Computational physics

BY YOUJUN HU

Institute of Plasma Physics, Chinese Academy of Sciences

Email: yjhu@ipp.cas.cn

1 Introduction

Physics becomes concrete, impressive, and fun when we compute it numerically and visualize the process by graphics. Computational physics are primarily about numerically solving three types of partial differential equations (PDEs), namely hyperbolic, parabolic, and elliptic PDEs, which respectively correspond to advection (wave) equations, diffusion equations, and Poisson's equations.

2 Advection equation

In one-dimensional case, an advection equation takes the following form:

$$\frac{\partial y}{\partial t} = -c\frac{\partial y}{\partial x}, \tag{1}$$

where c is a constant. A natural choice of differencing scheme for the above equation is

$$\frac{y_i^{(n+1)} - y_i^{(n)}}{\Delta t} = \frac{y_{(i+1)}^{(n)} - y_{(i-1)}^{(n)}}{2\Delta x}, \tag{2}$$

which however is unconditionally unstable (tested by me numerically. the stability analysis can prove that the above scheme is unconditional unstable[2]). The Lax-Friedrichs scheme modifies the above scheme to the following form:

$$\frac{y_i^{(n+1)} - \frac{y_{(i-1)}^{(n)} + y_{(i+1)}^{(n)}}{2}}{\Delta t} = -c\frac{y_{(i+1)}^{(n)} - y_{(i-1)}^{(n)}}{2\Delta x}, \tag{3}$$

which is stable if the CFL condition is satisfied. However this scheme introduces heavy damping in the solution, as is shown in Fig 1. The Lax-Friedrichs scheme is an explicit scheme. Let us try implicit schemes. One natural choice of implicit scheme is of the following form:

$$\frac{y_i^{(n+1)} - y_i^{(n)}}{\Delta t} = -\frac{1}{2}c\left[\frac{y_{(i+1)}^{(n+1)} - y_{(i-1)}^{(n+1)}}{2\Delta x} + \frac{y_{(i+1)}^{(n)} - y_{(i-1)}^{(n)}}{2\Delta x}\right], \tag{4}$$

which is called the Crank–Nicolson implicit scheme. An implicit scheme usually requires that an linear equations system be solved because the unknown future values are usually coupled together. The scheme (4) can be organized in the following form

$$y_i^{(n+1)} + \frac{\Delta t}{2}c\left[\frac{y_{(i+1)}^{(n+1)} - y_{(i-1)}^{(n+1)}}{2\Delta x}\right] = y_i^{(n)} - \frac{\Delta t}{2}c\left[\frac{y_{(i+1)}^{(n)} - y_{(i-1)}^{(n)}}{2\Delta x}\right], \tag{5}$$

which is a traditional equation system. Figure 1 compares the results calculated by the Lax-Friedrichs scheme and the Crank–Nicolson scheme, which shows that no damping is introduced by the Crank-Nicolson scheme.

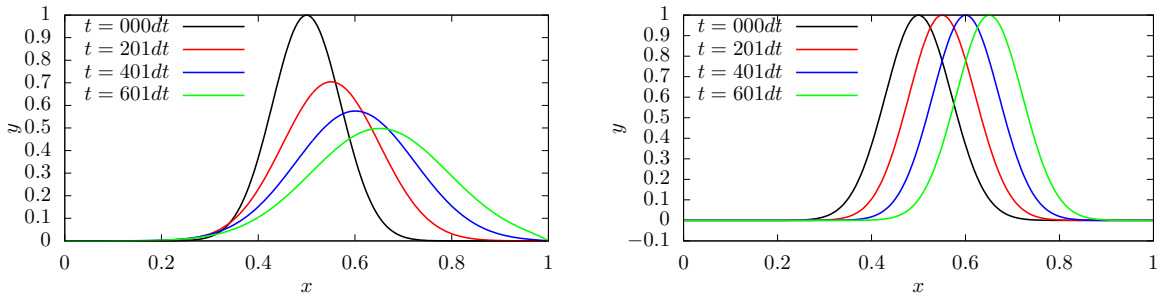

Figure 1. Evolution of the waveform computed by the Lax-Friedrichs scheme (left) and the Crank–Nicolson implicit scheme (right). Simulations are performed with the Initial condition given by $y(x, t=0) = \exp(-100(x - 0.5)^2)$, time-step size $dt = 0.05dx\,/\,c$, grid spacing $dx = 1\,/\,(N_x - 1)$, $N_x = 200$. Both schemes give correct the propagation speed, but the Lax-Friedrichs scheme introduces heavy damping in the solution.

3 Wave equation

A wave equation in one-dimension takes the following form:

$$\frac{\partial^2 y}{\partial t^2} = c\frac{\partial^2 y}{\partial x^2}, \tag{6}$$

which is a second order differential equation and can be written as two coupled advection equations. Define a new function ξ by

$$\frac{\partial \xi}{\partial t} = \frac{\partial y}{\partial x} \tag{7}$$

Then Eq. (6) can be written as

$$\frac{\partial^2 y}{\partial t^2} = c\frac{\partial \xi}{\partial t \partial x}, \tag{8}$$

which can be simplified as

$$\frac{\partial y}{\partial t} = c\frac{\partial \xi}{\partial x}. \tag{9}$$

Equation (7) and (9) are two couple advection equations.

