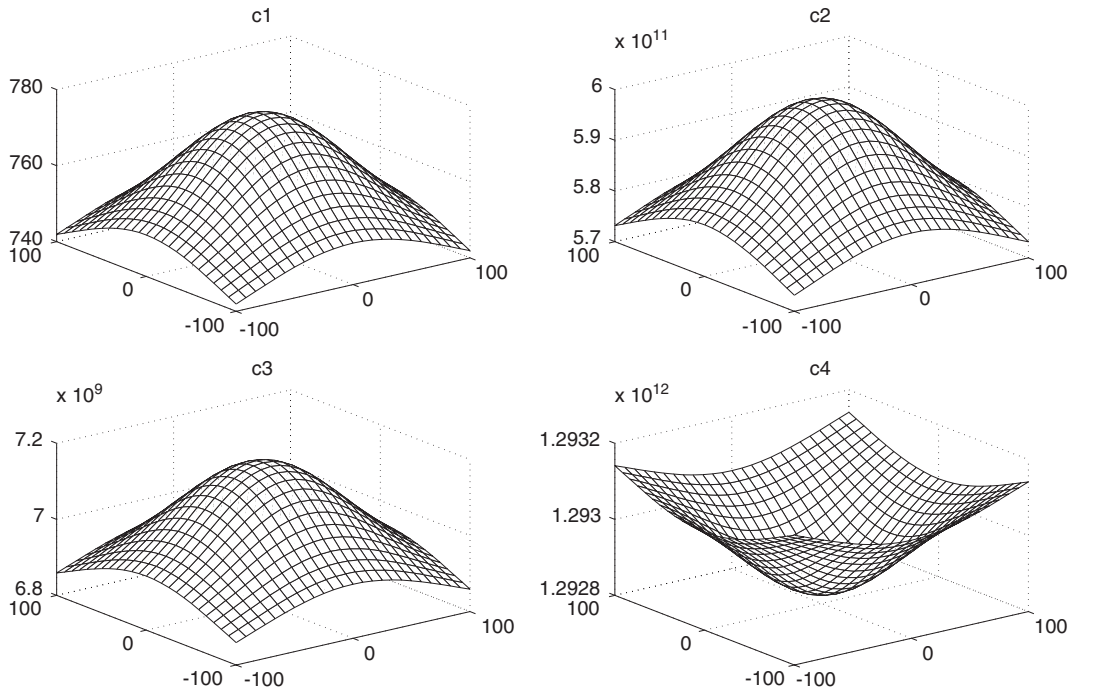

Figure 8. Solution $C=(C_1,C_2,C_3,C_4)$, at final time $t=3\times 10^5$ on the fine grid G_2.

the surface, so one must use adaptive mesh with fine space step near the ground. In such a situation, one usually uses an implicit solver in space to treat the diffusion term [1]. In this section we use our filtering methodology, which can work properly on reaction–diffusion systems with stretched space grid, space-dependent diffusion coefficient, and Robin boundary conditions.

Let us consider the simplified ozone model of Reference [4] adding some vertical diffusion effect:

$$\frac{\partial C}{\partial t}=\frac{\partial}{\partial z}\left(\mathbf{K}\frac{\partial C}{\partial z}\right)+F(C,t),\quad z\in(0,L) \tag{29}$$

with boundary conditions,

$$-\mathbf{K}\frac{\partial C}{\partial z_{|z=0}}=E-v\cdot C,\quad \frac{\partial C}{\partial z_{|z=L}}=0 \tag{30}$$

We take $\mathbf{K}(z)$ to be a function having the same structure as in Reference [1]. Figure 10 gives a graphic representation of $\mathbf{K}(z)$. More precisely, below the height of the base of a given lowest strong inversion layer denoted L_i, we have

