# Solving reaction–diffusion equations 10 times faster

Aly-Khan Kassam [1]

*Oxford University Computing Laboratory, Wolfson Bldg., Parks Road, Oxford OX1 3QD, UK.*

**Abstract**

The most popular numerical method for solving systems of reaction–diffusion equations continues to be a low order finite–difference scheme coupled with low order Euler time stepping. This paper extends previous 1D work and reports experiments that show that with high–order methods one can speed up such simulations for 2D and 3D problems by factors of 10–100. A short MATLAB code (2/3D) that can serve as a template is included.

*Key words:* ETD, exponential time–differencing, reaction–diffusion, stiff PDE, high–order time stepping, spectral method, Gray–Scott, complex Ginzburg–Landau, Schnakenberg.
*PACS:*

## 1 Introduction

Reaction–diffusion equations are interesting on many levels, displaying phenomena such as pattern formation far from equilibrium [15,22], Turing structures [3,30,33], nonlinear waves [24] such as solitons or spiral waves [12,27] and spatio–temporal chaos [23,26]. The efficient and accurate simulation of such systems, however, is difficult. This is because they couple a stiff diffusion term with a (typically) strongly nonlinear reaction term. When discretised this leads to large systems of strongly nonlinear, stiff ODEs. A general reaction–diffusion equation can be written as

$$u_t = D\nabla^2 u + N(u) \tag{1}$$

*Email address:* akk@comlab.ox.ac.uk (Aly-Khan Kassam).

[1] This work was supported by the Engineering and Physical Sciences Research Council (UK) and by the MathWorks, Inc.

where $D$ is the diffusion coefficient (assumed constant here, but not required to be so), and $N$ is the nonlinear reaction term. Coupled systems of two or more such equations are also of interest.

Although there has been much activity over the last few years in developing time stepping algorithms to deal with such problems [1,2,5,8,9,13,16,17,25,29], these do not seem to be well established – or indeed known – outside of the numerical analysis community. This is demonstrated by the fact that the most popular numerical algorithm currently being used to solve reaction–diffusion equations is a second order central difference scheme in space, coupled with an explicit forward Euler time stepping scheme (henceforth FD). This seems an attractive method for two reasons. Firstly it is easy to implement, and secondly many people feel that there is an element of 'overkill' when using a highly accurate high order method for a problem that does not require accurate solutions. It is unattractive, however, in that it is both inaccurate (for cases where this is important — studying spatio–temporal chaos, for example) and inefficient. Moreover finite–difference methods can sometimes lead to spurious, that is nonphysical, solutions [14] – a fate less often suffered by higher order spectral methods.

We have previously shown [16] how to stabilise a new high–order method, ETDRK4, and results from experiments in 1D proved it to be a powerful method. This paper presents results from experiments that show in 2D and 3D the ETDRK4 advantage grows further. These experiments show that one can typically beat the standard FD by a factor of 10–100 in computation time. As the space discretisation is done with a Fourier spectral method, one can often beat the standard FD scheme by orders of magnitude of accuracy as well.

Section 2 describes the numerical method. Section 3 compares its performance against the standard FD scheme for three reaction–diffusion systems in 2D and 3D, and finally the Appendix provides a MATLAB code to solve one variable 2D and 3D reaction–diffusion problems, for use as a template.

Although this paper focuses on reaction–diffusion equations, this method can apply equally to equations that have a similar form. This includes nonlinear wave problems and reaction–diffusion–convection problems, among others. Several examples are considered in [16].

## 2 Numerical method – Fourier spatial discretisation and exponential time differencing

The spatial discretisation for all of the equations studied here was done using a Fourier spectral method with periodic boundary conditions [4,6,10,32], and the time stepping was done with a fourth order exponential time differencing Runge–Kutta method (ETDRK4)[8,18]. Both methods are briefly described

here, starting with the Fourier spectral method. This discussion is adapted from [32].

Given a function $v$ which is periodic on an appropriate spatial grid $x_j$, one begins by defining the *discrete Fourier transform* (DFT)

$$\widehat{v} = h \sum_{j=1}^{N} e^{-ikx_j} v_j, \qquad k = -\frac{N}{2} + 1, \ldots, \frac{N}{2} \tag{2}$$

and the inverse DFT

$$v_j = \frac{1}{2\pi} \sum_{k=-N/2+1}^{N/2} e^{ikx_j} \widehat{v}_k, \qquad j = 1, \ldots, N. \tag{3}$$

where $k$ are the Fourier wavenumbers. With these definitions, one can approximate the derivatives of $v$ on the grid via the following procedure:

- given $v$ compute $\widehat{v}$ using (2)
- define $\widehat{w}$ to be $ik\widehat{v}$
- compute $w$ (the derivative of $v$) on the grid using (3).