3.1 Maxwell's equation in 1D case

For Maxwell's equation in one-dimension case, there are two independent TEM modes, one of which is described by

$$\frac{\partial B_y}{\partial t} = \frac{\partial E_z}{\partial x}, \tag{10}$$

$$\frac{\partial E_z}{\partial t} = c^2\frac{\partial B_y}{\partial x}, \tag{11}$$

where $c = 1\,/\,\sqrt{\mu_0 \varepsilon_0}$ is the speed of light in vacuum. In an electromagnetic wave, E_z is c times of B_z in SI units. To make the two variables in the above equation takes similar magnitude, define $\overline{E}_z = E_z\,/\,c$. Then using $(\overline{E}_z, B_y)$ as variables, the above equations are written

$$\frac{\partial B_y}{\partial t} = c\frac{\partial \overline{E}_z}{\partial x}, \tag{12}$$

$$\frac{\partial \overline{E}_z}{\partial t} = c\frac{\partial B_y}{\partial x}, \tag{13}$$

3.2 The Lax scheme

Similar to the case of advection equation, the following simple differencing scheme for the vacuum TEM equations (12) and (13):

$$\frac{B_{yi}^{(n+1)} - B_{yi}^{(n)}}{\Delta t} = c\frac{\overline{E}_{z\,(i+1)}^{(n)} - \overline{E}_{z\,(i-1)}^{(n)}}{2\Delta x}. \tag{14}$$

$$\frac{\overline{E}_{zi}^{(n+1)} - \overline{E}_{zi}^{(n)}}{\Delta t} = c\frac{B_{y\,(i+1)}^{(n)} - B_{y\,(i-1)}^{(n)}}{2\Delta x}. \tag{15}$$

is unstable (tested numerically by me). The Lax scheme modifies the above scheme to the following form:

$$\frac{B_{yi}^{(n+1)} - \frac{B_{y\,(i-1)}^{(n)} + B_{y\,(i+1)}^{(n)}}{2}}{\Delta t} = c\frac{\overline{E}_{z\,(i+1)}^{(n)} - \overline{E}_{z\,(i-1)}^{(n)}}{2\Delta x}. \tag{16}$$

$$\frac{\overline{E}_{zi}^{(n+1)} - \frac{\overline{E}_{z\,(i-1)}^{(n)} + \overline{E}_{z(i+1)}^{(n)}}{2}}{\Delta t} = c\frac{B_{y\,(i+1)}^{(n)} - B_{y\,(i-1)}^{(n)}}{2\Delta x}. \tag{17}$$

I tested this and found it induces heavy damping as it does in the case of advection equation.

3.3 The Crank–Nicolson scheme

Let us try the Crank-Nicolson implicit scheme:

$$\frac{B_{y(i)}^{(n+1)} - B_{y(i)}^{(n)}}{\Delta t} = \frac{1}{2}c\left(\frac{\overline{E}_{z(i+1)}^{(n+1)} - \overline{E}_{z(i-1)}^{(n+1)}}{2\Delta x} + \frac{\overline{E}_{z(i+1)}^{(n)} - \overline{E}_{z(i-1)}^{(n)}}{2\Delta x}\right), \tag{18}$$

$$\frac{\overline{E}_{z(i)}^{(n+1)} - \overline{E}_{z(i)}^{(n)}}{\Delta t} = \frac{1}{2}c\left(\frac{B_{y(i+1)}^{(n+1)} - B_{y(i-1)}^{(n+1)}}{2\Delta x} + \frac{B_{y(i+1)}^{(n)} - B_{y(i-1)}^{(n)}}{2\Delta x}\right). \tag{19}$$

The above differencing scheme can be organized in the following form:

$$B_{y(i)}^{(n+1)} - \frac{\Delta t}{2}c\left(\frac{\overline{E}_{z(i+1)}^{(n+1)} - \overline{E}_{z(i-1)}^{(n+1)}}{2\Delta x}\right) = \frac{\Delta t}{2}c\left(\frac{\overline{E}_{z(i+1)}^{(n)} - \overline{E}_{z(i-1)}^{(n)}}{2\Delta x}\right) + B_{y(i)}^{(n)}, \tag{20}$$

$$\overline{E}_{z(i)}^{(n+1)} - \frac{\Delta t}{2}c\left(\frac{B_{y(i+1)}^{(n+1)} - B_{y(i-1)}^{(n+1)}}{2\Delta x}\right) = \frac{\Delta t}{2}c\left(\frac{B_{y(i+1)}^{(n)} - B_{y(i-1)}^{(n)}}{2\Delta x}\right) + \overline{E}_{z(i)}^{(n)}. \tag{21}$$

which is a linear equation system for $\left(B_{yi}^{(n+1)},\ \overline{E}_{zi}^{(n+1)}\right)$ with $i = 1, 2, \ldots N_x$, where N_x is the number of grids in the x direction. This linear system is solved by using LU decomposition of the coefficient matrix (the LU decomposition can be viewed as the matrix form of Gaussian elimination.). The evolution of the wave form calculated by this scheme is plotted in Fig. 2.