$$\mathbf{K}(z)=\frac{d_1(z+d_0)}{0.74+4.7\dfrac{z}{L_i}},\quad z\in(0,L_i)$$

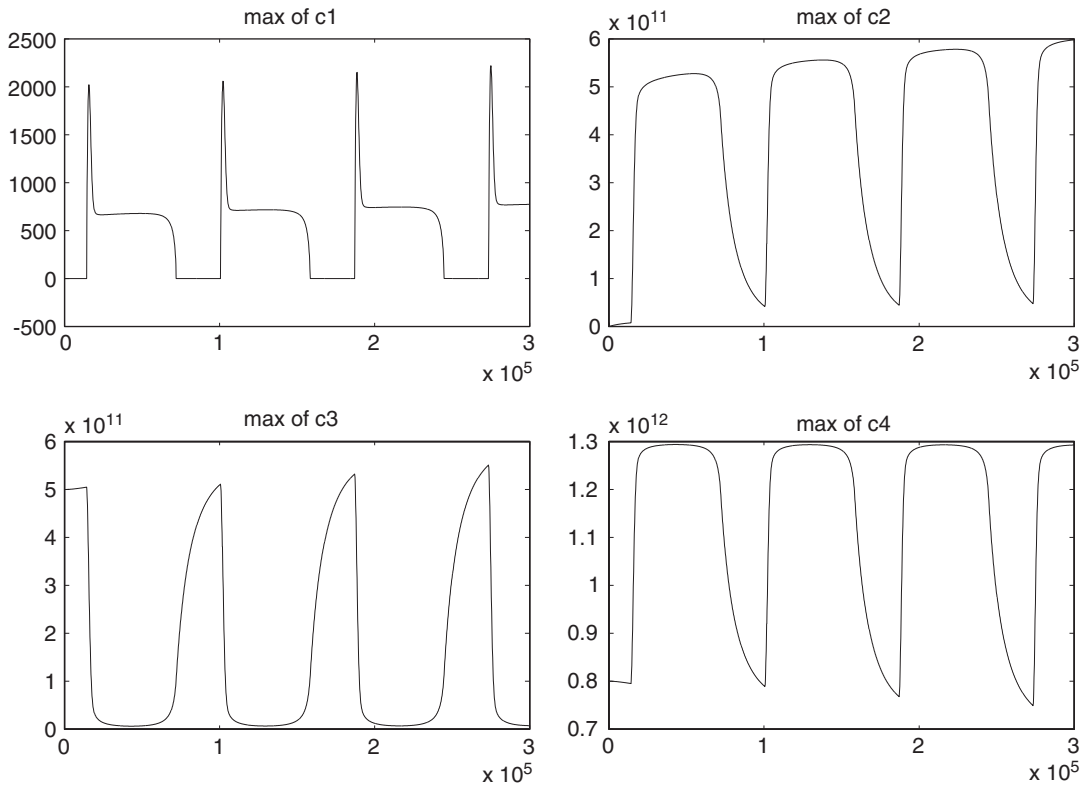

Figure 9. Evolution of L_∞ norm of (C_j), $j = 1, \ldots, 4$, for $t \in (0, 3 \times 10^5)$.

Above L_i, we take $\mathbf{K}(z)$, $z \in (L_i, L)$ to be a sharp decreasing hyperbolic tangent profile $a \tanh(-[(z - L_u)/\varepsilon]) + b$ decreasing from $\|K\|_\infty$ to $K_0 = 0.05\|K\|_\infty$. We require that $\mathbf{K}(z)$ be a continuous function with a derivative having a jump at $z = L_i$. E models the emission of C at surface level, v is for the dry deposition.

The source term in C_2 equation is written

$$S(z) = s_2 \exp(-C_s z^2/L^2)$$

In the numerical experiment reported in this paper, we take: $d_0 = 10$, $d1 = 0.03$, $L_i = 400$, $L_u = 1000$, $\varepsilon = 100$, $C_s = 5$, $L = 3870$, $E = (0, 0.1, 0, 0)^t$, $v = (0, 0, 0, 0.01)$. All physical parameters in this experiment are somehow artificial. Nevertheless, we believe that this test case is representative of the difficulties of the numerics. The mapping used to define the space grid is smooth and based on an hyperbolic tangent function. The solution, cf. Figures 10 and 11, was obtained with time step $dt = 651$. We have checked that this time step is 8 times larger than the maximum time step allowed with no filtering. Further, the difference between this numerical solution (see Figure 10), and the explicit solution with no filtering at $t = 10^5$ is of the order 1%.

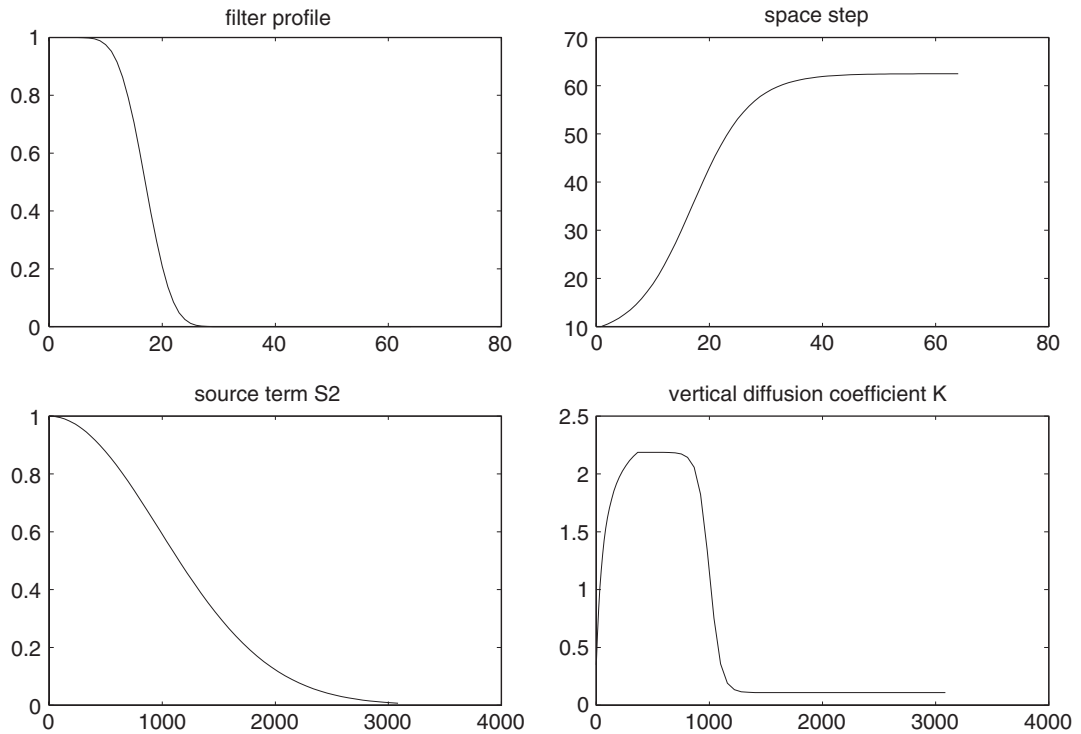

Figure 10. Filter profile σ, space step function $h(z)$, source term function $S(z)$, and $\mathbf{K}(z)$ function.