Applying this method to (1) and leaving all the time stepping in Fourier space gives the following system of ODEs:

$$\widehat{u}_t = -Dk^2 \widehat{u} + \widehat{N(u)} \tag{4}$$

The linear term of (1) is now diagonal – one of the great advantages of using a Fourier spectral method. Importantly, the nonlinear term is evaluated in physical space and then transformed to Fourier space. This can cause problems with aliasing [4,6], and one has to be careful to filter high frequencies appropriately.

Cox and Matthews' fourth order exponential time differencing formula of Runge–Kutta type [8] was used to advance the ODE (4). The formulae are:

$$\begin{aligned}
a_n &= e^{\mathbf{L}h/2} u_n + \mathbf{L}^{-1}(e^{\mathbf{L}h/2} - \mathbf{I})\mathbf{N}(u_n, t_n), \\
b_n &= e^{\mathbf{L}h/2} u_n + \mathbf{L}^{-1}(e^{\mathbf{L}h/2} - \mathbf{I})\mathbf{N}(a_n, t_n + h/2), \\
c_n &= e^{\mathbf{L}h/2} a_n + \mathbf{L}^{-1}(e^{\mathbf{L}h/2} - \mathbf{I})(2\mathbf{N}(b_n, t_n + h/2) - \mathbf{N}(u_n, t_n)), \\
u_{n+1} &= e^{\mathbf{L}h} u_n + h^{-2}\mathbf{L}^{-3}\{[-4 - \mathbf{L}h + e^{\mathbf{L}h}(4 - 3\mathbf{L}h + (\mathbf{L}h)^2)]\mathbf{N}(u_n, t_n) \\
&+ 2[2 + \mathbf{L}h + e^{\mathbf{L}h}(-2 + \mathbf{L}h)](\mathbf{N}(a_n, t_n + h/2) + \mathbf{N}(b_n, t_n + h/2)) \\
&+ [-4 - 3\mathbf{L}h - (\mathbf{L}h)^2 + e^{\mathbf{L}h}(4 - \mathbf{L}h)]\mathbf{N}(c_n, t_n + h)\}.
\end{aligned}$$

where, for reaction–diffusion problems, $\mathbf{L}$ is the linear diffusion term, $-Dk^2$, and $\mathbf{N}$ is the nonlinear reaction term.

Details of the derivation of the scheme can be found in [8,18]. In this form the method ETDRK4 needs some kind of numerical stabilisation to work properly. We handle this by computing certain coefficients by means of integrals around contours in the complex plane. Details of our stabilisation technique can be found in [16].

In summary, the numerical method is spectrally accurate in space, fourth order accurate in time, and uses periodic boundary conditions. The nonlinear terms are evaluated in physical space, but time stepping is carried out in Fourier space and one must be careful about de–aliasing.

## 3 Numerical results and comparisons

We have run a number of numerical simulations on a range of reaction–diffusion problems using this method. We present results from three equations in 2D and 3D:

**Schnakenberg equations** [21,31]

$$\begin{aligned} \frac{\partial u}{\partial t} &= \nabla^2 u + \gamma \left( a - u + u^2 v \right), \\ \frac{\partial v}{\partial t} &= d\nabla^2 v + \gamma \left( b - u^2 v \right), \end{aligned} \tag{5}$$

where $a = 1, \quad b = 0.9, \quad \mathrm{d} = 10$ and $\gamma = 1$, are positive constants, and with the initial condition

$$\begin{aligned} u(x, y, 0) &= 1 - \mathrm{e}^{-10((x-\Lambda/2)^2 + (y-\Lambda/2)^2)} \\ v(x, y, 0) &= \mathrm{e}^{\left[-10((x-\Lambda/2)^2 + 2(y-\Lambda/2)^2)\right]} \end{aligned} \tag{6}$$

where $\Lambda$ is the length of the domain. This equation is known to generate stationary Turing structures.

**Complex Ginzburg–Landau equation** [19]

$$u_t = (1 + ib)\, \nabla^2 u - (1 + ia)\, u|u|^2 \tag{7}$$

with both smooth and random initial conditions, in both two and three dimensions. The smooth initial conditions take the form of a series of Gaussian pulses:

$$u(x, y, 0) = \mathrm{e}^{-20((X-\Lambda/3)^2 + (Y-\Lambda/3)^2)/\Lambda} \tag{8}$$

$$
\begin{aligned}
&- \mathrm{e}^{-20((X-\Lambda/2)^2+(Y-\Lambda/2)^2)/\Lambda} \\
&+ \mathrm{e}^{-20((X-\Lambda/2)^2+(Y-\Lambda/3)^2)/\Lambda}
\end{aligned}
$$

In this investigation, $b = 0$ and typically $a = 1.3$. This is a prototype equation for generation and analysis of spiral waves.

