Series 11

A discretization of the Brusselator problem in Matlab

The aim of this series is to implement and study in Matlab numerical discretizations of the Brusselator problem (see Exercise 4 of Series 10)

$$\begin{aligned} \frac{\partial u}{\partial t} &= a + u^2 v - (b+1)u + \nu \Delta u, \\ \frac{\partial v}{\partial t} &= bu - u^2 v + \nu \Delta v. \end{aligned} \tag{1}$$

We consider the parameters $a = 1$, $b = 3, \nu = 0.02$, the boundary conditions

$$u(0,t) = u(1,t) = 1, \qquad v(0,t) = v(1,t) = 3, \tag{2}$$

and the initial conditions

$$u(x,0) = 1 + \sin(2\pi x), \qquad v(x,0) = 3. \tag{3}$$

Preliminaries: space discretization

We consider the discretization of the space $[0,1]$ into $N_x + 1$ subintervals of length $\Delta x = 1/(N_x+1)$. The space discretization for $u(x,t)$ (and similarly for $v(x,t)$) is

$$u(x_i,t) \approx u_i(t), \quad \text{where } x_i = \frac{i}{N_x+1} \qquad 0 \le i \le N_x + 1.$$

Since $u_0(t)$ and $u_{N_x+1}(t)$ are already known from the boundary conditions (3), the unknown function becomes the vector of size N_x, $U(t) = (u_1(t), \ldots, u_{N_x}(t))^T$.

Chemical reaction

Write a Matlab function `[fu,fv]=reac_fun(u,v)` which computes the vectors `fu` and `fv` corresponding to the chemical reaction in (1), i.e., without the diffusion terms, when two vectors $\mathtt{u} = (u_1, u_2, u_3, \ldots)^T$ and $\mathtt{v} = (v_1, v_2, v_3, \ldots)^T$ are provided, i.e., it calculates

$$\mathtt{fu}_i = a + u_i^2 v_i - (b+1)u_i, \qquad \mathtt{fv}_i = bu_i - u_i^2 v_i, \qquad 1 \le i \le N_x.$$

Explicit Euler method

Implement in Matlab the explicit Euler method for problem (1) without diffusion (i.e., for $\nu = 0$). Compute the solution on the time interval $[0,T]$ with $T = 10$ and time step $\Delta t = T/N_t$, where N_t is a chosen number of time steps.

The program should compute two matrices `vecu` and `vecv` of size $(N_x+2) \times (N_t+1)$, where the first columns correspond to the initial conditions $u(t=0)$ and $v(t=0)$ on the space $[0,1]$, the second columns are the solutions at time $t = \Delta t$, the third columns are the solutions at time $t = 2\Delta t$, etc.

3D plot

Plot in 3D the solution for $u(x,t)$ using the `surf` command:

```
x=linspace(0,1,Nx+2);
t=linspace(0,T,Nt+1);
vecx=repmat(x',1,Nt+1);
vect=repmat(t,Nx+2,1);
surf(vecx,vect,vecu);
```

Adding diffusion terms

The Laplacian Δu (and similarly for Δv) can be discretized by

$$\begin{pmatrix} \Delta u(x_1,t) \\ \vdots \\ \Delta u(x_{N_x},t) \end{pmatrix} \approx A \begin{pmatrix} u_1(t) \\ \vdots \\ u_{N_x}(t) \end{pmatrix} + B_u,$$

where the matrix A of size $N_x \times N_x$ is given by

$$A = \frac{1}{\Delta x^2} \begin{pmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & 1 & \ddots & \ddots \\ & & \ddots & \end{pmatrix},$$

and B_u is a constant vector determined by the boundary conditions (3). The matrix A can be defined in Matlab using the `diag` command (e.g., try `diag(ones(1,9),0);`) or when working with sparse matrices using the `spdiags` command.

Question: What are the vectors B_u and B_v for the boundary conditions (3)?

Stability of the explicit Euler method

Take a reasonable space discretization (e.g., $N_x = 30$) and compute the numerical solution of the Brusselator problem (1) with diffusion in the time interval $[0,T]$, where $T = 10$, using different time steps $\Delta t = T/N_t$.

Verify numerically that the explicit Euler method is stable for $\Delta t \leq \Delta x^2/(2\nu)$. This condition is called the Courant–Friedrichs–Lewy (CFL) condition, as derived in Exercise 4 of Series 10.

Implicit Euler method

Implement the implicit Euler method, which for a system of a differential equation is defined by

$$y_1 = y_0 + hf(y_1),$$

with the two following methods for solving the nonlinear system:

- **Fixed point iteration:**

$$Y^{k+1} = y_0 + hf(Y^k). \tag{4}$$

- **Newton iteration:**

$$(I - hf'(Y^k))(Y^k - Y^{k+1}) = (Y^k - y_n - hf(Y^k)). \tag{5}$$

In both cases, the value of y_1 is approximated by the sequence $\{Y^k\}_{k\geq 0}$, where $Y^0 = y_0$.

Compare the two implementations of the implicit Euler method and show that the one with Newton iteration has the correct stability behaviour (i.e., no restriction on the step size).

Convergence criterion

Implement the following stopping criterions for the iterations (4) and (5):

- *a fixed number of iterations:* stop when $k > K$;
- *small increment:* stop when $\|Y^{k+1} - Y^k\|_\infty < tol$ where, e.g., $tol = 10^{-13}$;
- *iterations until machine precision:* stop when $\|Y^{k+1} - Y^k\| = 0$, i.e., the sequence does not change anymore, or $\|Y^{k+1} - Y^k\| > \|Y^k - Y^{k-1}\|$, i.e., the sequence starts to oscillate due to roundoff errors.