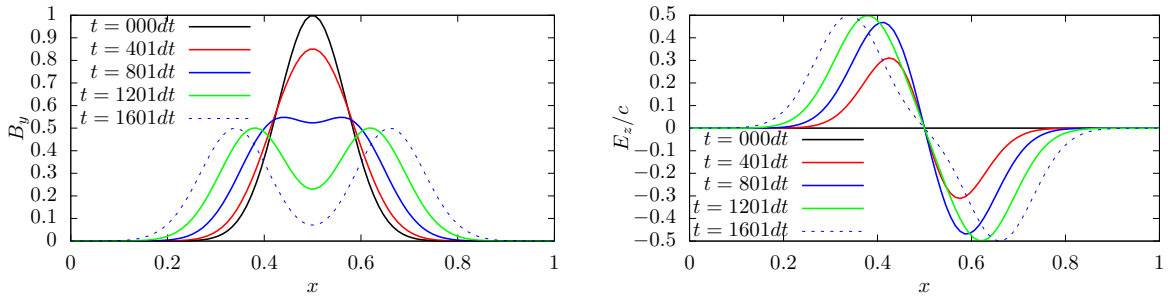

Figure 2. Evolution of the TEM waveform (B_y, E_z) computed by the Crank–Nicolson implicit scheme. Simulations are performed with the initial condition $E_z(x, t=0) = 0$, $B_y(x, t=0) = \exp(-(x-x_0)^2/(0.1L)^2)$, $x_0 = 0.5m$, $L = 1m$, time-step size $dt = 0.01\,dx/c$, grid spacing $dx = 1/(N_x - 1)$, $N_x = 100$. Fixed zero boundary condition is used: $E_z(x=0) = E_z(x=L) = 0$, $B_y(x=0) = B_y(x=L) = 0$. Since the waveform has not reach the boundary, the boundary has no effect on the evolution. The results show that two traveling waves emerge from the Gaussian waveform of B_y, propagating in the opposite directions. The propagation speed is correct.

4 Diffusion equation

The stencil used in the explicit, implicit, and Crank–Nicolson implicit method is given in Figure (3).

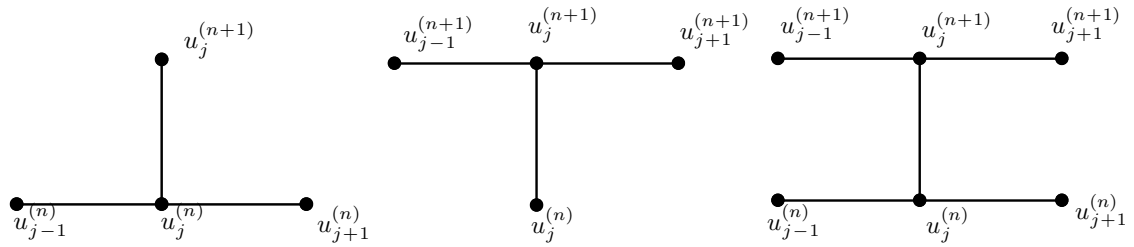

Figure 3. Stencil of an explicit scheme (left) implicit scheme (middle) and the Crank-Nicolson implicit scheme (right) for the diffusion equation $u_t = u_{xx}$.

This part is to be continued.

5 Poisson's equation

Poisson's equation is written

$$\nabla^2 \varphi = S, \tag{22}$$

where S is a known source term. In Cartesian coordinates and in the 2D case, the above equation is written

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial x^2} = S(x, y). \tag{23}$$

Consider solving the above equation in a rectangular domain with $x_a \leqslant x \leqslant x_b$ and $y_a \leqslant y \leqslant y_b$ and with $\varphi = 0$ on the boundary.

5.1 Discretized form using finite differencing

Discretize x as $x_i = x_a + (i+1)\Delta_x$, where $\Delta_x = (x_b - x_a)/(M+1)$ and $i = 0, 1, 2, ..., M-1$. Similarly, discretize y as $y_j = y_a + (j+1)\Delta_y$, where $\Delta_y = (y_b - y_a)/(N+1)$ and $j = 0, 1, 2, ..., N-1$. Then the above equation can be discretized by using the following finite difference:

$$\frac{\varphi_{i+1,j} - 2\varphi_{i,j} + \varphi_{i-1,j}}{\Delta_x^2} + \frac{\varphi_{i,j+1} - 2\varphi_{i,j} + \varphi_{i,j-1}}{\Delta_y^2} = S_{i,j}, \tag{24}$$

where $\varphi_{i,j} = \varphi(x_i, y_j)$ and $S_{i,j} = S(x_i, y_j)$. Equation (24) can be arranged as

$$a\varphi_{i+1,j} + a\varphi_{i-1,j} + c\varphi_{i,j} + b\varphi_{i,j+1} + b\varphi_{i,j-1} = S_{i,j}, \tag{25}$$

where $a = 1/\Delta_x^2$, $b = 1/\Delta_y^2$, and $c = -\left(\frac{2}{\Delta_x^2} + \frac{2}{\Delta_y^2}\right)$. This is a 5-points stencil, as is illustrated in Fig. 4.

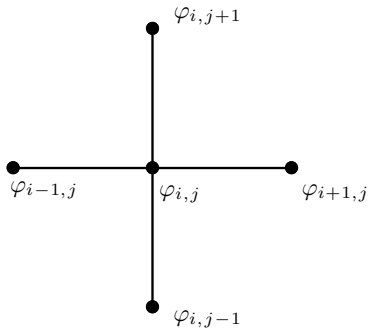

Figure 4. Five-points stencil of the finite differencing scheme in Eq. (25).

In this discritization, the boundary conditions are written as $\varphi_{-1,j} = \varphi_{M,j} = 0$ and $\varphi_{i,-1} = \varphi_{i,N} = 0$.

5.2 Matrix form of the difference scheme

In order to solve the linear equation system (25), we prefer to formulate it in a matrix form. In order to do this, we need to order the 2D discrete unknowns $\varphi_{i,j}$ in a 1D sequence. Two natural ordering schemes are the row-ordering and the column ordering. I choose the row ordering, as is illustrated in Fig. 5.