**Gray–Scott equations** [11,28]

$$
\begin{aligned}
\frac{\partial u}{\partial t} &= D_u \nabla^2 u - uv^2 + F(1-u) \\
\frac{\partial v}{\partial t} &= D_v \nabla^2 v + uv^2 - (F+k)v
\end{aligned} \tag{9}
$$

where $D_u > 0$ and $D_v > 0$ are diffusion parameters while $F$ and $k$ can be viewed as bifurcation parameters. Again, this was studied in 2D and 3D, and for a variety of initial conditions. The smooth initial conditions used to check convergence were:

$$
\begin{aligned}
u(x,y,0) &= 1 - \mathrm{e}^{-150((x-\Lambda/2)^2+(y-\Lambda/2)^2)} \\
v(x,y,0) &= \mathrm{e}^{-150((x-\Lambda/2)^2+2(y-\Lambda/2)^2)}
\end{aligned} \tag{10}
$$

This equation produces a wide range of patterns including spots, stripes and spatio–temporal chaos.

To check the convergence we fix the time step and run several simulations with increasing numbers of grid points. We then refine the time step and re-run the convergence test. The error was calculated by comparing the successive values of a point in the center of the grid. Figures 1 – 3 show the results of this experiment for both FD and ETDRK4. In every case, ETDRK4 is faster than FD by a factor of 10–100.

In 3D these tests make ETDRK4 seem *worse* than it is. As the convergence tests run over a relatively short time, the time needed to construct the ETDRK4 coefficients is significant and this makes ETDRK4 look slower than it is. As these coefficients are only calculated once at the beginning of the simulation, the initialisation cost becomes less significant as the simulation time increases. In a test that neglected initialisation costs, ETDRK4 timestepping performed about 20 times faster than FD in 3D.

This section ends with a selection of figures that show 2D and 3D evolution of the three equations. Three dimensional simulations are still rare because of the large memory requirements and CPU time that low order methods require. To reproduce Figures 6 – 10 using the standard FD scheme would be a hard task.

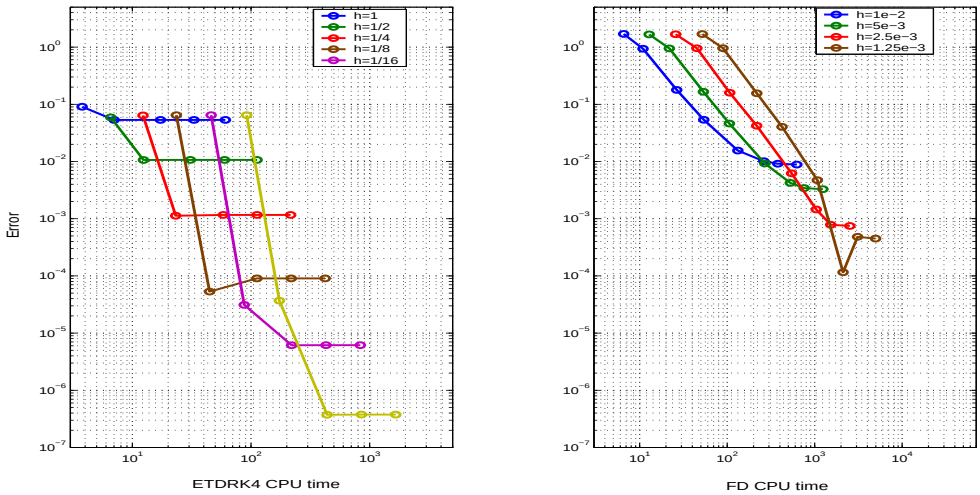


Fig. 1. Convergence of ETDRK4 and FD for the 2D complex Ginzburg–Landau equation. ETDRK4 achieves an accuracy of $10^{-3}$ in about 20 seconds compared with 1000 seconds for FD.

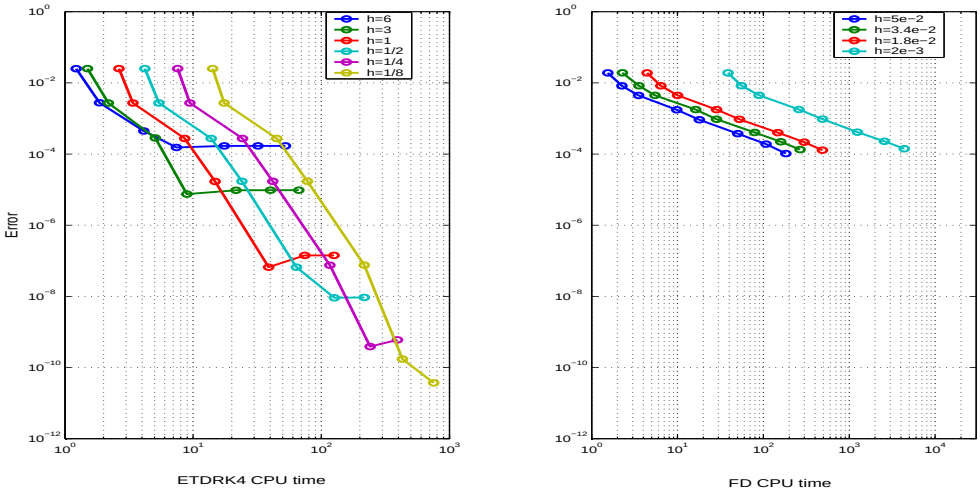


Fig. 2. Convergence of ETDRK4 and FD for the 2D Gray–Scott equations. ETDRK4 achieves an accuracy of $10^{-4}$ in under 10 seconds compared with about 200 seconds for FD.