We now are going to describe some critical elements of the parallel implementation of our method for multidimensional air pollution problems.

5. ON THE STRUCTURE AND PERFORMANCE OF THE PARALLEL ALGORITHM

For simplicity, we will first restrict our system of reaction–diffusion equations to two space dimensions. A performance analysis for the general case with three space dimensions will then follow.

5.1. Two-dimensional case

The code must process a three-dimensional array $U(1:N_c, 1:Nx, 1:Ny)$ where the first index corresponds to the chemical species, and the second and third correspond to space dependency. The method that we have presented in Section 2 can be decomposed into two steps:

Step 1: Evaluation of a formula

$$U(:,i,j) := G(U(:,i,j), U(:,i+1,j), U(:,i-1,j), U(:,i,j+1), U(:,i,j-1)) \tag{31}$$

at each grid point provided appropriate boundary conditions.

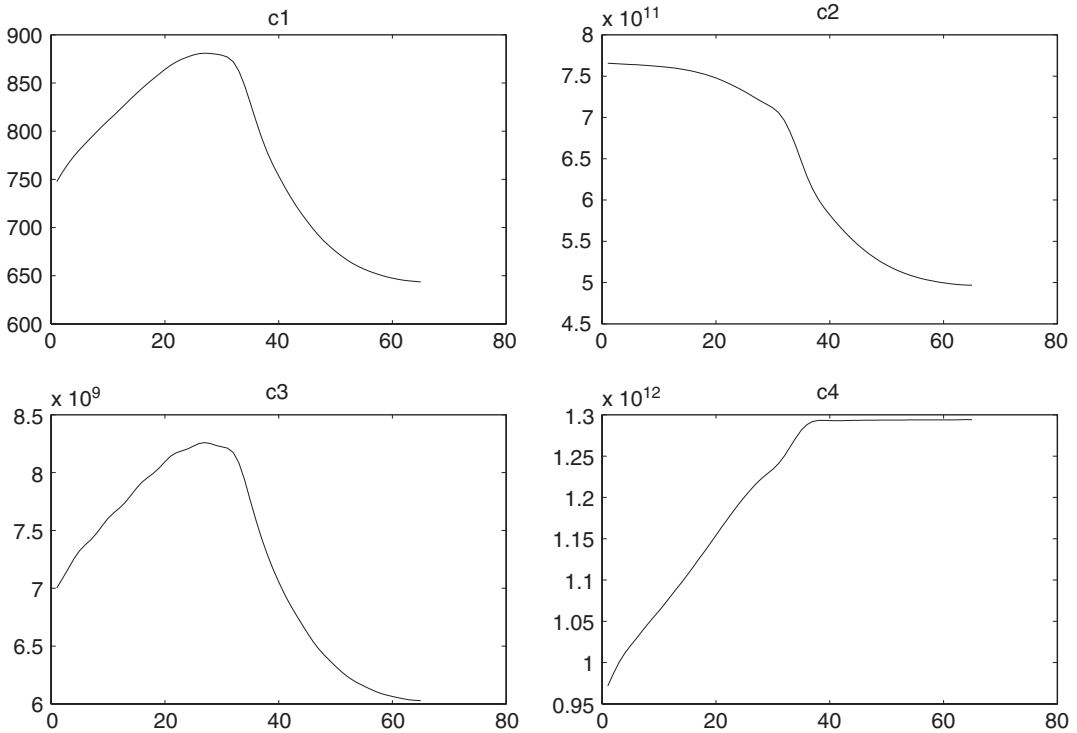

Figure 11. Solution $C = (C_1, C_2, C3, C_4)$, at final time $t = 10^5$.

Step 2: Shifted filtering of $U(:, i, j)$ with respect to i and j directions.

Step 1 corresponds to the semi-explicit time marching (28) and is basically parametrized by space variables. In addition, the data dependencies with respect to i and j correspond to the classical five-point graph common in second-order central finite differences. The parallel implementation of Step 1 is straightforward: one decomposes the array $U(:, 1:Nx, 1:Ny)$ into subblocks distributed on a two-dimensional Cartesian grid of $px \times py$ processors with an overlap of at most one row and column in each direction. Parallel performance of this algorithm is well known. If the load per processor is high enough (which is likely the case with the pointwise integration of the chemistry) this algorithm scales very well (see for example Reference [22]).

