COMP5930M - Scientific Computing

Coursework 1

September 21, 2022

Deadline

Monday 21st November at 09:00

Task

The three numbered sections of this document describe problems which can be modelled by nonlinear equations. In each case a proper formulation of the problem as a nonlinear system is required with appropriate initial data. You will use one or more of the algorithms we have covered in the course to produce a numerical solution.

MATLAB scripts referred to in this document can be downloaded from the VLE under `Learning Resources / Coursework`. MATLAB implementations of some algorithms have been provided as part of the course but you can implement your solutions in any other language (e.g. Python) if you prefer.

You should submit your answers as a single PDF document via the VLE before the stated deadline. Source code submitted as part of your answers should be included in the document. All functions should include appropriate `help` information that describe the purpose and use of that function.

Standard late penalties apply for work submitted after the deadline.

You should properly cite any sources of information used outside of the course material.

Disclaimer

This is intended as an individual piece of work and, while discussion of the work is encouraged, plagiarism in any form is strictly prohibited.

1. Implicit solution of a physical ODE system

[10 marks total]

The ODE system

$$\begin{aligned} \frac{d\theta}{dt} &= \phi \\ \frac{d\phi}{dt} &= -\sin(\theta) \end{aligned} \tag{1}$$

models the motion of an undamped, unit-length pendulum in terms of the angle to vertical $\theta(t)$, and angular velocity $\phi(t)$. Initial conditions can be taken at $t = 0$ as $\theta_0 = -\frac{\pi}{4}$, $\phi_0 = 0$. We consider the numerical solution of this problem with two different numerical methods.

- A simple explicit scheme to solve (1) is obtained using the Forward Euler approximation: solve for (θ_i, ϕ_i) at times $t_i = i\ dt, i = 1, 2, ..., n$ s.t.

$$\begin{aligned} \theta_i &= \theta_{i-1} + dt\ \phi_{i-1} \\ \phi_i &= \phi_{i-1} - dt\ \sin(\theta_{i-1}) \end{aligned}$$

  starting from an appropriate initial state (θ_0, ϕ_0) and using a fixed time step dt. This scheme is implemented in the Matlab function `feuler.m` provided.

- One possible implicit scheme to solve the nonlinear equation $\frac{du}{dt} = f(u(t), t)$ is obtained using the Midpoint Method: solve for u_i at times $t_i = i\ dt, i = 1, 2, ..., n$ s.t.

$$\begin{aligned} u_{i-1/2} &= u_{i-1} + \frac{dt}{2} f(u_{i-1/2}, t_{i-1/2}), \\ u_i &= 2u_{i-1/2} - u_{i-1}, \end{aligned}$$

  starting from an appropriate initial state u_0 and using a fixed time step dt. Here $u_{i-1/2}$ is an intermediate solution obtained by taking a half-step with the Backward Euler Method, followed by an update step for u_i. The Midpoint Method is an example of a *symplectic method*, i.e. it conserves numerically the energy of a conservative system[1].

(a) Solve problem (1) with the Forward Euler method with parameters $n = 40$, $dt = 0.25$ and plot both solution variables in time up to $t = 10$. What do you observe? Is this a physically realistic solution of the pendulum problem?

(b) Formulate the Midpoint Method approximation for (1) as a system of two nonlinear equations using a fixed timestep dt. Write down the nonlinear equations to solve at each time step for the solution (θ_i, ϕ_i).

(c) Implement the Midpoint Method for problem (1) in MATLAB or Python. You can use the file `feuler.m` as a basis for your implementation. Include the source code in your answer.

(d) Use your Midpoint Method implementation `midpoint.m` to solve the problem with parameters $n = 40$, $dt = 0.25$ and plot both solution variables at time $t = 10$. What do you observe? Is this a physically realistic solution of the pendulum problem?

[1] The Backward Euler method is not a symplectic method and does not typically conserve energy numerically.

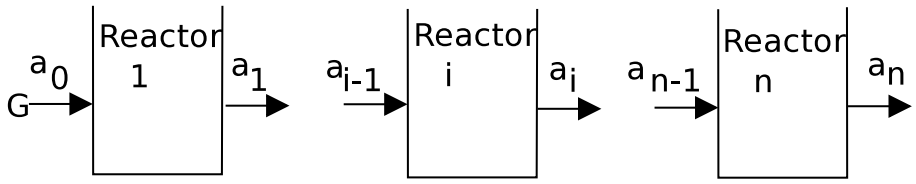

Figure 1: A series of reactor vessels

2. Chemical engineering

[10 marks total]

Figure 1 shows a series of n continuously-stirred reactor vessels, each of volume V, that are used for a particular chemical reaction. Materials are fed into reactor 1 at rate G. The chemical of interest has concentration a_{i-1} at the inlet and concentration a_i at the outlet to reactor i in the series, which feeds into reactor $i+1$.

Given a reaction rate constant k, we can write a nonlinear equation for the steady-state concentration in the reactor as

$$\beta a_i^2 + a_i - a_{i-1} \;=\; 0$$

where $\beta = \frac{kV}{G}$.

This leads to a system of n nonlinear equations for the series of n reactors.

We assume that values for V, G, a_0 and k are known constants.