## 4 Conclusion

Although some researchers are making use of higher order and more efficient methods for solving reaction–diffusion problems, this remains the exception rather than the rule. Our results in 2D and 3D demonstrate a strong case for abandoning the FD method in favour of ETDRK4 coupled to a Fourier spectral method. One can gain orders of magnitude of accuracy but, more importantly for many applications, one gains a factor of 10–100 in computer time.

In the words of Leppänen et al. [20], higher order methods are literally opening up a whole new dimension in the numerical solution of reaction–diffusion problems.

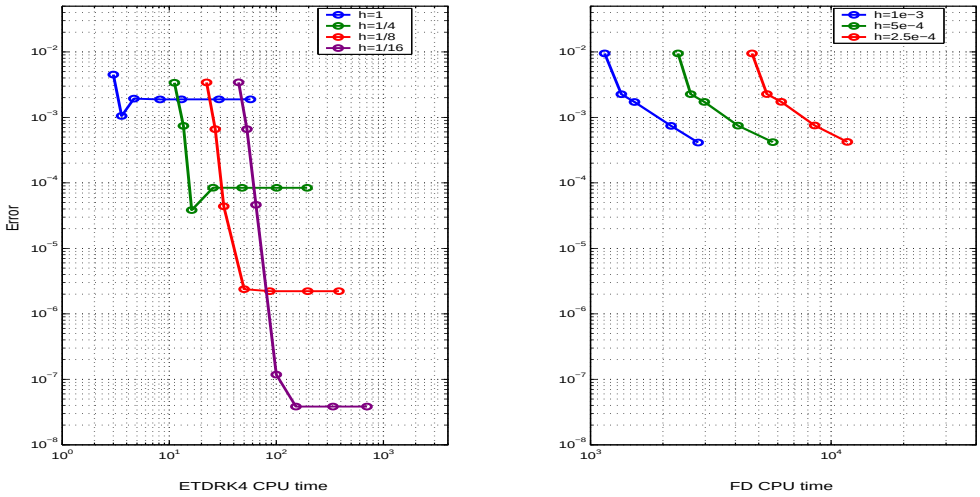


Fig. 3. Convergence of ETDRK4 and FD for the 2D Schnakenberg equations. ETDRK4 achieves an accuracy of $10^{-4}$ in about 15 seconds compared with about 2000 seconds for FD.

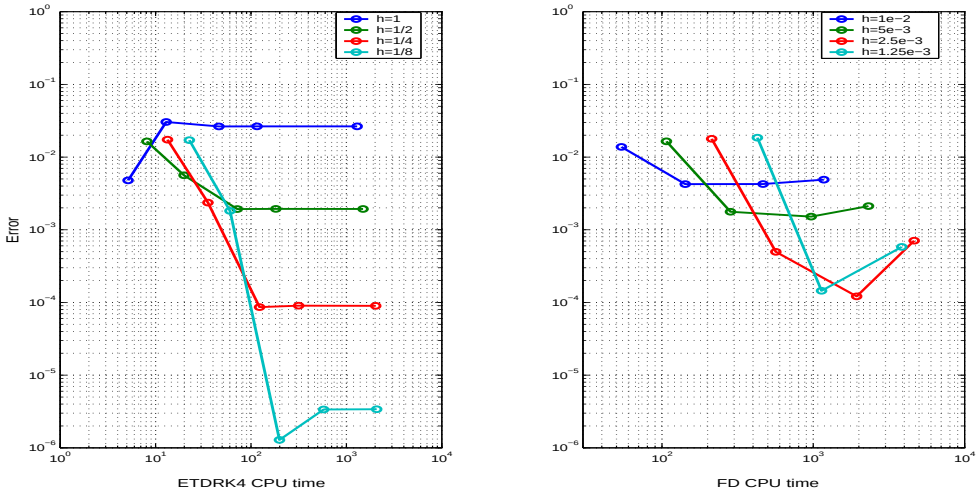


Fig. 4. Convergence of ETDRK4 and FD for the 3D complex Ginzburg–Landau equations. ETDRK4 achieves an accuracy of $10^{-4}$ in about 50 seconds compared with about 500 seconds for FD.

## Acknowledgments

The author would like to offer sincere thanks to Prof. L. N. Trefethen for his invaluable advice and his infectious enthusiasm for seeing good algorithms implemented well. Thanks are also due to P. Matthews, A. Iserles, P. Maini, M. Hochbruck and U. Ascher all of whom have offered much advice and insight into the world of time stepping and reaction–diffusion equations.

