Prof. Fabien Evrard & Fabian Dettenrieder
AE 410/CSE 461 – Computational Aerodynamics
Spring 2024

Homework #3

Due date: February 29, 2024

Problem 1 · Develop a general purpose solver for systems of equations of the form

$$\frac{\mathrm{d}\mathbf{U}(t)}{\mathrm{d}t} = \mathbf{F}\left(t, \mathbf{U}(t)\right)$$

where $\mathbf{U}(t)$ is a vector of functions of time, and $\mathbf{F}$ is an operator that depends on the governing equations under consideration. Your code should take the following inputs:

- The ability to choose between periodic or non-periodic domains.
- The ability to choose the finite-difference spatial derivative operator.
- The ability to choose the governing equation.
- The length L of the physical domain, $x \in [0, L]$.
- The length T of the temporal domain, $t \in [0, T]$.
- The number N_x of discrete grid points $x_n, n \in \{0, 1, \ldots, N_x - 1\}$.
  - ↪ If the domain is periodic, then $\Delta x = L/N_x$ (end-point $x = L$ excluded).
  - ↪ If the domain is not periodic, then $\Delta x = L/(N_x - 1)$ (end-point $x = L$ included).
- The number N_t of discrete time levels $t_m, m \in \{0, 1, \ldots, N_t - 1\}$ (used only for reporting the solution!).
  - ↪ The time levels are separated by $\Delta t = T/(N_t - 1)$ (end-point $t = T$ included).
- The initial condition $\mathbf{U}(t = 0)$.

It should return:

- The set of discrete time levels $t_m, m \in \{0, 1, \ldots, N_t - 1\}$.
- The set of solutions $\mathbf{U}(t_m), m \in \{0, 1, \ldots, N_t - 1\}$.

You may use the following Python code, or develop your own:

```python
import numpy as np
from scipy.linalg import circulant
from scipy.integrate import solve_ivp

# Operator from HW1
def D_operator_periodic(N,L,R,a):
  first_row = np.zeros(N); first_row[0:L+R+1] = a; first_row = np.roll(first_row,-L)
  return np.array(circulant(first_row)).transpose()

```

```python
# Functor for the 1D advection equation
def LinearAdv1D(t,U,D):
  # Initialize velocity
  a = 1
  # Return F(t,U)
  return (-a*D)@U

# General integrator function
def Integrator(periodic,operator,problem,L,T,Nx,Nt,U0):
  ######################## Inputs of the function "Integrator" ##########################
  #                                                                                     #
  # periodic : boolean flag to select periodicity (options: True of False)              #
  # operator : string to select the spatial derivative operator                         #
  #  problem : string to select the governing equations                                 #
  #        L : length of the physical domain, x runs from 0 to L                        #
  #        T : length of the temporal domain, t runs from 0 to T                        #
  #       Nx : number of points to use in x                                             #
  #       Nt : number of points to use in t (for reporting the solutions)               #
  #       U0 : initial condition                                                        #
  #                                                                                     #
  ######################## Outputs of the function "Integrator" #########################
  #                                                                                     #
  #        t : the discrete time levels (in a vector of size Nt)                        #
  #        U : the solutions (in a matrix of size Nt x Nx)                              #
  #                                                                                     #
  #######################################################################################

  # Initialize spatial domain
  x  = np.linspace(0, L, Nx, endpoint=(not periodic))
  dx = x[1] - x[0]

  # Initialize temporal domain
  t  = np.linspace(0, T, Nt, endpoint=True)

  # Construct spatial matrix operator
  match (operator,periodic):
    case ('ForwardOrder1FirstDeriv',True):   # Periodic 1st-order forward differences
      D = D_operator_periodic(Nx,0,1,[-1/dx,1/dx])
    case ('BackwardOrder1FirstDeriv',True):  # Periodic 1st-order backward differences
      D = D_operator_periodic(Nx,1,0,[-1/dx,1/dx])
    case _:
      raise Exception("The %s operator '%s' is not yet implement!" % ('periodic' if periodic
          else 'non-periodic', operator))

  # Solve and return solutions!
  match problem:
    case 'LinearAdv1D':
      # Solve initial value problem; see documentation at:
      # https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html
      sol = solve_ivp(LinearAdv1D, [0, T], U0, args=(D,), t_eval=t, rtol=1.0e-6, atol=1.0e-6)
      # Transpose solution vector so that U has the format (Nt x Nx)
      U = sol.y.transpose()
      # Return outputs
      return t, U
    case _:
      raise Exception("The case '%s' is not yet implement!" % problem)
```

Problem 2 · Using the code of Problem 1, solve the problem that we have mainly considered in class so far, i.e., the periodic semi-discrete 1D advection with the wavespeed $a = 1$, domain length $L = 1$, final time $T = 10$, $N_x = 50$ discrete points, and the initial condition

$$u_0(x) = \exp\left(\frac{-\left(x - \frac{L}{2}\right)^2}{2\sigma^2}\right) \quad , \quad \sigma = \frac{3}{40} \ .$$

For each scheme listed below:

(1) Plot the eigenvalue spectrum of the matrix $\mathbf{A} = -a\mathbf{D}$.

(2) Compare your numerical solution to the exact expected solution at the times $t_m, m \in \{1, 2, \dots, 10\}$.

(3) Name and explain the different phenomena you observe based on the previously computed eigenvalue spectra and the known properties of the scheme.

The list of schemes:

- First-order upwind:
$$\left.\frac{\partial u}{\partial x}\right|_i = \frac{u_i - u_{i-1}}{\Delta x}$$
- Second-order central:
$$\left.\frac{\partial u}{\partial x}\right|_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x}$$
- Third-order upwind:
$$\left.\frac{\partial u}{\partial x}\right|_i = \frac{2u_{i+1} + 3u_i - 6u_{i-1} + u_{i-2}}{6\Delta x}$$
- Fourth-order central:
$$\left.\frac{\partial u}{\partial x}\right|_i = \frac{-u_{i+2} + 8u_{i+1} - 8u_{i-1} + u_{i-2}}{12\Delta x}$$
- Sixth-order Padé (**this scheme only needs to be studied by students taking AE 410/CSE 461 for four credit hours**):

$$\left.\frac{\partial u}{\partial x}\right|_i = \frac{u_{i+2} + 28u_{i+1} - 28u_{i-1} - u_{i-2}}{36\Delta x} - \frac{1}{3}\left(\left.\frac{\partial u}{\partial x}\right|_{i-1} + \left.\frac{\partial u}{\partial x}\right|_{i+1}\right)$$

Problem 3 · For each of the schemes you have used in Problem 2:

(1) Derive the analytical expression of the modified wavenumber $\kappa^\star$ using Fourier error analysis.

(2) Plot the real and imaginary parts of $\kappa^\star \Delta x$ as a function of $\kappa\Delta x$, for $\kappa\Delta x \in [0, \pi]$.

(3) On the same plots as in (2), plot the discrete values of $\kappa^\star_n, n \in \{0, \dots, N_x - 1\}$, obtained from the eigenvalues of the matrix $\mathbf{A}$.

Submission guidelines · Instructions on how to prepare and submit your report are available on the course's Canvas page at `https://canvas.illinois.edu/courses/43781/assignments/syllabus`