Numerical relativity — Exercise sheet # 1

Boris Daszuta
boris.daszuta@uni-jena.de

21.04.2021

1 Numerical exercises

We investigate numerical solutions based on finite difference schemes of two important classes of (initial) boundary value problems in one spatial dimension.

Exercise 1.1: Poisson equation

The Poisson equation provides a prototypical example of an *elliptic* problem.
Suppose $u : [a, b] \to \mathbb{R}$ satisfies $-u''(x) = f(x)$ subject to the Dirichlet boundary conditions (BC) $u_a := u(a)$ and $u_b := u(b)$.

- For concreteness particularize to the domain $[0, 1]$ and set BC as $u_a = u_b = 0$. The source term we choose as $f(x) = -x(x+3)\exp(x)$.
- The exact solution is $u^{(e)}(x) = x(x-1)\exp(x)$ which we use for verification.

The domain may be discretized by subdividing $[0, 1]$ into $N+1$ equidistant points $x_i := i\Delta x$ where $i = 0, \ldots, N$ and $\Delta x := 1/N$. Thus, our goal is to construct approximate values of the solution $u_i := u(x_i)$.

- At the boundary nodes $i = 0$ and $i = N$ we must impose BC such that $u_0 = 0 = u_N$.
- On the interior nodes $i = 1, \ldots, N-1$ we can approximate the second derivative operator as:

$$u''(x)\big|_{x=x_i} \simeq \mathrm{D}_x^2[u]_i = \frac{1}{(\Delta x)^2}(u_{i-1} - 2u_i + u_{i+1}). \tag{1}$$

- A linear system representing an approximation to our original problem is thus (note especially how the BC are handled here and what equations are imposed component-wise):

$$\underbrace{\begin{bmatrix} 1 & 0 & \cdots & & & \\ 1 & -2 & 1 & 0 & & \cdots \\ 0 & \ddots & \ddots & \ddots & 0 & \cdots \\ \vdots & & 0 & 1 & -2 & 1 \\ & & & & 0 & 1 \end{bmatrix}}_{=:A} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \\ u_N \end{bmatrix} = -(\Delta x)^2 \begin{bmatrix} 0 \\ f_1 \\ \vdots \\ f_{N-1} \\ 0 \end{bmatrix}. \tag{2}$$

- The PYTHON3 script "PoissonD_1d.py" demonstrates a possible implementation.

1. Derive the approximation of Eq.(1). Hint: use Taylor series. What is the formal order of accuracy?

2. Extend the example script to investigate the accuracy of the solution as N is varied. Is what you observe compatible with Eq.(1)?
   Consistency checks and convergence testing are *crucial* in verifying a method.

3. The example script uses direct matrix inversion. This is more expensive than need be (recall naive Gauss elimination requires $\mathcal{O}(N^3)$ operations). The matrix A only features non-trivial entries on the main three diagonals and in principle the linear system can be more cheaply solved in $\mathcal{O}(N)$ operations. Modify the script to make use of a tridiagonal solver (either write your own or see "scipy.linalg.solve_banded".)

4. How would you approach verifying that your solver is working correctly without knowledge of $u^{(e)}$?

Exercise 1.2: Wave equation

The wave equation provides a prototypical example of a *hyperbolic* problem.
Suppose $\phi : (0, \infty) \times \mathbb{R} \to \mathbb{R}$ satisfies $\partial_t^2[\phi(t, x)] = \partial_x^2[\phi(t, x)]$ and $\phi(t, x) = \phi(t, x+2n\pi)$ for all integer n. Our problem is spatially periodic and second order in space and time.

- For convenience introduce a new variable $\pi(t, x) := \partial_t[\phi(t, x)]$. Thus our problem may be thought of as:

$$\partial_t \begin{bmatrix} \phi \\ \pi \end{bmatrix} = \begin{bmatrix} \pi \\ \partial_x^2[\phi] \end{bmatrix}. \tag{3}$$

- The system of Eq.(3) is closed by supplying initial data $\phi_0 := \phi|_{t=0}$ and $\pi_0 := \pi|_{t=0}$ compatible with the periodic BC.