For the reaction–convection–diffusion solvers used in air pollution [4], it is most common to use one-side second-order finite differences formula for the convection terms in order to deal with sharp gradient of the velocity field. Rather than the five-point formula (31), one computes

$$U(:, i, j) := G(U(:, i, j), U(:, i+1, j), U(:, i-1, j), U(:, i, j+1), U(:, i, j-1), \\ U(: i \pm 2, j) U(:, i, j \pm 2)) \tag{32}$$

and the stencil depends locally on the orientation of the transport vector field **a**. The data distribution of $U(:, i, j)$ requires then an overlap of at most *two* rows and columns in each direction.

The parallel efficiency of the implementation can still be high even for the BDF scheme applied to the linearized problem, provided that the network of the parallel computer is good enough. We refer to Tables I–III that report on performance with a Cray T3E with varying number of equations N_c. To obtain a lower estimate of the parallel efficiency of step one, we have skipped the integration of the chemistry in the runs. Nevertheless, we observe that the efficiency of the runs with 64 processors, growths with the number of species N_c. From these tables, it can be seen that at fixed number of processors $px \times py$, it is best for $N_c = 1$ and 4, to maximize py and take $px = 1$. This dissymetry in performance with respect to px and py, is due to the fact that arrays in Fortran are stored by column.

The data structure is imposed by Step 1 and we will proceed with the analysis of the parallel implementation of Step 2.

Step 2 introduces global data dependencies across i and j. It is therefore more difficult to parallelize the filtering algorithm. The kernel of this algorithm is to construct the

Table I. Efficiency on a Cray T3E with $N_c = 1$, $Nx = Ny = 128$.

$px \times py$ proc.	$py = 1$	$py = 2$	$py = 4$	$py = 8$	$py = 16$
$px = 1$	100.00	97.5	90.8	93.0	93.8
$px = 2$	91.7	87.2	88.8	94.5	85.0
$px = 4$	80.8	84.9	89.6	80.5	56.1
$px = 8$	77.2	86.3	77.7	57.4	
$px = 16$	75.8	73.3	54.4		

Table II. Efficiency on a Cray T3E with $N_c = 4$, $Nx = Ny = 128$.

$px \times py$ proc.	$py = 1$	$py = 2$	$py = 4$	$py = 8$	$py = 16$
$px = 1$	100.00	101.3	97.9	94.3	79.5
$px = 2$	102.5	97.2	89.6	83.4	80.2
$px = 4$	92.6	87.8	82.6	80.0	77.8
$px = 8$	80.6	75.2	76.2	79.7	
$px = 16$	62.9	63.9	70.2		

Table III. Efficiency on a Cray T3E with $N_c = 20$, $Nx = Ny = 128$.

$px \times py$ proc.	$py = 1$	$py = 2$	$py = 4$	$py = 8$	$py = 16$
$px = 1$	100.00	95.4	92.7	88.4	78.9
$px = 2$	110.0	102.6	96.9	89.9	84.6
$px = 4$	107.3	96.2	94.1	94.0	78.3
$px = 8$	94.7	89.6	92.0	82.4	
$px = 16$	75.6	76.5	73.3		

two-dimensional sine expansion of $U(:,i,j)$ modulo a shift, and its inverse. One may use an off the shelf parallel FFT library that supports two-dimension distribution of matrices (e.g. http://www.fftw.org). In principle the arithmetic complexity of this algorithm is of order $N_c N^2 \log(N)$ if $Nx \sim N$, $Ny \sim N$. It is well known that the inefficiency of the parallel implementation of the FFTs comes from the global transpose of $U(:,i,j)$ across the two-dimensional network of processors. Although for air pollution problems on medium-scale parallel computers, we do not expect to have Nx and Ny much larger than 100 because of the intense pointwise computation induced by the chemistry. An alternative approach to FFTs that can use fully the vector data structure of $U(:,i,j,)$ is to write Step 2 in matrix multiply form:

$$\forall k = 1 :: N_c, \quad U(k,:,:) := A_{x,\sin}^{-1} \times (F_x \cdot A_{x,\sin}) U(k,:,:) (A_{y,\sin}^{t} \cdot F_y) \times A_{y,\sin}^{-t} \tag{33}$$

where $A_{x,\sin}$ (respectively, $A_{y,\sin}$) is the matrix corresponding to the sine expansion transform in x direction and F_x (respectively, F_y) is the matrix corresponding to the filtering process. In (33), $\cdot$ denotes the multiplication of matrices component by component. Let us define $A_{\text{left}} = A_{x,\sin}^{-1} \times (F_x \cdot A_{x,\sin})$ and $A_{\text{right}} = (A_{y,\sin}^{t} \cdot F_y) \times A_{y,\sin}^{-t}$. These two matrices A_{left} and A_{right} can be computed once for all and stored in the local memory of each processor. Since $U(:,i,j)$ is distributed on a two-dimensional network of processors, one can use an approach very similar to the systolic algorithm [23] to realize in parallel the matrix multiply $A_{\text{left}} U(k,:,:) A_{\text{right}}$ for all $k = 1, \ldots, N_c$. Let $p_x \times p_y$ be the size of the two-dimensional grid of processors. The first one does $p_y - 1$ shifts of every subblock of $U(k,:,:)$ in y direction assuming periodicity in order to construct $\forall k$, $V(k,:,:) = A_{\text{left}} U(k,:,:)$. The second one does $p_x - 1$ shifts of every subblocks of $V(k,:,:)$ in x direction assuming periodicity in order to construct $\forall k, U(k,:,:) = V(k,:,:) A_{\text{right}}$. If one assumes a linear communication cost model with latency τ and cost per word t_w, then the communication cost is of order:

$$(p_y - 1)\left(\tau + t_{\text{w}} \frac{N^2}{p_x p_y}\right) + (p_x - 1)\left(\tau + t_{\text{w}} \frac{N^2}{p_x p_y}\right)$$

One can easily use nonblocking communication in the implementation, in order to overlap the communication by the computation. Basically the time necessary to move the data in x and then y direction is negligible in comparison with the time spent in the matrix multiply.

