**Numerical Methods II**, Courant Institute, Spring 2024
http://www.math.nyu.edu/faculty/goodman/teaching/NumericalMethodsII2024.html
Jonathan Goodman, February, 2024

# Assignment 3, Solving ODE,

This is a more traditional assignment with some theory exercises and a computation.

1. Runge Kutta methods are sometimes applied to linear ODEs

$$\dot{x} = Ax \ .$$

Show that there is a $p$ stage method that uses $p$ matrix/vector multiplications per time step and achieves order $p$ accuracy. One way to do this is a matrix version of what is called *Horner's rule* for evaluating a polynomial $u(x)$:

$$\begin{aligned} u(x) &= a_p x^p + a_{p-1} x^{p-1} + \cdots + a_1 x + a_0 \\ &= \left( a_p x^{p-1} + a_{p-1} x^{p-1} + \cdots + a_1 \right) x + a_0 \\ &= \left( \left( a_p x^{p-2} + \cdots + a_2 \right) x + a_1 \right) x + a_0 \\ &\vdots \\ &= (\cdots ( a_p x + a_{p-1} ) x + a_{p-2} ) x + \cdots + a_1 ) x + a_0 \ . \end{aligned}$$

The Horner's rule algorithm for evaluating a polynomial is

$$\begin{aligned} y_p &= a_p x + a_{p-1} \\ y_{p-1} &= y_p x + a_{p-2} \\ &\vdots \\ y_1 &= y_2 x + a_1 \\ u(x) &= y_1 x + a_0 \ . \end{aligned}$$

You can apply a matrix version of this to the Taylor approximation of the matrix exponential

$$e^{\Delta t A} = I + \Delta t A + \cdots + \frac{1}{p!} \Delta t^p A^p + O(\Delta t^{p+1}) \ ,$$

applied to a vector $x$.
**Remark**. In particular there is a five stage fifth order method for linear problems while there is no five stage fifth order method for general nonlinear problems.
**Remark**. This is also *low storage* because you do not have to store the individual stage evaluations $k_j$.

2. The *Verlet*[1] method is a special leap-frog type centering idea that applies to second order ODEs of the form

$$\ddot{x} = f(x) \ .$$

One version of Verlet starts write it in first order form

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= f(x) \ . \end{aligned}$$

We compute $X_n \approx x(t_n)$ and $V_{n+\frac{1}{2}} \approx v(t_{n+\frac{1}{2}})$. This allows us to create centered approximations to $\dot{x}$ and $\dot{v}$

$$\begin{aligned} \frac{V_{n+\frac{1}{2}} - V_{n-\frac{1}{2}}}{\Delta t} &= f(X_n) \\ \frac{X_{n+1} - X_n}{\Delta t} &= V_{n+\frac{1}{2}} \ . \end{aligned}$$

Show that this method is formally second order accurate (the residual is order $\Delta t^2$).

3. The computing exercise involves a non-linear *lattice wave equation*[2]

$$\ddot{U}_{jk} = U_{j,k+1} + U_{j,k-1} + U_{j+1,k} + U_{j-1,k} - 4U_{jk} - \beta U_{jk}^3 \ . \tag{1}$$

Use the "boundary condition" that $U$ values on the boundary (with $j$ or $k$ equal to 0 or $n+1$) equal to zero. The problem is linear if $\beta = 0$. Consider the discrete Dirichlet sum for the linear discrete Laplace operator with a quartic (fourth order) term added

$$\Phi(U) = \frac{1}{2} \sum \left(U_{j+1,k} - U_{jk}\right)^2 + \left(U_{j,k+1} - U_{jk}\right)^2 + \frac{\beta}{4} \sum U_{jk}^4 \ .$$

Show that the equation (1) has the abstract form

$$\ddot{U} = -\nabla \Phi(U) \ . \tag{2}$$

Consider the *Hamiltonian*

$$H(U, \dot{U}) = \Phi(U) + \frac{1}{2} \sum \dot{U}_{jk}^2 \ . \tag{3}$$

Show that if $U$ satisfies the dynamical equation (1), then

$$\frac{d}{dt} H(U(t), \dot{U}(t)) = 0 \ . \tag{4}$$

[1] This is a French name so you don't pronounce the $t$. In the English approximate pronunciation, the first syllable is "ver", as in "version". The second syllable is like "lay".

[2] Lattice means grid in this context. The atoms in a crystal are arranged in a regular pattern called a *lattice*. You can look on the web for images of different kinds of lattice. This one is a 2D square lattice. A *triangular* lattice is a regular array of equilateral triangles. A simple 3D analogue of 2D square lattice would be a *cubic* lattice, but there are two kinds.