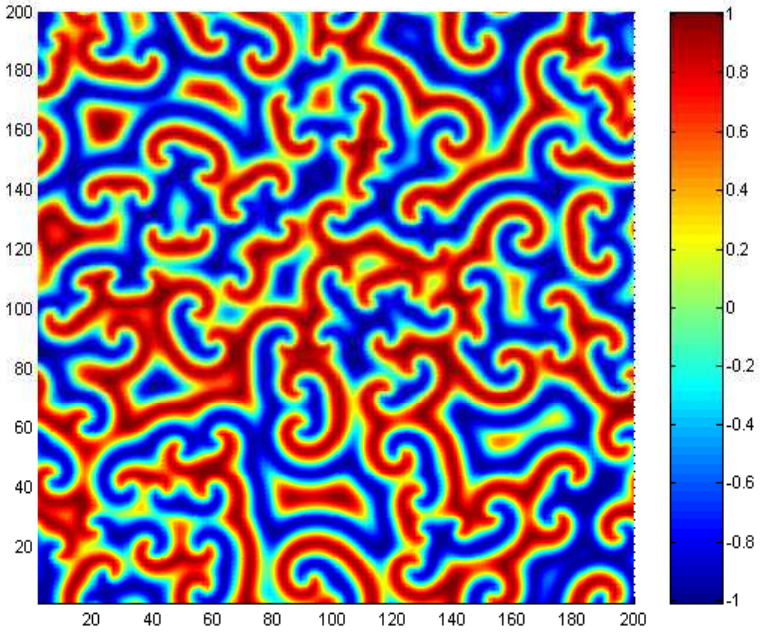


Fig. 5. Solution of the 2D complex Ginzburg–Landau equation on $[0, 200] \times [0, 200]$ with $a = 1.3$, $T_{final} = 150$, and random initial conditions ($N = 128$).

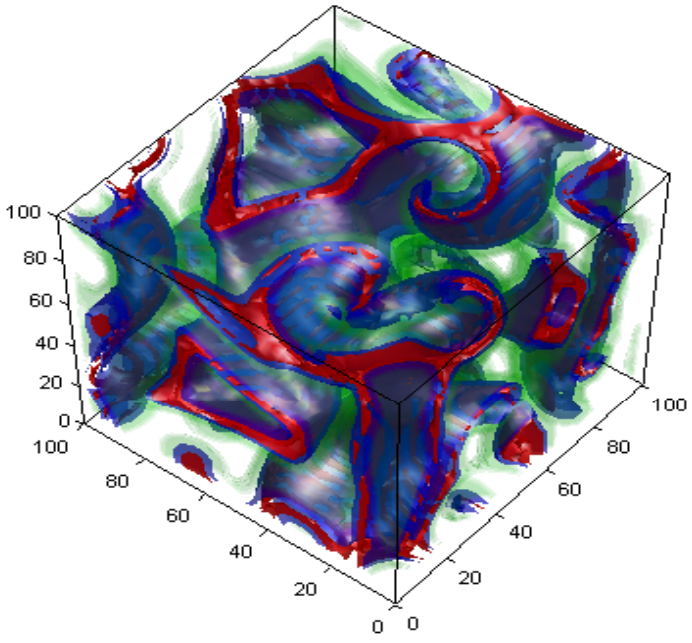


Fig. 6. Solution of the 3D complex Ginzburg–Landau equation on $[0, 100]^3$ with $a = 1.1$, $T_{final} = 300$ ($N = 40$).

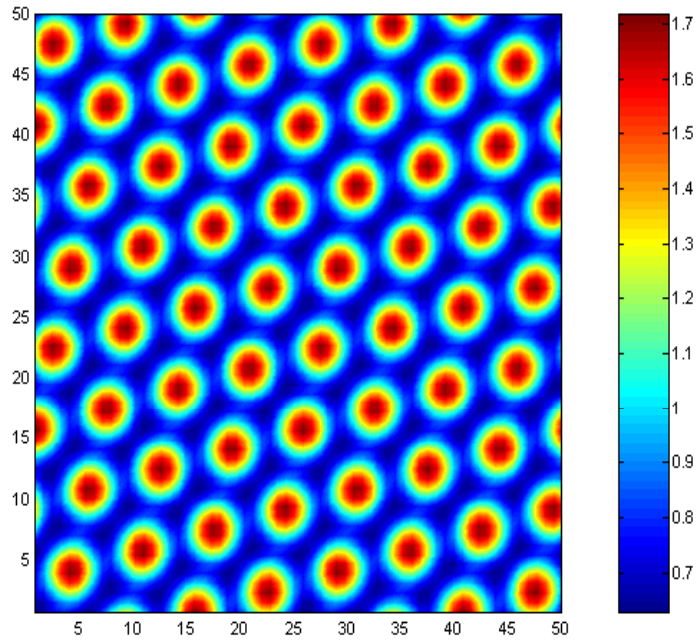


Fig. 7. Solution of the 2D Schnakenberg equations on $[0, 50] \times [0, 50]$ with $\gamma = 3$, $T_{final} = 1000$ ($N = 64$).