Further, we observe that the matrices can be approximated by sparses matrices, if one neglects the matrix coefficients less than some small tolerance number tol. This comes from the fact that in the limit case, $\kappa = 0$, A_{left} and A_{right} are the identity. The number of 'nonneglectable' coefficients then grows with κ. We did check that the time accuracy is then no larger than $O(\text{tol})$ for small tolerance number. Figure 12 gives the elapsed time on an EV6 processor at 500 MHz obtained for the filtering procedure for various problem sizes, $\kappa = 2$, and using or not the fact that the matrices A_{left} and A_{right} can be approximated by sparse matrices. However, the use of optimum basic linear algebra subroutines customized to the processor can make the use of the approximated sparse structure unnecessary and possibly inefficient. This method should be competitive to a filtering process using FFT for large N_c and not so large N_x and N_y. But the parallel efficiency of the algorithm as opposed to FFT on such small data sets is very high (see Tables IV–VI).

For $N_c = 1, 4$, we benefit once again from the cache memory effect, and obtain perfect speedup with up to 32 processors. For larger numbers of species, $N_c = 20$ for example, we observe a deterioration of performance, and we should introduce a second level of parallelism

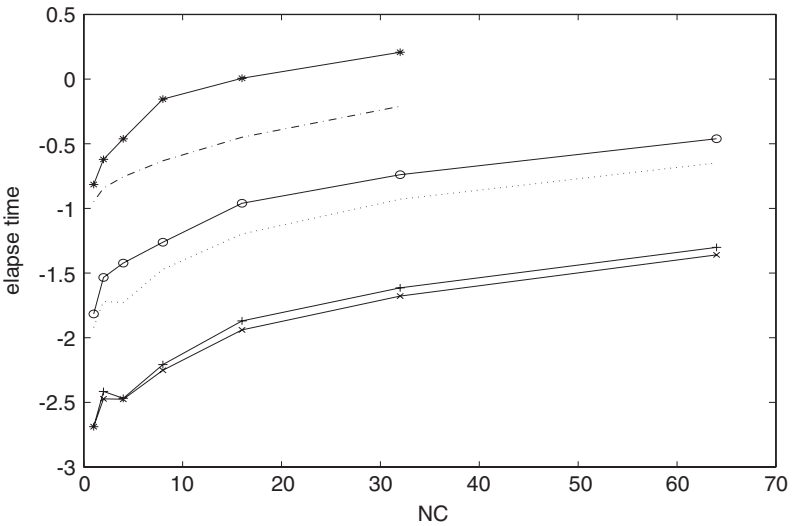

Figure 12. Elapse time of the matrix multiply form of the filtering process as a function of N_c. '*' for $N_x = N_y = 128$, 'o' for $N_x = N_y = 64$, '+' for $N_x = N_y = 32$ and $\mathrm{tol} = 0$, '−.' for $N_x = N_y = 128$, '.' for $N_x = N_y = 64$, 'x' for $N_x = N_y = 32$ and $\mathrm{tol} = 1^{-5}$.

Table IV. Efficiency on a Cray T3E with $N_c = 1$, $Nx = Ny = 128$.

$px \times py$ proc.	$py = 1$	$py = 2$	$py = 4$	$py = 8$	$py = 16$
$px = 1$	100.00	98.4	93.0	86.2	70.3
$px = 2$	268.1	256.7	228.9	187.8	112.5
$px = 4$	244.5	230.1	191.8	128.8	54.9
$px = 8$	215.1	185.4	127.6	60.3	
$px = 16$	180.6	118.8	60.0		

Table V. Efficiency on a Cray T3E with $N_c = 4$, $Nx = Ny = 128$.

$px \times py$ proc.	$py = 1$	$py = 2$	$py = 4$	$py = 8$	$py = 16$
$px = 1$	100.00	98.0	90.9	84.2	70.0
$px = 2$	171.3	166.2	149.0	127.8	93.2
$px = 4$	158.1	151.9	130.1	100.0	60.4
$px = 8$	140.8	128.7	102.7	61.8	
$px = 16$	114.6	96.0	61.3		

with domain decomposition to lower the dimension of each subproblem and get a data set that fits in the cache.

Let us now discuss the results for a three-dimensional problem in space.

Table VI. Efficiency on a Cray T3E with $N_c = 20$, $Nx = Ny = 128$.