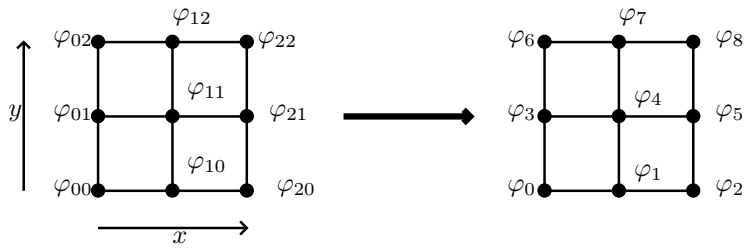

Figure 5. Row-ordering of the 2D discrete unknowns $\varphi_{i,j}$ on a 3×3 mesh.

Using the above ordering, the linear equation system (25) for the special case of a 3×3 mesh is written as the following matrix form:

$$\begin{pmatrix} c & a & & b & & & & & \\ a & c & a & & b & & & & \\ & a & c & & & b & & & \\ b & & & c & a & & b & & \\ & b & & a & c & a & & b & \\ & & b & & a & c & & & b \\ & & & b & & & c & a & \\ & & & & b & & a & c & a \\ & & & & & b & & a & c \end{pmatrix} \begin{pmatrix} \varphi_0 \\ \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \varphi_4 \\ \varphi_5 \\ \varphi_6 \\ \varphi_7 \\ \varphi_8 \end{pmatrix} = \begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \\ S_7 \\ S_8 \end{pmatrix}, \tag{26}$$

where all the blank elements are zeros. This 9×9 matrix is sparse but is not tridiagonal. Each row of the matrix corresponds to one difference equation at a grid point. It is not difficult to generalize the above 9×9 matrix to a general $MN \times MN$ matrix. The general pattern is that those rows that corresponds to inner grid points have the following pattern $(..., b, ..., a, c, a, ..., b, ...)$, where c is on the diagonal location and the distance between b and c is M. For those rows that correspond to boundary grid points, some b and/or a can be absent. Specifically, (1) the left a is absent for all the rows corresponding to the left boundary grids; (2) the right a is absent for all the rows corresponding to the right boundary grids; (3) the left b is absent for all the rows corresponding to bottom boundary grids; (4) the right b is absent for all the rows corresponding to the top boundary grids. The following Fotran code illustrates how to set up the matrix elements for this kind of sparse matrix:

```
dx=1.0/(m+1)
dy=1.0/(n+1)
a_coef= 1./dx**2
b_coef=1./dy**2
c_coef = -2*(a_coef+b_coef)
A=0 !initialize coefficent matrix
do  II=0,m*n-1
     A(II,II)=c_coeff !diagonal elements
```

```
     j = II/m !recover the original index in the y direction of the 2D mesh
     if (j.gt.0) A(II,II-m)=b_coef
     if (j.lt.n-1) A(II,II+m)=b_coeff
     i = II - j*m !recover the original index in the x direction of the 2D mesh
     if (i.gt.0) A(II,II-1)=a_coeff
     if (i.lt.m-1) A(II,II+1)=a_coeff
enddo
```

I use PETSc parallel library[1] to solve the above linear system. In this case, the corresponding code for setting up the matrix in parallel is as follows:

```
dx=1.0/(m+1)
dy=1.0/(n+1)
a_coef= 1./dx**2
b_coef=1./dy**2
c_coef = -2*(a_coef+b_coef)
call MatGetOwnershipRange(A,Istart,Iend,ierr)
do  II=Istart,Iend-1
     call  MatSetValues(A,ione,II,ione,II,c_coef,INSERT_VALUES,ierr) !diagonal
elements

     j = II/m !recover the original index in the y direction of the 2D mesh
     if (j.gt.0) then
        JJ = II - m
        call MatSetValues(A,ione,II,ione,JJ,b_coef,INSERT_VALUES,ierr)
     endif
     if (j.lt.n-1) then
        JJ = II + m
        call MatSetValues(A,ione,II,ione,JJ,b_coef,INSERT_VALUES,ierr)
     endif

    i = II - j*m !recover the original index in the x direction of the 2D mesh
     if (i.gt.0) then
        JJ = II - 1
        call MatSetValues(A,ione,II,ione,JJ,a_coef,INSERT_VALUES,ierr)
     endif
     if (i.lt.m-1) then
        JJ = II + 1
        call MatSetValues(A,ione,II,ione,JJ,a_coef,INSERT_VALUES,ierr)
     endif
enddo
call MatAssemblyBegin(A,MAT_FINAL_ASSEMBLY,ierr)
call MatAssemblyEnd  (A,MAT_FINAL_ASSEMBLY,ierr)
```

All the elements that are not updated by `MatSetValues` in the above code are by default zero.

