Advanced Programming in Engineering
Other Differential Equations (ODE 3)

Problem

Many problems in engineering can be expressed and solved in the form of differential equations. The preceding exercises were based on ordinary differential equations (ODEs) with given initial values. In this exercise, the repertoire is extended to ordinary differential equations with boundary conditions, partial differential equations and eigenvalue problems. Solving these equations requires different techniques from the ones used to solve ODEs (or smart invocation of `ode45` by other routines provided in matlab).

Flow between cylinders

Consider a liquid confined between two concentric cylinders. The inner cylinder has a radius R_{in}, the outer cylinder has a radius R_{out}, and both are parallel to the z-axis. The newtonian fluid has a viscosity η. A pressure gradient along the z-direction causes the fluid to flow in the positive z direction. In the central section of the pipe, well away from inlet and outlet, the steady flow is rotational symmetric and directed along the z axis. The radial dependence of this velocity profile, $v_z(r)$, can be solved from the ordinary differential equation

$$\frac{1}{r}\frac{d}{dr}\left(r\frac{dv_z}{dr}\right) = \frac{1}{\eta}\frac{\Delta p}{L}, \tag{1}$$

with r the distance to the central axis of the pipe and Δp the pressure difference (negative, the pressure decreases between inlet and outlet) over the pipe of length L. The fluid obeys stick (a.k.a. non-slip) boundary conditions at the walls,

$$v_z(R_{\text{in}}) = v_z(R_{\text{out}}) = 0. \tag{2}$$

Our objective is to solve this *boundary value problem*.

To discretize the differential equation, we divide the distance between the cylinders into N intervals of equal size, $\Delta r = (R_{\text{out}} - R_{\text{in}})/N$. The velocity profile $v_z(r)$ will be represented by the array[1] `vz`, with `vz[i]` denoting the velocity at $r = i * \Delta r + R_{\text{in}}$ for integer indices from 0 to N. Expanding the first and second derivatives by symmetrised finite difference expressions, the differential equation takes the form

$$\frac{v_z(r+\Delta r) - 2v_z(r) + v_z(r-\Delta r)}{(\Delta r)^2} + \frac{1}{r}\frac{v_z(r+\Delta r) - v_z(r-\Delta r)}{2\Delta r} = \frac{1}{\eta}\frac{\Delta p}{L}, \tag{3}$$

which is readily rewritten into an expression relating the velocity `vz[i]` at `r` $= $ `i`$*\Delta r + R_{\text{in}}$ to the two neighbouring velocities,

```
(1+dr/(2*r))*vz[i+1] - 2*vz[i] + (1-dr/(2*r))*vz[i-1] = ...
                                                (dp/L)*(dr^2)/eta.
```

[1] For clarity we will use the following notational convention:

- *Italics* for analytical expressions and `typewriter` for (quasi-)program code.
- Numerical quantities always have a value that is equivalent, either exactly or approximately, to that of the matching analytical quantity, e.g. `dr` $= \Delta r$ and `vz[i]` $\approx v_z($`i`$\Delta r + R_{\text{in}})$. Note that in some languages arrays start at 1, in which case `vz[i + 1]` $\approx v_z($`i`$\Delta r + R_{\text{in}})$.
- Functions have their argument between round brackets (...), arrays have their (integer) index between square brackets `[...]`.

The resulting set of $(N-1)$ coupled linear equations, one for every `i` $\in \{1, 2, \ldots, N-1\}$, can be combined with the two boundary conditions, `vz[0] = vz[N] = 0`, into a single matrix-vector expression of the form[2]

$$\mathbf{M}\mathbf{v} = \mathbf{c}, \tag{4}$$

where $\mathbf{M}$ is a tri-diagonal $(N-1)\times(N-1)$ matrix, $\mathbf{v}$ the column vector of $(N-1)$ velocities `vz[i]`, and $\mathbf{c}$ a column vector whose $(N-1)$ elements are all equal to $(\Delta p/L)(\Delta r^2/\eta)$. Once the matrix elements have been calculated, the velocity profile is readily solved from

$$\mathbf{v} = \mathbf{M}^{-1}\mathbf{c}. \tag{5}$$

This calculation can be performed faster, avoiding the inversion of $\mathbf{M}$, by using

```
v = M \ c.
```

This `mldivide` routine builds on a technique called matrix decomposition.