- With the stipulated conditions the wave equation is *well-posed* in the sense that a unique solution exists that depends continuously on the initial data.

To discretize the problem to a form we can solve numerically we initially focus on the spatial part and consider the fundamental domain $\Omega := [0,\, 2\pi)$.

- For the spatial grid we can take $x_i := i\Delta x$ where $i = 0,\, 1,\, \ldots,\, N$ and $\Delta x := 2\pi/N$. In order to make use of Eq.(1) at the edges of the domain notice that $x_{i\pm J} = x_{N\pm J}$. Thus we can form a vector of values extended by so-called "ghost nodes":

$$\begin{bmatrix} \cdots & f(x_{N-1}) & \underbrace{|\, f(x_0) \quad \cdots \quad f(x_N)\,|}_{\text{physical}} & f(x_1) & \cdots \end{bmatrix}^t, \tag{4}$$

  which allows for computation of the spatial derivative when it is required in the vicinity of the boundary.

- Equation (3) may now be written in semi-discrete form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\phi_i \\ \pi_i\end{bmatrix} = \begin{bmatrix}\pi_i \\ \mathrm{D}_x^2[\phi]_i\end{bmatrix}; \tag{5}$$

  which is a coupled, linear ODE system of $2(N+1)$ variables. This can be integrated forward in time from some initial point. The overall approach is known as the *method of lines.*

- Runge-Kutta (RK) methods are robust schemes for integrating systems such as that of Eq.(5). Suppose:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = F(t,\, u), \qquad\qquad u(0) = u_0;$$

  where $u \in \mathbb{R}^N$ and $F : \mathbb{R} \times \mathbb{R}^N \to \mathbb{R}^N$. Recall that an RK scheme can be used to propagate a solution by Δt i.e. $u_n \to u_{n+1}$ and is typically specified through:

$$\begin{aligned} v_i &= u_n + \Delta t \sum_{j=1}^{i-1} a_{ij} F(t_n + c_j \Delta t,\, v_j), \quad (1 \leq i \leq m); \\ u_{n+1} &= u_n + \Delta t \sum_{j=1}^{m} b_j F(t_n + c_j \Delta t,\, v_j); \end{aligned}$$

  where $u_n := u(t_n)$ and $\Delta t := t_{n+1} - t_n$. The v_i are the so-called *stages* of the method. The Butcher coefficients a_{ij}, b_j and c_j are commonly arranged into a table:

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & & \ddots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array}$$

Two common schemes are respectively the midpoint (second-order) and RK4 methods:

$$
\begin{array}{c|cc}
0 & & \\
\frac{1}{2} & \frac{1}{2} & \\
\hline
 & 0 & 1
\end{array}
\qquad\qquad
\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & \\
1 & 0 & 0 & 1 & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}
\tag{6}
$$

- The PYTHON3 script "waveP_1p1d.py" provides some relevant functionality.

1. Define the functions:

$$
s_m(x) := \sin(mx), \qquad f(x) := \sqrt{x}, \qquad g(x) := \exp(-\sin(x)); \tag{7}
$$

   Perform a convergence test based on Eq.(1) and Eq.(4). Explain what you observe.

2. Complete the provided script to furnish numerical solutions to the wave equation. Implement the RK4 method of Eq.(6).

3. The solution (ϕ, π) at a given point $P := (t_j, x_i)$ depends on the information in its *domain of dependence* (DOD). For the wave equation this can be imagined as points contained in the past-cone of P. A numerical method can be convergent only if its numerical DOD contains the true DOD of the underlying PDE. In the present case this can be characterised by the so-called Courant-Friedrich-Lewy (CFL) condition which takes the form $\Delta t \leq \alpha \Delta x$. Typically $0 < \alpha \leq 1$ though it can depend also on the chosen time-integrator. CFL is a necessary but not sufficient condition for convergence. Experiment with varying α and comment on what you observe.

4. Verify based on an analytical solution or otherwise that convergence properties are compatible with the anticipated orders of the approximations made.