5.3 Verification of the numerical solution

For the particular source term given by

$$S(x,y) = \sin(\pi x)\sin(\pi y), \tag{27}$$

then

$$\varphi(x,y) = -\frac{1}{2\pi^2}\sin(\pi x)\sin(\pi y), \tag{28}$$

satisfies the equation and the boundary condition $\psi = 0$ at $x_a = 0$, $x_b = 1$, $y_a = 0$, and $y_b = 1$. Therefore the above expression is an analytic solution to the problem. PETSc code for setting the source term (right-hand-side of the linear equation system `Ax=b`) is similar to setting the coefficient matrix. The code is as follows:

```
call VecCreateMPI(PETSC_COMM_WORLD,PETSC_DECIDE,m*n,b,ierr)
call VecSetFromOptions(b,ierr)
call VecDuplicate(b,x,ierr) !We form 1 vector from scratch and then duplicate as
needed.
call VecGetOwnershipRange(b, i_low, i_upp, ierr)
ns=i_upp-i_low
do k=1,ns
     ix(k)=i_low+k-1
     j = ix(k)/m
     i = ix(k) - j*m
     xp=xa+(i+1)*dx
     yp=ya+(j+1)*dy
     val(k)=sin(pi*xp)*sin(pi*yp) !for the source term, anlaytic solution exists
enddo
call VecSetValues(b, ns, ix, val, Insert_values, ierr)
call VecAssemblyBegin(b,ierr)
call VecAssemblyEnd(b,ierr)
```

The last step is to call PETSc KSP solver to solve the linear equation system. The code is as follows:

```
 call KSPCreate(PETSC_COMM_WORLD,ksp,ierr)
 call KSPSetOperators(ksp,A,A,ierr)
 call KSPSetFromOptions(ksp,ierr)
 call KSPSolve(ksp,b,x,ierr)
```

Figure 6 compares the numerical solution with the analytic one, which indicates the two results agree with each other, and thus verifying the correctness of the numerical solution.

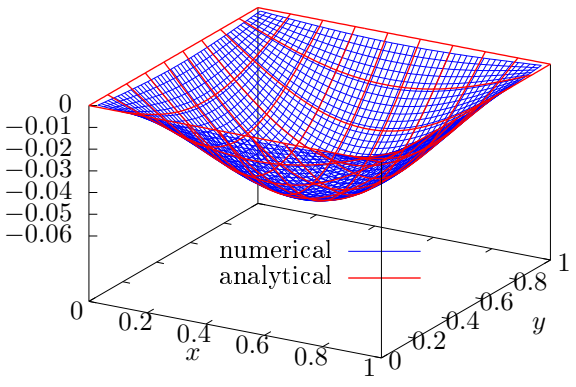

Figure 6. Comparison between the numerical solution and analytic solution of Poisson' equation. Numerical parameters: grid number $M = 50$, $N = 50$. The resulting linear system has 2500 unknowns. PETSc provides a flexible way of choosing different algorithms for solving the linear system via command line options. The command line options used for this case is: `mpiexec -n 3 ./poisson -pc_type bjacobi -sub_pc_type ilu -ksp_type bcgs -ksp_monitor`, which chooses the preconditioner type and Krylov subspace method type. The KSP residual norm is 1.456×10^{-7} after 25 iterations.

6 A simple example of numerical instability

For the following ordinary differential equation:

$$\frac{dy}{dt} = -ay, \tag{29}$$

where a is a positive constant, with the initial condition $y(0) = y_0$, the analytic solution is given by

$$y = y_0 \exp(-at), \tag{30}$$

which is a monotonically decreasing function of t. Let us try to solve this initial value problem numerically. Discritizing time as $t_n = n\Delta t$ with $\Delta t > 0$, we use the following explicit differencing scheme:

$$\frac{y^{(n+1)} - y^{(n)}}{\Delta t} = -ay^{(n)}, \tag{31}$$

i.e.,

$$y^{(n+1)} = (1 - \Delta t\, a)\, y^{(n)}, \tag{32}$$

where $y^{(n)} = y(t_n)$ and $y^{(n+1)} = y(t_{n+1})$. If we choose a large time-step Δt with $\Delta t > 2/a$, then $|1 - \Delta t\, a| > 1$ and the above scheme gives a numerical solution with amplitude increasing with time, instead of decaying. This is totally different from the analytic solution, which indicates the numerical solution is wrong in this case. This kind of wrong numerical solution is called a numerical instability. An example is shown in Fig. 7.

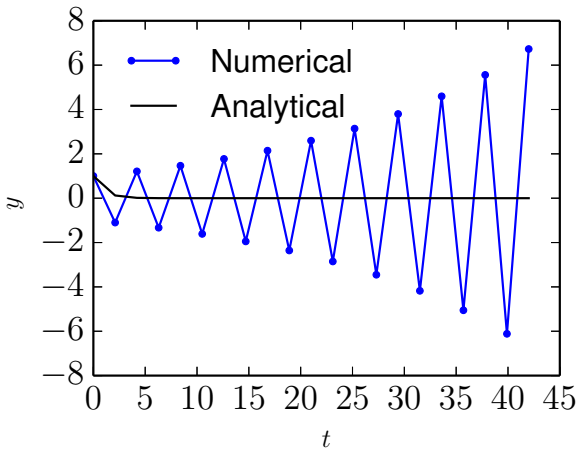

Figure 7. Comparison between the analytic solution (30) (black) and the numerical solution (blue) calculated by the scheme (32) with $\Delta t = 2.1$. Other parameters: $y_0 = 1$, $a = 1$.

Let us consider the following implicit scheme

$$\frac{y^{(n+1)} - y^{(n)}}{\Delta t} = -ay^{(n+1)}, \tag{33}$$

(where the right-hand side is evaluated at the future time time), i.e.,

$$y^{(n+1)} = \frac{y^{(n)}}{1 + a\Delta t}. \tag{34}$$

Note that no matter how large the time step-length Δt is, the above scheme always give a solution which is decreasing with time, i.e., no numerical instability appears. An example is shown in Fig. 8.