Questions

1 Determine the elements of $\mathbf{M}$ and calculate the velocity profile in a pipe.
The relevant numerical parameters (Δp, R_{in}, etc) may be chosen freely.

2 Calulate the velocity profile using the matlab boundary value solver for ordinary differential equations, `bvp4c`.
Have a close look at the data in the structure returned by `bvp4c` to notice that it does something to your data – helpful to the calculation, but naughty if you are not expecting this – at small N (below $\sim$10) that it does not do for large N.

3 Compare both above numerical answers with the analytic solution

$$v(r) = -\frac{1}{4\eta}\frac{\Delta p}{L}\left[\frac{\ln(r/R_{\text{in}})}{\ln(R_{\text{out}}/R_{\text{in}})}\left(R_{\text{out}}^2 - R_{\text{in}}^2\right) - \left(r^2 - R_{\text{in}}^2\right)\right], \tag{6}$$

by plotting all three curves in the same figure.

4 Calculate the average values of the absolute differences between the numerical solutions and the analytic solution, `dv[i]` = `abs(v[i]` $-$ v`(r[i]))`, and determine the convergence rates of both numerical solutions with increasing number of intervals, N.

Heat diffusion

The gradual spreading of an ink droplet dropped in a bath of water, and the diffusion of heat from hot to cold, are examples of a *partial differential equation*. Consider a one-dimensional thermally isolated bar of length L, directed along the x axis, in thermal equilibrium at a uniform temperature T_0. At time $t = 0$, the central quarter of the bar is briefly warmed up (or cooled down), creating the smooth temperature profile

$$T(x,0) = \begin{cases} T_0 & \text{for} \quad |x - L/2| > l = L/8 \\ T_0 + \delta T\left\{\cos\left[\dfrac{\pi}{l}\left(x - \dfrac{L}{2}\right)\right] + 1\right\} & \text{for} \quad |x - L/2| \le l. \end{cases} \tag{7}$$

[2]Note that the known values at the two boundary points, `vz[0]` and `vz[N]`, are included in the *right* hand side of the equation.

Subsequently, the temperature profile relaxes to a new equilibrium distribution. The partial differential equation for thermal diffusion reads as

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}, \tag{8}$$

where α denotes the ratio of the heat conduction coefficient to the heat capacity per unit length. Since the bar is isolated, the temperature gradient will equal zero at both ends,

$$\left.\frac{\partial T(x,t)}{\partial x}\right|_{x=0} = \left.\frac{\partial T(x,t)}{\partial x}\right|_{x=L} = 0. \tag{9}$$

To calculate the evolving temperature profile, we will store the profile in a two-dimensional array: the array element `T[i,j]` represents the temperature at position $x = \mathtt{i} * \Delta x$, with the integer i running from 0 to N and $\Delta x = L/N$, and time $t = \mathtt{j} * \Delta t$, with integer j.

Questions

5 Given the differential equation of Eq. (8), construct an algorithm that uses the forward Euler method to advance the temperature profile $T(x,t)$ at time t to the profile $T(x, t+\Delta t)$ a timestep Δt later. Note that the advancement of $\mathtt{T}[\mathtt{0},\mathtt{j}] = T(0,t)$ and $\mathtt{T}[\mathtt{N},\mathtt{j}] = T(L,t)$ by Eq. (8) requires knowledge of the temperatures outside the bar, at $\mathtt{T}[-\mathtt{1},\mathtt{j}]$ and $\mathtt{T}[\mathtt{N}+\mathtt{1},\mathtt{j}]$, which is clearly not permitted. Use the boundary conditions of Eq. (9) to eliminate these two temperatures from your discretization of Eq. (8).[3]
The relevant numerical parameters (T_0, δT, L, etc) may be chosen freely.