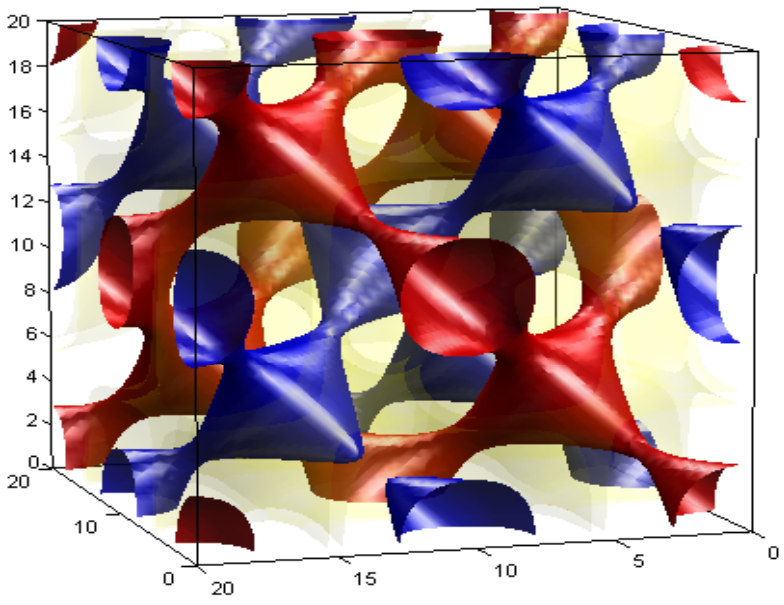


Fig. 8. Solution of the 3D Schnakenberg equations on $[0, 20]^3$ with $\gamma = 1$, $T_{final} = 400$ ($N = 64$).

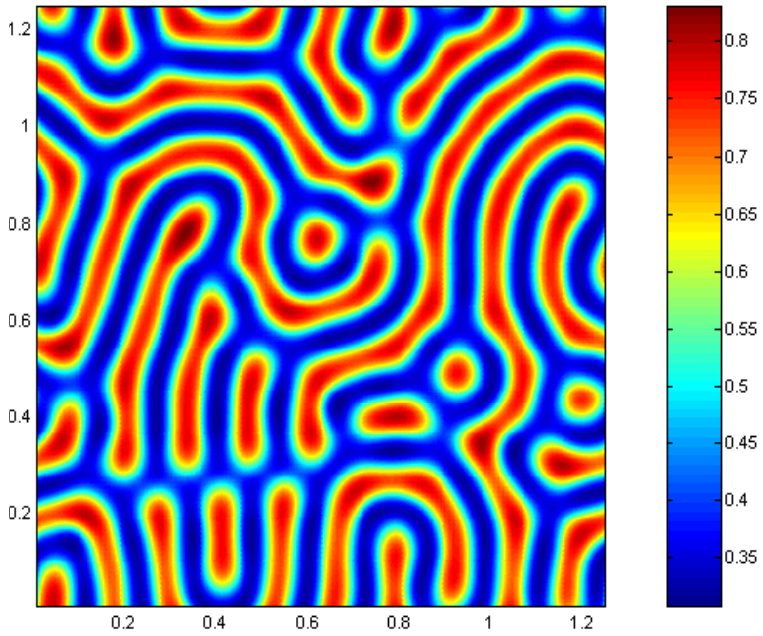


Fig. 9. Solution of the 2D Gray–Scott equations on $[0, 1.25] \times [0, 1.25]$ with $F = 0.04, \quad k = 0.06, \quad T_{final} = 25000$ $(N = 128)$.

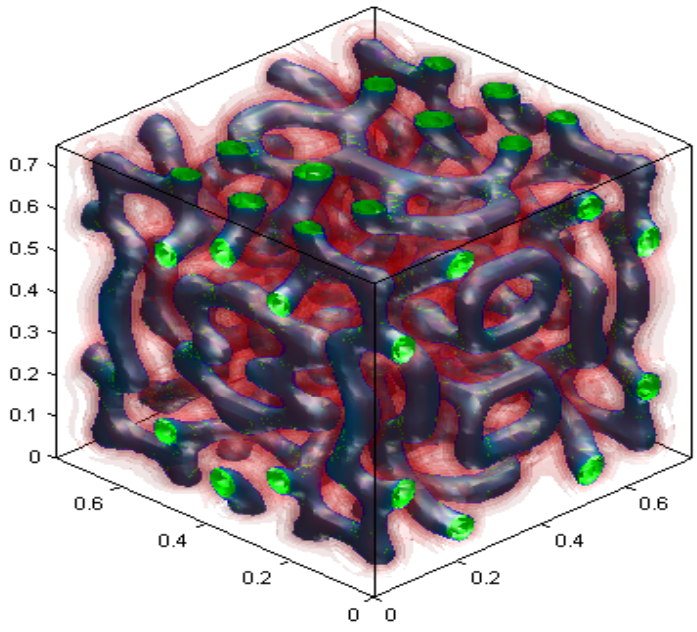


Fig. 10. Solution of the 3D Gray–Scott equations on $[0, 0.75]^3$ with $F = 0.025, \quad k = 0.06, \quad T_{final} = 4400$ $(N = 48)$.