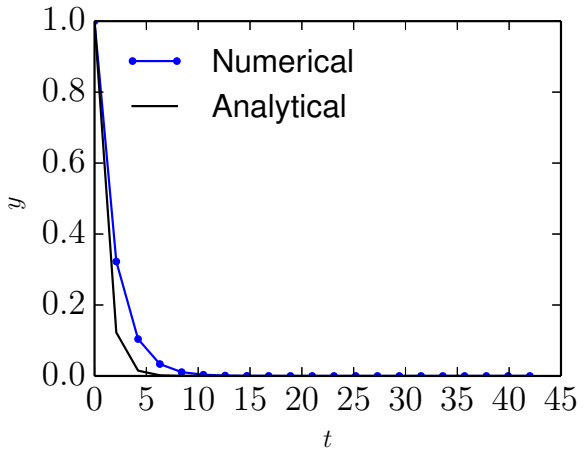

Figure 8. Comparison between the analytic solution (30) (black) and the numerical solution (blue) calculated by the scheme (34) with $\Delta t = 2.1$. Other parameters: $y_0 = 1$, $a = 1$.

If a one-step explicit scheme is unstable, then the corresponding implicit scheme is stable. This is because that an implicit scheme corresponds to a time-reversed version of the corresponding explicit scheme.

7 Finite difference

Taylor expansion of $f(x)$ at $x = x_i$ is written as

$$f(x_i + h) = f(x_i) + h f^{(1)}(x_i) + \frac{h^2}{2} f^{(2)}(x_i) + O(h^3) \tag{35}$$

8 The predictor-corrector method

This method is very similar to and often confused with the Runge-Kutta method. Consider the following differential equation

$$\frac{dy}{dt} = f(y) \tag{36}$$

A natural discretized form that is time-centered would be

$$y_{n+1} = y_n + \frac{\Delta t}{2}[f(y_{n+1}) + f(y_n)]. \tag{37}$$

Unfortunately the presence of y_{n+1} on the right-hand side makes this scheme implicit, and thus a direct solution is possible only for some special cases (e.g. $f(y)$ is a linear function of y). Generally, we need to use iterations to solve the above equation. A convenient initial guess is $y_{n+1} \approx y_n$. If we iterate for only twice, i.e.,

$$y_{n+1}^{(1)} = y_n + \Delta t f(y_n) \tag{38}$$

$$y_{n+1}^{(2)} = y_n + \frac{\Delta t}{2}\big[f\big(y_{n+1}^{(1)}\big) + f(y_n)\big]. \tag{39}$$

Then this is the predictor-corrector method (also called Heun's method). This method consists of a guess for y_{n+1} based on the Euler method (the Prediction) followed by a correction using trapezoidal rule.

If we iterate until convergence, then this is a general implicit scheme.

9 Spectral method——to be revised

Spectral methods refers to methods of using linear combination of global basis functions to approximate a unknown function. Here "global" means that the basis functions extending over the whole spatial domain of interest, i.e., has a support as large as the whole domain of interest (contrast to the finite element method, which use local basis functions). Here we consider Fourier spectral method, which uses trigonometric functions as basis functions. Consider the following two-point boundary value problem:

$$L\psi(x) \equiv \left[-\frac{\partial^2}{\partial x^2} + V(x) \right] \psi(x) = S(x), \tag{40}$$

with the boundary condition $\psi(x = L) = \psi(x = 0)$, where $V(x)$ and $S(x)$ are known functions. Expand $\psi(x)$ in terms of the Fourier basis functions:

$$\psi(x) \approx \sum_{n=0}^{N-1} \hat{\psi}_n \exp\left(n \frac{2\pi i\, x}{L} \right), \tag{41}$$

where $\hat{\psi}_n$ are unknown coefficients. Substitute this expression into the left-hand side of Eq. (40), we obtain

$$\sum_{n=0}^{N-1} \left(\frac{2\pi n}{L} \right)^2 \hat{\psi}_n \exp\left(n \frac{2\pi i\, x}{L} \right) + \sum_{n=0}^{N-1} V(x) \hat{\psi}_n \exp\left(n \frac{2\pi i\, x}{L} \right). \tag{42}$$

Define the residual R as the difference between the above expression and the source term $S(x)$, i.e.,

$$R = \sum_{n=0}^{N-1} \left(\frac{2\pi n}{L} \right)^2 \hat{\psi}_n \exp\left(n \frac{2\pi i\, x}{L} \right) + \sum_{n=0}^{N-1} V(x) \hat{\psi}_n \exp\left(n \frac{2\pi i\, x}{L} \right) - S(x). \tag{43}$$

We want the residual to be as small as possible in the whole domain of interested. We need to define how to measure the smallness of the residual. A general method is to choose some "test functions" and take the inner product of the test functions with the residual over the whole domain. Then the inner product is used to measure the smallness of the residual. Different spectral methods are classified by the different "test functions" chosen for the inner product.