6 Given the initial profile of Eq. (7), calculate the profiles at several later times.
Do these profiles agree with your expectations? Test whether your program obeys the appropriate conservation law.

7 Use the matlab solver `pdepe` to calculate the evolving temperature profile.
Compare these results with those of your own routine by plotting them in the same figure.

8 In the limit of l going to zero, the solution at short times[4] reads as

$$T(x,t) - T_0 \propto \frac{1}{\sqrt{2\pi\alpha t}} \exp\left(-\frac{(x-L/2)^2}{2\alpha t}\right). \tag{10}$$

This solution, known in the literature as the Wiener process, shows scaling behaviour: with proper time-dependent multiplication factors – applied along both the x and T direction – the profiles at various times coalesce onto a single 'master' curve.[5] Determine these scaling factors from the given analytic solution, and test whether your rescaled numerical solutions obey this scaling behaviour.

Supplementary questions (for grade > 8)

Vibrating string

Differential equations are also encountered in musical instruments. The sound produced by a string of a guitar or piano is determined by the length L, the tension σ and the mass

[3] As usual, a symmetric approximation of the derivative works better than an asymmetric approximation.
[4] Short times implies that the diffusing heat has not yet reached the ends of the rod.
[5] And with some more work, one can even make the profiles coalesce onto a single straight line.

density ρ of the vibrating string. For a string oriented along the x direction and vibrating in the y direction,[6] the equation of motion reads as

$$\frac{\partial^2 u}{\partial t^2} = \frac{\sigma}{\rho}\frac{\partial^2 u}{\partial x^2}, \tag{11}$$

where $u(x,t)$ denotes the displacement from the equilibrium straight line at position x and time t. Both ends of the string are fixed,

$$u(0,t) = u(L,t) = 0. \tag{12}$$

To determine the solutions periodic in time, we start with the trial function

$$u(x,t) = y(x)e^{i\omega t}. \tag{13}$$

Insertion into Eq. (11) gives

$$\frac{\partial^2 y}{\partial x^2} = -\omega^2\frac{\rho}{\sigma}y, \tag{14}$$

where both sides of the equation were divided by the non-zero factor $\exp(i\omega t)$. The combination of Eq. (11) with the boundary conditions presents an *eigenvalue problem.*

Questions

a Introduce $N+1$ points on the interval from $x = 0$ to $x = L$, at constant intervals of length L/N. Discretize the differential equation, and use the two boundary conditions, to arrive at an eigenvalue problem in matrix-vector form,

$$\mathbf{M}\mathbf{y} = \lambda\mathbf{y}, \tag{15}$$

where $\mathbf{M}$ is an $(N-1)\times(N-1)$ matrix and $\mathbf{y}$ a column vector with $N-1$ elements.

b Use matlab to numerically solve the eigenvalues λ_n and eigenvectors $\mathbf{y}_n$ of Eq. (15), with n the ordinal number labelling the solutions.
Plot the first couple of eigenvectors, *i.e.* those with eigenvalues closest to zero, and verify whether they agree with your expectations.

c Plot the eigenfrequency ω_n as a function of the wavenumber $k = \pi n/L$, where the modes are assumed sorted by ascending frequency.[7] The analytic solution to the discretized wave equation yields the dispersion curve[8]

$$\omega(k) = \frac{2N}{L}\sqrt{\frac{\sigma}{\rho}}\sin\left(\frac{kL}{2N}\right). \tag{16}$$

Compare your results against this expression. The slope of $\omega(k)$ at low k gives the velocity of wave propagation, v_s, in the string. Calculate this slope and compare it with the theoretical result $v_s = \sqrt{\sigma/\rho}$.

[6] We ignore motion in the third spatial dimension.
[7] That is, $0 \le \omega_1 \le \omega_2 \ldots$.
[8] The analytic solution to the continuous wave equation yields the dispersion curve $\omega = \sqrt{\sigma/\rho}\,k$.