## A MATLAB code template

```
% 2D {3D} complex Gizburg-Landau equation on a periodic domain.
%          u_t = u -(1+iA)u*abs(u)^2 + D(u_xx + u_yy)
% For a 3D code, replace relevant sections in code with code in { }.
% For a different equation, change the diffusion paramter, D, and
% nonlinear function, g. In Nv, Na, Nb, Nc, g must be called properly.

%======================== CUSTOM SET UP  ========================
A=1.3; lam=200; N=128; D=1; h=1; %domain/grid size, diff par, timestep
g=inline('u-(1+1i*A)*u.*(abs(u).^2)','u','A'); %nonlinear function
%======================== GENERIC SET UP  ========================
x=(lam/N)*(1:N)'; [X,Y]=ndgrid(x,x);          %{[X,Y,Z]=ndgrid(x,x,x)}
u=randn(N,N)*0.1;  v=fftn(u);  %random IC.              {randn(N,N,N)}
k=[0:N/2-1 0 -N/2+1:-1]'/(lam/(2*pi)); %wave numbers
[xi,eta]=ndgrid(k,k);  %2D wave numbers. {[xi,eta,zeta]=ndgrid(k,k,k)}
L=-D*(eta.^2+xi.^2);   %2D Laplacian.      {-D*(eta.^2+xi.^2+zeta.^2)}
Fr=logical(zeros(N,1)); %High frequencies for de-aliasing
Fr([N/2+1-round(N/6) : N/2+round(N/6)])=1;
[alxi,aleta]=ndgrid(Fr,Fr);    %{[alxi,aleta,alzeta]=ndgrid(Fr,Fr,Fr)}
ind=alxi | aleta; %de-aliasing index.          {alxi | aleta | alzeta}
%==============  PRECOMPUTING ETDRK4 COEFFS  ====================
E=exp(h*L); E2=exp(h*L/2);
M=16; % no. of points for complex mean
r=exp(i*pi*((1:M)-0.5)/M); % roots of unity
L=L(:); LR=h*L(:,ones(M,1))+r(ones(N^2,1),:);      %{r(ones(N^3,1),:)}
Q=h*real(mean(           (exp(LR/2)-1)./LR              ,2));
f1=h*real(mean(  (-4-LR+exp(LR).*(4-3*LR+LR.^2))./LR.^3  ,2));
f2=h*real(mean(     (4+2*LR+exp(LR).*(-4+2*LR))./LR.^3   ,2));
f3=h*real(mean(  (-4-3*LR-LR.^2+exp(LR).*(4-LR))./LR.^3  ,2));
f1=reshape(f1,N,N); f2=reshape(f2,N,N); f3=reshape(f3,N,N);
L=reshape(L,N,N); Q=reshape(Q,N,N); clear LR       %{reshape(*,N,N,N)}
%===================  TIME STEPPING LOOP    ======================
tmax=150; nmax=round(tmax/h);
for n = 1:nmax
   t=n*h;        %**Nonlinear terms are evaluated in physical space**
   Nv=fftn( g(ifftn(v),A) );        %Nonlinear evaluation. g(u,*)
   a=E2.*v + Q.*Nv;                 %Coefficient 'a' in ETDRK formula
   Na=fftn( g(ifftn(a),A) );        %Nonlinear evaluation. g(a,*)
   b=E2.*v + Q.*Na;                 %Coefficient 'b' in ETDRK formula
   Nb=fftn( g(ifftn(b),A) );        %Nonlinear evaluation. g(b,*)
   c=E2.*a + Q.*(2*Nb-Nv);          %Coefficient 'c' in ETDRK formula
   Nc=fftn( g(ifftn(c),A) );        %Nonlinear evaluation. g(c,*)
   v=E.*v + Nv.*f1 + (Na+Nb).*f2 + Nc.*f3;   %update
   v(ind) = 0;   % High frequency removal --- de-aliasing
end
u=real(ifftn(v));
```

## References


[1] U. M. Ascher, S. J. Ruuth and B. T. R. Wetton, Implicit-explicit methods for time-dependent partial differential equations, *SIAM J. Numer. Anal.* **32**, 797–823 (1995).

[2] U. M. Ascher, S. J. Ruuth and R. J. Spiteri, Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations, *Appl. Numer. Math.* **25**, 151–167 (1997).

[3] P. Borckmans, G. Dewel, A. De Wit, E. Dulos, J. Boissonade, F. Gauffre and P. De Kepper, Diffusive instabilities in chemical reactions, *Int. J. Bif. Chaos* **12**(11), 2307–2332 (2002).