$px \times py$ proc.	$py = 1$	$py = 2$	$py = 4$	$py = 8$	$py = 16$
$px = 1$	100.00	97.3	88.1	79.9	66.2
$px = 2$	120.2	117.0	103.4	91.7	73.1
$px = 4$	110.9	106.6	94.8	81.7	60.5
$px = 8$	99.9	96.5	83.0	66.1	
$px = 16$	83.7	78.0	61.4		

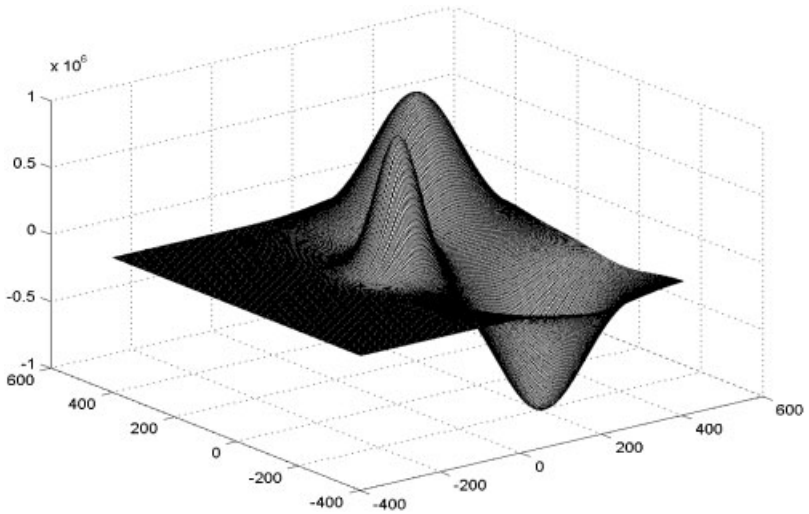

Figure 13. Horizontal slice of the source term.

5.2. *Three-dimensional case*

The problem writes,

$$\frac{\partial C}{\partial t} = \Delta C + F(C, x, y, z, t), \quad (x, y) \in (-L, L)^2, \ \ (z) \in (0, h) \tag{34}$$

with

$$\Delta C = \partial_x^2 C + \partial_y^2 C + \frac{\partial}{\partial z}\left(\mathbf{K}\frac{\partial C}{\partial z}\right) \tag{35}$$

where **K** has been defined in Section 4. To have strong dependency on space of the solution, we take a source term $S(x, y, z)$ that exhibits two sharp peaks and one sink in the horizontal (x, y) direction and vanishes exponentially in the vertical direction (see Figure 13). The mesh is refined in the neighbourhood of the ground, (i.e. $z = 0$), as in Section 4. The parallel efficiency of the filter with a Cray T3E is given in Table VII, and we observe slightly better performances for the heat equation code.

Table VII. Parallel Efficiency of the filter based on matrix multiply with $N_c = 4$, $Nx = Ny = 64$, $Nz = 32$.

$px \times py$ proc.	$py = 1$	$py = 2$	$py = 4$
$px = 1$	100.00	70.6	50.4
$px = 2$	71.3	62.8	72.9
$px = 4$	50.2	71.7	65.1

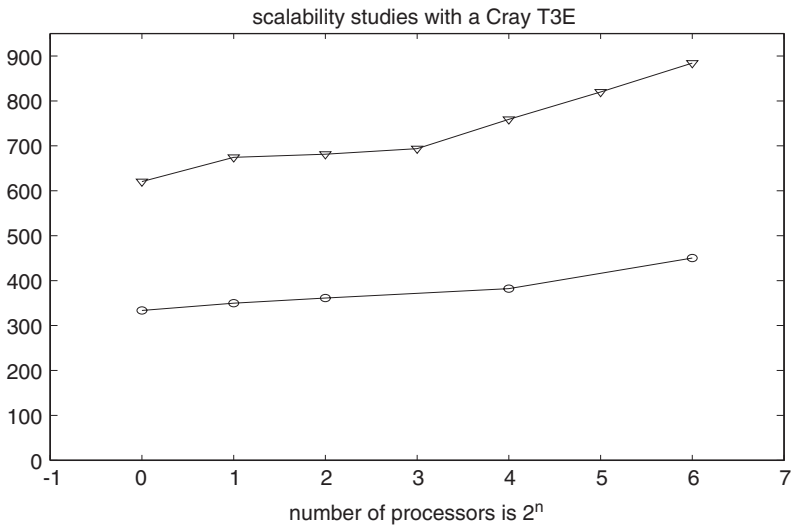

Figure 14. Scalability performance for the Ozone case on a Cray T3E with 450 MHz alpha processors. The latency of the network is 12 μs and the bandwidth 320 MB/s. Each processor has $16 \times 16 \times N_z$ grid points with $N_z = 16$ for the 'o' curve, respectively, $N_z = 32$ for the 'v' curve.