9.1 Pseudo-spectral method

In the Pseudo-spectral method, the test functions are chosen to be $\delta(x - x_m)$, where δ is the Dirac-delta function and x_m with $m = 0, 1, 2, ..., N - 1$ are special spatial points chosen for a set of basis functions. These points are called collocation points and differs for different basis functions used. For Fourier basis functions the collocation points are points with uniform interval given by $x_m = m\,L\,/\,N$ with $m = 0, 1, 2, ..., N - 1$. Performing the inner product $\int_0^L (...)\delta(x - x_m)dx$ on the residual and demand the result to be zero, we obtain

$$\sum_{n=0}^{N-1} \left(\frac{2\pi n}{L} \right)^2 \hat{\psi}_n \exp\left(n \frac{2\pi i\, x_m}{L} \right) + \sum_{n=0}^{N-1} \hat{\psi}_n V(x_m) \exp\left(n \frac{2\pi i\, x_m}{L} \right) - S(x_m) = 0, \tag{44}$$

which can be organized as

$$\sum_{n=0}^{N-1} \hat{\psi}_n \exp\left(n \frac{2\pi i\, x_m}{L} \right) \left[\left(\frac{2\pi n}{L} \right)^2 + V(x_m) \right] = S(x_m). \tag{45}$$

where $m = 0, 1, ...N - 1$. Equation (45) is a linear equation system for the expansion coefficients $\hat{\psi}_n$. Since taking the inner product with a Dirac-delta function $\delta(x - x_m)$ correspond to choosing a particular spatial point x_m, the above equation is actually demanding that the approximate function satisfies the original differential equation exactly on the set of collocation points.

9.2 Galerkin method

Choose the set of test functions as $\exp(-2\pi i m x/L)$ with $m=0,1,...,N-1$. Perform the inner product of the residual with the test functions $\exp(-2\pi i m x/L)$, i.e., $\frac{1}{L}\int_0^L R\exp(-2\pi i m x/L)dx$, and demand the result to be zero, yielding

$$\left(\frac{2\pi m}{L}\right)^2\hat{\psi}_m+\sum_{n=0}^{N-1}V_{mn}\hat{\psi}_n-\frac{1}{L}\int_0^L S(x)\exp\left(-\frac{2\pi i m x}{L}\right)dx=0, \tag{46}$$

where use has been made of

$$\frac{1}{L}\int_0^L\exp\left(\frac{2\pi i\,x}{L}(n-m)\right)dx=\delta_{nm},$$

with δ_{nm} is the Kronicle-delta function, and

$$V_{mn}=\frac{1}{L}\int_0^L V(x)\exp\left(\frac{2\pi i\,x}{L}(n-m)\right)dx. \tag{47}$$

Direct evaluating the integration of the source term as appearing in Eq. (46) involves N operation for each value of m and thus total N^2 operations are needed. The computational efficiency can be improved by first expanding $S(x)$ in terms of the basis function (as what is done for ψ):

$$S(x)\approx\sum_{n=0}^{N-1}\hat{S}_n\exp\left(n\frac{2\pi i\,x}{L}\right).$$

Then the integration of the source term reduces to $\hat{S}_m$. Then equation (46) is written

$$\left(\frac{2\pi m}{L}\right)^2\hat{\psi}_m+\sum_{n=0}^{N-1}V_{mn}\hat{\psi}_n=\hat{S}_m. \tag{48}$$

Since computing $\hat{S}_m$ with $m=0,1,...,N-1$ using FFT involves only $N\log N$ operations, this method is more efficient than directly evaluating the integration. Similar situation apply to the computation of V_{mn}. The matrix V_{mn} depends on m and n through the combination $(n-m)$. Since both m and n are in the range $[0\colon N-1]$, the range of $(n-m)$ is also in $[0\colon N-1]$. Therefore the matrix V_{mn} has N independent matrix elements. Computing each one of these N elements by directly evaluating the integration (47) involves N operations. Therefore, to obtain all the N independent elements, the number of operations is N^2. The same method used to compute the source term can be applied to compute V_{mn}, which reduces the operation number to $N\log N$. Expand $V(x)$ as

$$V(x)\approx\sum_{k=0}^{N-1}\hat{V}_k\exp\left(k\frac{2\pi i\,x}{L}\right)$$

then V_{mn} is written as

$$V_{mn}=\frac{1}{L}\int_0^L\sum_{k=0}^{N-1}\hat{V}_k\exp\left(k\frac{2\pi i\,x}{L}\right)\exp\left(\frac{2\pi i\,x}{L}(n-m)\right)dx=\hat{V}_{m-n}. \tag{49}$$

Therefore Eq. (48) is finally written

$$\left(\frac{2\pi m}{L}\right)^2\hat{\psi}_m+\sum_{n=0}^{N-1}\hat{V}_{m-n}\hat{\psi}_n=\hat{S}_m, \tag{50}$$

which is a linear equation system for $\hat{\psi}_m$ with $m=0,1,...,N-1$.

9.2.1 Computation of $\sum_{n=0}^{N-1}\hat{V}_{m-n}\hat{\psi}_n$ in initial value problems

Equation (50) can be considered as a stead-state equation of the following time-dependent equation

$$\frac{\partial\hat{\psi}_m}{\partial t}=\left(\frac{2\pi m}{L}\right)^2\hat{\psi}_m+\sum_{n=0}^{N-1}\hat{V}_{m-n}\hat{\psi}_n-\hat{S}_m, \tag{51}$$

Note that the term $\sum_{n=0}^{N-1}\hat{V}_{m-n}\hat{\psi}_n$ on the right-hand side of the above equation involves matrix multification, which involves N^2 operations. When solving Eq. (51) as an initial value problem, where $\hat{\psi}_m$ is known at the current time step, there is an efficient way of evaluating $\sum_{n=0}^{N-1}\hat{V}_{m-n}\hat{\psi}_n$ which avoids the computationally expensive matrix multification. Note that the term $\sum_{n=0}^{N-1}\hat{V}_{m-n}\hat{\psi}_n$ is actually the Fourier transform of $V(x)\psi(x)$. Thus an efficient method of computing this term is to first transform $\hat{\psi}_n$ back to real space and doing the multification between $V(x)$ and $\psi(x)$ in real space. Then transform the result back to Fourier space. Since the transform can be performed by FFT, which involves only $N\log N$ operations, this method is more efficient than directly computing the matrix multification $\sum_n \hat{V}_{m-n}\hat{\psi}_n$.

