Fluid Mechanics: Worksheet 2

June 2019

Excercises

The idea of this worksheet is to program forward and backward Euler. By some of the examples, we want to highlight the numerical instability of forward methods.

For the following exercises, we set the following notation. We consider a curve

$$X : [0, \infty) \to \mathbb{R}^n,$$

which is a solution to the ODE equation

$$\begin{cases} \frac{dX(t)}{dt} = F(t, X(t)) & \text{for } t > 0 \\ X(0) = X_0, \end{cases} \tag{1}$$

where $X_0 \in \mathbb{R}^n$ and $F : [0, \infty) \times \mathbb{R}^n \to \mathbb{R}^n$.

1. Implement forward Euler as a function that given the input of initial condition X_0, the step size h, the final simulation time T and the function F, it computes the discrete approximation X_h to X.

   - Create an m file script called Feuler.m
   - To make it a function write in the first line
     function Xh=Feuler(X0,h,T,F)
   - Compute the number of steps necessary by taking the floor function of
     N_iterations=floor(T/h)
   - Initialize Xh:

     Xh=zeros(n,N_iterations)

     where n is the dimension of the problem.
   - Implement the initial condition:
     Xh(:,1)=X0
   - Implement a for loop that computes the approximated value of the rest of Xh:
     for k=2:N_iterations
     Xh(k)=Xh(k-1)+h F((k-1)*h,Xh(k-1))
     end

- For 1-dimensional problems, you can easily plot the solution by using:
  Y=linspace(0,T,N_iterations) plot(Y,Xh)
  For 2-dimensional problems you can use:
  plot(Xh(1,:),Xh(2,:))
  For 3-dimensional problems you can use:
  plot3(Xh(1,:),Xh(2,:),Xh(3,:))

2. Use your implementation to solve the following problems. Do you see any instabilities? Do they dissapear with smaller h?

   Make sure you take into consideration the dimensionality of each problem.

   - n=1, h=0.1, T=10, $F(t,x) = -100 * x$.
   - n=1, h=0.01, T=10, $F(t,x) = -100 * x$.
   - n=1, h=0.1, T=10, $F(t,x) = cos(t) * x$.
   - (Lotka-Volterra) n=2, h=0.1, T=10

     $$F(t,x,y) = \begin{pmatrix} ax - bxy - cx^2 \\ dy + exy - jy^2 \end{pmatrix}$$

     with parameters
     - a=2, b=2, c=0, d=-1, e=1, j=0
     - a=1, b=8, c=1, d=-1, e=4, j=0

     To handle functions in Matlab or Octave. You can input in the console

     $$F = @(t,x,y) \ \ [2 * x - 2 * x * y, -y + 4 * x * y]$$

     This function you can then feed into your implementation of forward or backward matlab. I have added the time variable, even if we are not using it.

3. For the 1-dimensional examples, perform an error analysis. Solve explicitly the ODE and compare the value of the exact solution with the approximated one. For example, for F(t,x)=-100*x, we have

   $$X(t) = X_0 e^{-100t}.$$

   We can feed this into a vector, by doing the following steps

   - Initialize
     X=zeros(N_iterations);
   - Load information
     for i=1:N_iterations
       X(i)=X0*exp(-100* (i-1)*h);
     end

- Compare with the approximated solution.
  max(abs(Xh-X))

How does the error behave with h? Can you plot different values?

4. Implement backward Euler in the same way, for time independent problems. The only difference is in the loop that computes the approximated values. For 1-dimensional problems, use the function

   Xh(k)=vpasolve(x+h*F(x)==Xh(k-1),x,Xh(k-1))

   For 2-dimensional problems use the function

   Xh(:,k)=vpasolve([x+h*F1(x)==Xh(1:k-1) y+h*F2(y)==Xh(2:k-1)] , [x y] , [Xh(1,k-1) Xh(2,k-1)])

   where F1 and F2 are respectively the first and second component of the function $F : \mathbb{R}^2 \to \mathbb{R}^2$.

   Try the same examples as for forward Euler. Do you see instabilities?

5. Interacting particle simulations:

   We consider a hundred interacting particles in 2-dimensional space, which are represented by the vector $(Z_1, Z_2, ..., Z_{100})$, where each $Z_i \in \mathbb{R}^2$. We consider the coupled set of equations

$$\begin{cases} \dot{Z}_1 &= \frac{1}{100}\sum_{j\neq 1}\frac{Z_1-Z_j}{|Z_1-Z_j|^2} - Z_1 \\ \dot{Z}_2 &= \frac{1}{100}\sum_{j\neq 2}\frac{Z_2-Z_j}{|Z_2-Z_j|^2} - Z_2 \\ &\vdots \\ \dot{Z}_{100} &= \frac{1}{100}\sum_{j\neq 100}\frac{Z_{100}-Z_j}{|Z_{100}-Z_j|^2} - Z_{100}. \end{cases}$$

   Write down a forward Euler code to run this simulation.

   - Use random initial conditions in the cube $[-2, 2] \times [-2, 2]$:
     Z0=4.*(rand(2,100)-0.5)
   - You don't have to save all the iterates! You can plot the solution at every step by the command
     scatter(Z(1,:),Z(2,:))
     You can use the pause command if the simulation is too fast.
     pause(0.1)

   Do you see any patterns, when you let time evolve?