[4] J. P. Boyd, *Chebyshev and Fourier Spectral Methods* (Dover, 2001). Online edition at: http://www-personal.engin.umich.edu/∼jpboyd/.

[5] M. P. Calvo, J. de Frutos and J. Novo, Linearly implicit Runge-Kutta methods for advection-diffusion equations, *Appl. Numer. Math.* **37**, 535–549 (2001).

[6] C. Canuto, M. Y. Hussaini, A. Quarteroni and T. A. Zang, *Spectral Methods in Fluid Dynamics* (Springer-Verlag, Berlin, 1988).

[7] M. C. Cross and P. C. Hohenberg, Pattern formation outside of equilibrium, *Rev. Mod. Phys.* **81**(3), 851–1112 (1993).

[8] S. M. Cox and P. C. Matthews, Exponential time differencing for stiff systems, *J. Comp. Phys.* **176**, 430–455 (2002).

[9] T. A. Driscoll, A composite Runge–Kutta method for the spectral solution of semilinear PDEs, *J. Comp. Phys.* **182**, 357–367 (2002).

[10] B. Fornberg, *A Practical Guide to Pseudospectral Methods* (Cambridge University Press, Cambridge, UK, 1996).

[11] P. Gray and S. K. Scott, Autocatalytic reactions in the isothermal, continuous stirred tank reactor: isolas and other forms of multistability, *Chem. Eng. Sci.***38**, 29–43 (1983).

[12] P. Hagan, Spiral waves in reaction–diffusion equations, *SIAM J. Appl. Math.* **42**(4), 762–786 (1982).

[13] M. Hochbruck, C. Lubich and H. Selhofer, Exponential integrators for large systems of differential equations, *SIAM J. Sci. Comp.* **19**, 1552–1574 (1998).

[14] W. B. Jones and J. O'Brian, Psuedo-spectral methods and linear instabilities in reaction-diffusion fronts, *Chaos* **6**, 219–228 (1996).

[15] R. Kapral, Pattern formation in chemical systems, *Physica D* **86**, 149–157 (1995).

[16] A-K Kassam and L. N. Trefethen, Fourth–order time stepping for stiff PDEs, *SIAM J. Sci. Comp.* to appear.

[17] C. A. Kennedy and M. H. Carpenter, Additive Runge–Kutta schemes for convection–diffusion–reaction equations, *Appl. Numer. Math.* **44**, 139–181 (2003).

[18] S. Krogstad, Generalized integrating factor methods for stiff PDEs, *preprint*.

[19] Y. Kuramoto, *Chemical Oscillations, Waves and Turbulence* (Springer, Berlin, 1984).

[20] T. Leppänen, M. Karttunen, K. Kaski, R. Barrio and L. Zhang, A new dimension to Turing patterns, *Physica D* **168–169**, 35–44 (2002).

[21] A. Madzvamuse, *A numerical approach to the study of spatial pattern formation* D.Phil. thesis, Oxford University Computing Laboratory (2000).

[22] P. K. Maini, K. J. Painter and H. N. P. Chau, Spatial pattern formation in chemical and biological systems, *J. Chem. Soc. Faraday Trans.* **93**(20), 3601–3610 (1997).

[23] J. H. Merkin, V. Petrov, S. K. Scott and K. Showalter, Wave-induced chemical chaos, *Phys. Rev. Lett.* **76**(3), 546–549 (1996).

[24] A. G. Merzhanov and E. N. Rumanov, Physics of reaction waves, *Rev. Mod. Phys.* **71**(4), 1173–1211 (1999).

[25] D. R. Mott, E. S. Oran and B. van Leer, A quasi-steady state solver for the stiff ordinary differential equations of reaction kinetics, *J. Comp. Phys* **164**, 407–428 (2000).

[26] Y. Nishiura and D. Ueyama, Spatio–temporal chaos for the Gray–Scott model, *Physica D* **150**, 137–162 (2001).

[27] T. Ohta, Y. Hayase and R. Kobayashi, Spontaneous formation of concentric waves in a two–component reaction–diffusion system, *Phys. Rev. E* **54**(6), 6074–6082 (1996).

[28] J. E. Pearson, Complex patterns in a simple system, *Science* **261**, 189–192 (1993).

[29] S. J. Ruuth, Implicit-explicit methods for reaction-diffusion problems in pattern formation, *J. Math. Biol.* **34**(2), 148–176 (1995).

[30] R. Satnoianu, M. Menzinger and P. K. Maini, Turing patterns in general systems, *J. Math. Biol.* **41**, 493–512 (2000).

[31] J. Schnakenberg, Simple chemical reaction systems with limit cycle behaviour, *J. Theor. Biol.* **81**, 380–400 (1979).

[32] L. N. Trefethen, *Spectral Methods in* MATLAB (SIAM, Philadelphia 2000).

[33] A. M. Turing, The chemical basis of morphogenesis, *Phil. Trans. R. Soc. Lond.* **B237**, 37–72 (1952).