(a) For the case $n = 5$ reactors, write out the nonlinear equation system in the form $\mathbf{F}(\mathbf{U}) = \mathbf{0}$ defining $\mathbf{F}$ and $\mathbf{U}$. Implement the system in MATLAB code.

(b) Derive the analytical formulas for elements of the Jacobian matrix in the case $n = 5$.

(c) Produce a numerical solution for the $n = 5$ case using Newton's method with line-search (provided in `newtonSysL.m`) and the following parameters: $V = 1.0,\ G = 45.0,\ k = 0.7,\ a_0 = 6.0$. Use the initial guess $\mathbf{a} = (5, 4, 3, 2, 1)$, the stopping tolerance $tol = 10^{-12}$, and maximum of 500 Newton iterations. How many Newton iterations do you observe? What is the numerical solution vector?

(d) Modify the Newton code to implement the *gradient descent method with line-search* instead. Provide the modified code as part of your answer

(e) Solve the problem for $n = 5$ with the same parameters as before using the gradient descent method. Compare the convergence to that of Newton's. What do you observe?

3. Visualisation of a Newton fractal

[10 marks total]

The convergence of Newton's method from different initial guesses is very unpredictable. This can be illustrated by visualising *Newton fractals*. A Newton fractal for a complex-valued polynomial function, which we choose here as

$$f(z) = z^3 - z^2 + z - 1, \quad z \in \mathbb{C},$$

can be computed by solving the equation $f(z) = 0$ with Newton's method starting from different initial guesses $z_0 \in \mathbb{C}$ in the complex plane and visualising either the number of iterations required for convergence or the specific root that is found (see for example Figure 1 of the Newton fractal for $f(z) = z^3 - 1$).

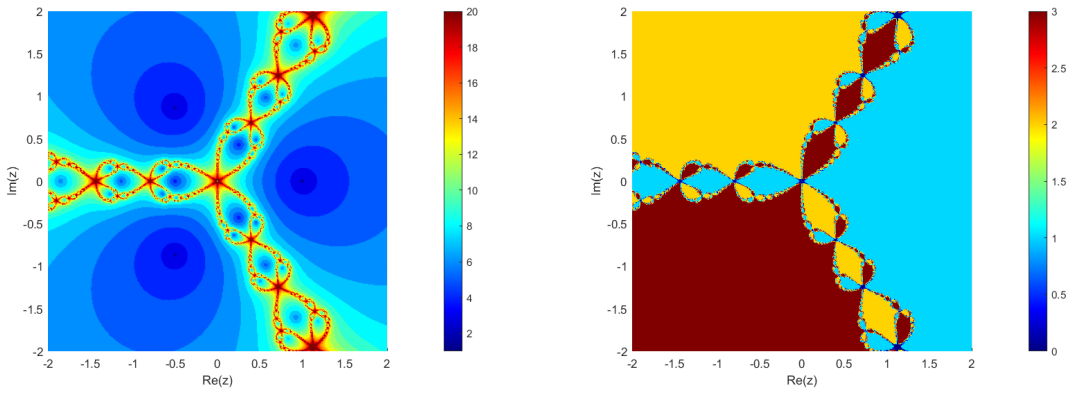

Figure 2: Newton fractals plotted by number of iterations (left) and the root reached (right)

(a) Write a MATLAB/Python code to solve the equation $z^3 - z^2 + z - 1 = 0$ using Newton's method for initial values z_0 in the rectangle $R := [-2, 2] \times [-2, 2] \subset \mathbb{C}$. Discretise the rectangle R using a regular grid of size 401×401 and for each grid point z_0 solve the problem using Newton's method with the initial guess z_0, stopping tolerance 10^{-6} and a maximum of 20 Newton iterations.

(b) Plot the Newton fractal for $z^3 - z^2 + z - 1$ in the rectangle R *coloured by the number of iterations required* $iters(z_0)$ to converge from each grid point z_0. Provide a colourbar indicating how many Newton iterations the method requires to converge from each point z_0. Indicate in the figure where the roots of the function $f(z)$ are located.

(c) Plot the Newton fractal for $z^3 - z^2 + z - 1$ in the rectangle R *coloured by which of the roots was reached* $z^*(z_0)$ to converge from each grid point z_0. Provide a colourbar indicating which of the roots was reached. Indicate in the figure where the roots of the function $f(z)$ are located.

Marking scheme

There are 30 marks in total.

1. Implicit solution of a physical ODE system [10 marks total]
   (a) [2 marks] analysis of Forward Euler solution
   (b) [3 marks] analytical formulas for Midpoint Method
   (c) [3 marks] implementation of Midpoint Method
   (d) [2 marks] analysis of Midpoint Method solution
2. Chemical engineering [10 marks total]
   (a) [2 marks] nonlinear system
   (b) [2 marks] Jacobian matrix
   (c) [2 marks] Newton solution
   (d) [2 marks] gradient descent code
   (e) [2 marks] gradient descent solution
3. Visualisation of a Newton fractal [10 marks total]
   (a) [4 marks] code for computing the roots
   (b) [3 marks] Newton fractal by number of iterations
   (c) [3 marks] Newton fractal by root found