*Hint.* This property, the conservation of the Hamiltonian on trajectories, is a general property of dynamical systems of the form (2). It does not depend on the specific functional form of $\Phi$. This fact would be a part of an upper level physics class on dynamics, where $\Phi$ would be called *potential energy* (or just *potential*) and the second term of (3) would be called *kinetic* energy.

**Programming and computing exercise.**

The goal of this exercise is to create a movie solutions of the lattice wave equation (1). The process should illustrate some steps developing and validating computational software. First is checking that the code is correct in a mathematical sense. Here, we do this by testing on a model problem and doing a convergence analysis to verify the order of accuracy. Second, we check whether beautiful lattice wave movies are correct and how they can be wrong. The code should be *modular* so that it is easy to change problems and solution algorithms. That means that there can be several functions, possibly in separate code files, to take a time step, to compute a solution trajectory, to implement the ODE (evaluate the $f$ in $\dot{x} = f(x)$), to do a convergence analysis, to extract data for the movie, etc.

**ODE solver.** Write an ODE solver that uses either forward Euler or the four stage Runge Kutta method to compute the solution up to time $t$. The Runge Kutta time step should be a separate function (or "method" or "subroutine" or "procedure" depending on your programming language) that takes as input $x$, $\Delta t$, and the function $f$ and produces the result of one time step, which would be called $X_{n+1}$ if $x$ were $X_n$. The ODE solver should compute a trajectory from $x_0 = x(t)$ up to $x(T)$ where $T$ and $\Delta t$ are inputs (also $f$). For the convergence study, it only needs $x(T)$, but for the movie it needs to return an array of *frames* which are the solution at equally spaced times that can be visualized to make frames of a movie. You can "record" a frame (write it into the output array of the ODE solver) every $m$ time steps or with time intervals $T_{\rm fr}$.

**Code validation.** Try your solver on the two-component ODE[3]

$$\begin{aligned} \dot{x} &= -(x^2 + y^2)\, y \\ \dot{y} &= \phantom{-}(x^2 + y^2)\, x \\ x(0) &= 1 \\ y(0) &= 0 \\ x(t) &= \cos(t) \\ y(t) &= \sin(t)\ . \end{aligned}$$

[3]It might seem simpler to use the linear system with the same solution, but it is important to "exercise" the code fully. Exercise 1 shows that it is possible for a method to be fifth order accurate on linear problems but only fourth order for non-linear problems.

Take $T = 2\pi$ and check that forward Euler produces a first order accurate approximation while the fourth order RK produces fourth order accurate solutions. You may find that verifying the fourth order method is a challenge because of roundoff in the ODE solver and possibly because of off-by-one errors in the number of time steps (also caused by roundoff).

**Code play.** Still using the two component ODE, make a plot of the solution curve in 2D up to a longer time such as $20\pi$ or $60\pi$. You may find that the numerical solution blows up if $\Delta t$ is too large. Choose a $\Delta t$ so that the first order "solution" deviates significantly from the exact solution (maybe, doubles in size, or grows by 20%, or try both) and observe the behavior of the fourth order computation with the same $\Delta t$.

**Linear movie.** Apply your code to the lattice wave equation (1) with $\beta = 0$ (the linear case). Choose initial data with $U_{jk}(0) = 0$ except that $U_{\frac{n}{2},\frac{n}{2}}(0) = 1$. That is, take an initial lattice with the only disturbance being one value in the center of the lattice. The movie will show a circular wave expanding outward from the center with speed 1 (one lattice length per unit of time). This is the *light cone*. There is almost no activity outside the light cone but lots inside. You should see lots of small amplitude waves and some very large disturbances that move outward more slowly than the circular light cone wave. This is a *caustic*. You may need to use a large lattice to see these structures clearly. Run the simulation up to two or three times the time it takes the light cone to reach the boundary. After that, the solution is too complicated to be described in a simple way. Experiment with the first and fourth order methods. Note how the light cone and caustic structures are computed inaccurately unless the time step is small (fourth order) or extremely small (forward Euler). Try to describe the different *numerical artifacts* produced by the different time stepping methods. One method "damps" waves so that you see structures with small $\Delta t$ that you don't see with larger $\Delta t$. The other method makes structures bigger than they should be.

**Non-linear movie.** For this, use only the fourth order method. You should have seen by now that it produces accurate solutions much faster than forward Euler. Explore the effect of taking $\beta > 0$. How large does $\beta$ have to be to change the solution noticeably? What happens to the light cone and to the caustic structures inside?

**Extra credit.** Solve the lattice wave equation using the Verlet method. Describe the time needed and artifacts produced. Make a plot of the value of the Hamiltonian. This is constant for the exact solution but not for the numerical solutions. See how different solvers change the Hamiltonian in different ways.