The reason this parallel efficiency is lower than in the two-dimensional case, is that the ratio of flops per communication is less advantageous. The ratio of unknowns in the vertical pencil of data owned by each processor per number of unknowns on the boundary is $N^{3/2}$ when it was N^2 in the two-dimensional case. Nevertheless, we do have excellent parallel scalability of our parallel algorithm considering the fact that we can keep dt of order the space step with our filtering method. Figures 14 and 15 give the elapsed time on, respectively, a Cray T3E and a Compacq cluster of four EV6 four processors alpha servers connected by a quadrix switch. The number of grid points on each processor is $N_x \times N_y \times N_z$. It is fixed in the two-dimensional partitioning no matter the number of processors. In these runs, $N_x = N_y = 16$ and $N_c = 4$. The global size of the problem processed in parallel is $p_x N_x \times p_y N_y \times Nz \times N_c$, with the two-dimensional grid of processors $p_x \times p_y$. The only part of the algorithm that has nonlinear arithmetic complexity is the filter, (i.e. the matrix–matrix vector product (33)). With $N_z = 16$, the elapsed time spent in the filtering procedure grows from 10 to 28% of the total elapsed time while the number of processors grows from one to sixteen on the Compacq cluster. This growth rate is much less than linear, as it can be seen also on Figures 14 and 15.

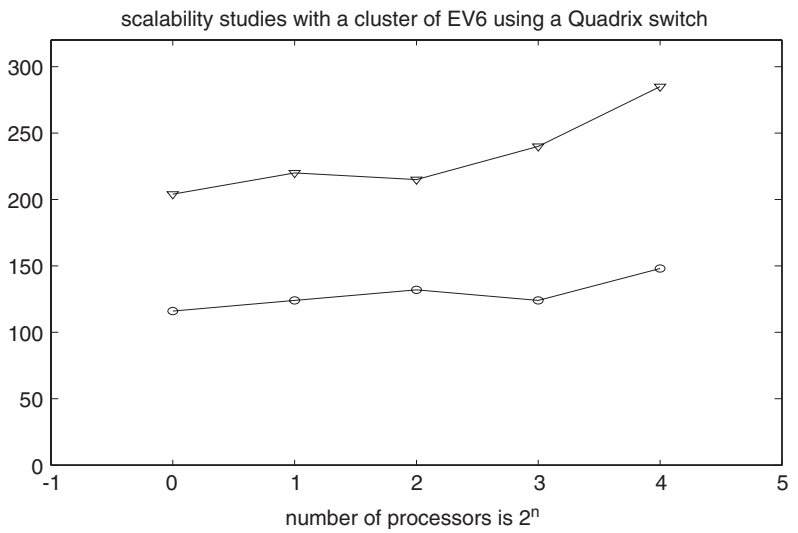

Figure 15. Scalability performance for the Ozone case on a Compacq cluster of 4 EV67/677 alpha processors linked by a Quadrix switch. Each processor has $16 \times 16 \times N_z$ grid points with $N_z = 16$ for the 'o' curve, respectively, $N_z = 32$ for the 'v' curve.

Considering the fact that we have a stabilized semi-implicit scheme that can run at time step comparable to a fully implicit scheme, this result is better than just scalability.

However, this conclusion on good scalability is valid only when the concentration of the chemical species in space is smooth enough to allow the shifted Fourier expansion used in filtering to converge faster than second order. This is obviously very much application dependent and not necessarily trivial for air quality problems.

6. CONCLUSION

In this paper, we have introduced a new family of fast and numerically efficient reaction–convection–diffusion solvers based on a filtering technique that stabilizes the explicit treatment of the diffusion terms. We have demonstrated the potential of this numerical scheme with two examples in air quality models that usually requires the implicit treatment of diffusion terms. Further, we have shown that our solver has a high level of parallel efficiency for two-dimensional problems and excellent parallel scalability in three dimensions. Thanks to the high-order convergence of Fourier expansion, we can eventually have linear scalability, if the solution is smooth enough in space and the chemistry complex enough. As a matter of fact, one can filter out a larger proportion of high frequencies when the size of the problem grows.

We are now looking at large-scale computation with our new solver, for meta-computing architecture and possibly applications to the grid. In order to obtain scalable performance of our solver on large distributed parallel systems with $O(1000)$ processors, one must introduce a second level of parallelism with for example, the overlapping domain decomposition algorithm described in Reference [12]. This is an essential feature of our new method, because

at this second level of parallelism we get only local communication between subdomains with high load of parallel computation per subdomain. This is the best situation to obtain efficient parallelism on meta-computing architectures as demonstrated in References [24, 25]. We have recently run large-scale meta-computing experiments with standard Ethernet connections that, thanks to the overlapping domain decomposition scale, are compatible with the high latency and low bandwidth of the grid Reference [26]: the loss in efficiency due to the Ethernet network links is of the order of few per cents. However, the quotidian use of the grid is very challenging, because fault tolerance and load balancing will need to be addressed rigorously [27, 28].

ACKNOWLEDGEMENTS

We thank M. Resch and J. Morgan for many interesting discussions. We thank the High Performance Computing Center Stuttgart (HLRS) for giving us nice access on their world class computing resources as well as the 'Centre pour le Developpement du Calcul Scienctifique Parallele' (CDCSP) in University Lyon 1-Claude Bernard.

This work was supported in part by NSF Grant ACI-0305405. Although the research described in this has been funded in part by the US EPA through cooperative agreement CR829068-01 to UH it has been subjected to the agency's required peer and policy review, and therefore does not necessarily reflect the views of the agency and no official endorsement should be inferred.