10 Interpolating

ddd

11 von Neuman stability analysis

Ampere's equation is written

$$\nabla\times\delta\mathbf{B}=\mu_0\delta\mathbf{J}_i+\mu_0\delta\mathbf{J}_e \tag{52}$$

Neglecting the ion current, then the above equation is written

$$\nabla\times\delta\mathbf{B}=\mu_0\delta\mathbf{J}_e \tag{53}$$

Using $\delta\mathbf{J}_e=-en_0\delta\mathbf{u}_e$, the above equation is written

$$\nabla\times\delta\mathbf{B}=-\mu_0 en_{e0}\delta\mathbf{u}_e \tag{54}$$

$$\Rightarrow(\nabla\times\delta\mathbf{B})\times\mathbf{B}_0=-\mu_0 en_{e0}\delta\mathbf{u}_e\times\mathbf{B}_0 \tag{55}$$

Using $\delta\mathbf{u}_e\times\mathbf{B}_0=-\delta\mathbf{E}$, the above equation is written

$$\Rightarrow(\nabla\times\delta\mathbf{B})\times\mathbf{B}_0=\mu_0 en_{e0}\delta\mathbf{E} \tag{56}$$

Define $\beta_e=\mu_0 en_{e0}$, then the above equation is written as

$$\delta\mathbf{E}=\frac{1}{\beta_e}(\nabla\times\delta\mathbf{B})\times\mathbf{B}_0 \tag{57}$$

$$\Rightarrow\delta\mathbf{E}^{(n+1)}=\frac{1}{\beta_e}(\nabla\times\delta\mathbf{B}^{(n+1)})\times\mathbf{B}_0 \tag{58}$$

Faraday's law is written as

$$\frac{\delta\mathbf{B}^{(n+1)}-\delta\mathbf{B}^{(n)}}{\Delta t}=-[\alpha\nabla\times\delta\mathbf{E}^{(n+1)}+(1-\alpha)\nabla\times\delta\mathbf{E}^{(n)}] \tag{59}$$

Assume $\mathbf{B}_0$ is uniform and performing Fourier transformation over the space, equations () and () are written

$$\delta\hat{\mathbf{E}}^{(n+1)}=\frac{1}{\beta_e}\big(i\mathbf{k}\times\delta\hat{\mathbf{B}}^{(n+1)}\big)\times\mathbf{B}_0 \tag{60}$$

$$\frac{\delta\hat{\mathbf{B}}^{(n+1)}-\delta\hat{\mathbf{B}}^{(n)}}{\Delta t}=-\big[\theta i\mathbf{k}\times\delta\hat{\mathbf{E}}^{(n+1)}+(1-\theta)i\mathbf{k}\times\delta\hat{\mathbf{E}}^{(n)}\big] \tag{61}$$

Consider the case that $\mathbf{B}_0$ is along the $\mathbf{z}$ direction and $\mathbf{k}=k\hat{\mathbf{z}}$, equation () is written as

$$\delta\hat{\mathbf{E}}^{(n+1)}=\frac{1}{\beta_e}\big(i\,kB_0\delta\hat{\mathbf{B}}^{(n+1)}+i\mathbf{k}\delta\hat{B}_z^{(n+1)}B_0\big) \tag{62}$$

Using this in Eq. (), yielding

$$\frac{\delta\hat{\mathbf{B}}^{(n+1)}-\delta\hat{\mathbf{B}}^{(n)}}{\Delta t}=-\left[-\theta\frac{kB_0}{\beta_e}\mathbf{k}\times\delta\hat{\mathbf{B}}^{(n+1)}+(1-\theta)i\mathbf{k}\times\delta\hat{\mathbf{E}}^{(n)}\right] \tag{63}$$

$$\frac{\delta\hat{\mathbf{B}}^{(n+1)}-\delta\hat{\mathbf{B}}^{(n)}}{\Delta t} = -\left[-\theta\frac{kB_0}{\beta_e}k\delta\hat{B}_x^{(n+1)}\hat{\mathbf{y}} + \theta\frac{kB_0}{\beta_e}k\delta\hat{B}_y^{(n+1)}\hat{\mathbf{x}} + (1-\theta)i\,k\delta\hat{E}_x^{(n)}\hat{\mathbf{y}} - (1-\theta)i\,k\delta\hat{E}_y^{(n)}\hat{\mathbf{x}}\right] \tag{64}$$

Assume $\delta\hat{\mathbf{E}}^{(n)}$ and $\delta\hat{\mathbf{B}}^{(n)}$ take the following form

$$\delta\hat{\mathbf{E}}^{(n)}=\delta\hat{\mathbf{E}}^{(0)}e^{i(-n\omega\Delta t)} \tag{65}$$

$$\delta\hat{\mathbf{B}}^{(n)}=\delta\hat{\mathbf{B}}^{(0)}e^{i(-n\omega\Delta t)} \tag{66}$$

Bibliography

[1] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc Web page. 2018. http://www.mcs.anl.gov/petsc.

[2] Richard Fitzpatrick. *Computational Physics:An introductory course*. Richard Fitzpatrick, 2004.