REFERENCES

1. Berkvens PJF, Botchev MA, Verwer JG, Krol MC, Peters W. Solving vertical transport and chemistry in air pollution models. *MAS-R0023*, 31 August, 2000.
2. Dabdub D, Steinfeld JH. Parallel computation in atmospheric chemical modeling. *Parallel Computing* 1996; **22**:111–130.
3. Harley RA, Russel AG, McRae GJ, Cass GR, Seinfeld JH. Photochemical modeling of the Southern California air quality study. *Environmental Science and Technology* 1993; **27**:378–388.
4. Verwer JG, Hundsdorfer WH, Blom JG. Numerical time integration for air pollution models. *MAS-R9825*, *International Conference on Air Pollution Modelling and Simulation APMS'98*, http://www.cwi.nl
5. Mc Dougall SR, Anderson ARA, Chaplain MAJ, Sherratt JA. Mathematical modelling of flow through vascular networks: implication for tumor induced angiogenesis and chemotherapy strategies. *Bulletin of Mathematical Biology* 2002; **64**:673–702.
6. Painter KJ, Savill NJ, Shochat E. Computing evolution in brain tumours, http://www.ma.hw.ac.uk/painter/publications, submitted.
7. Wang H, Shi X, Ewing RE. An Ellam scheme for multidimensional advection–reaction equations and its optimal error estimate. *SIAM Journal on Numerical Analysis* 2001; **38**(6):1846–1885.
8. Pironneau O. *Finite Element Methods for Fluids*. Wiley: New York, 1989.
9. Knoll DA, Keyes DE. Jacobian-free Newton Krylov methods: a survey of approaches and applications. *Journal of Computational Physics* 2004; **193**:357–397.
10. Verwer JG, Sportisse B. A note on operator splitting in a stiff linear case. *MAS-R9830*, December 1998, http://www.cwi.nl
11. Ropp DL, Shadid JN, Ober CC. Studies of the accuracy of time integration methods for reaction–diffusion equations. *Journal of Computational Physics* 2004; **194**:544–574.
12. Garbey M, Kaper HG, Romanyukha N. A some fast solver for system of reaction–diffusion equations. In *Domain Decomposition Methods in Science and Engineering*, Debit N *et al.* (eds), *13th International Conference on Domain Decomposition DD13*, CIMNE, Bracelona, 2002; 387–394.
13. Elbern H. Parallelization and load balancing of a comprehensive atmospheric chemistry transport model. *Atmospheric Environment* 1997; **31**(21):3561–3574.
14. Droux JJ. Simulation numerique Bidimensionnelle et Tridimensionnelle de Processus de Solidification. *These No 901*, lausanne EPFL, 1990.
15. Lebedev VI. Explicit difference schemes for solving stiff problems with a complex or separable spectrum. *Computational Mathematics and Mathematical Physics* 2000; **40**(12):1801–1812.

16. Fitzgibbon WE, Garbey M, Dupros F. On a fast parallel solver for reaction–diffusion problems: application to air quality-simulation. In *Parallel Computational Fluid Dynamics—Practice and Theory*, Wilders P *et al.* (eds). North-Holland: Amsterdam, 2002; 111–118.
17. Petzold L. Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM Journal on Scientific and Statistical Computing* 1983; **4**(1):136–148.
18. Sandu A, Verwer JG, Van Loon M, Carmichael GR, Potra FA, Dadbud D, Seinfeld JH. Benchmarking stiff ODE solvers for atmospheric chemistry problems. I: implicit versus explicit. *Atmospheric Environment* 1997; **31**:3151–3166.
19. Thomas JW. *Numerical Partial Differential Equations*. Texts in Applied Mathematics. Springer: New York, 1995.
20. Gottlieb D, Chi-Wang Shu. On the Gibbs phenomenon and its resolution. *SIAM Review* 1997; **39**(4):644–668.
21. Fitzgibbon WE, Morgan J. Long term dynamics of a simple air quality model. *Technical Report*, Department of Mathematics, University of Houston, 2003.
22. Ecer A, Tarkan I, Lemoine E. Parallel CFD test case. *Parallel CFD Conference*, http://www.parcfd.org
23. Foster I. *Designing and Building Parallel Programs*: *Concepts and Tools for Parallel Software Engineering*. Addison-Wesley Longman Publishing Co. Inc.: Boston, MA, U.S.A., 1995.
24. Barberou N, Garbey M, Hess M, Resch M, Rossi T, Toivanen J, Dervout DT. On the efficient meta-computing of linear and nonlinear elliptic problems. *Journal of Parallel and Distributed Computing* (*Special issue on grid computing*) 2003; **63**(5):564–577.
25. Garbey M, Dervout DT. On some Aitken like acceleration of the Schwarz method. *International Journal for Numerical Methods in Fluids* 2002; **40**(12):1493–1513.
26. Garbey M, Hamon V, Keller R, Ltaief H. Fast parallel solver for the metacomputing of reaction–convection–diffusion problems. *Proceeding of the Parallel CFD04 Conference*, Canary Islands, Spain, 24–27 May 2004.
27. S. Browne, J. Dongarra, A. Trefethen. Numerical libraries and tools for scalable parallel cluster computing. *International Journal of High Performance Applications and Supercomputing* 2001; **15**(2):175–180.
28. Foster I, Kesselman C, Nick J, Tuecke S. Grid services for distributed system integration. *Computer* 2002; **35**(6):37–46.

