Responsible TA:
Asbjörn Rasmussen
asbjoern.rasmussen@chem.ethz.ch

Statistical and Numerical Methods
Exercise 4, HS24

15/10/2024

1 Systems of Nonlinear Equations (core)

The steady state concentrations of a CSTR with two second order reactions taking place reads

$$\begin{aligned} 0 &= (x_1^{in} - x_1) + \tau(-k_1 x_1 x_2) \\ 0 &= (x_2^{in} - x_2) + \tau(-k_1 x_1 x_2 - k_2 x_2 x_3) \\ 0 &= -x_3 + \tau(k_1 x_1 x_2 - k_2 x_2 x_3) \\ 0 &= -x_4 + \tau(k_2 x_2 x_3) \end{aligned} \tag{1}$$

1. Write down the analytical Jacobian matrix for the system of equations (1).
2. Implement the basic Newton method.
   - The multi-dimensional Newton iteration formula reads
   $$x_{k+1} = x_k - \mathbf{J}^{-1}(x_k) f(x_k)$$
   - Your function file header should read something like `function [x, info] = newtonMethod(f, J, x0, tol)` where `f` is a function handle to the function you want to solve, `J` is a function handle that returns the Jacobian matrix, `x0` is an initial guess and `tol` is a vector of tolerances for stopping criteria (relative, absolute errors and number of iterations).
   - As in with the secant method, use a while loop to find the solution.
   - Suggest stopping criteria and failure checks. When can the Newton method fail in general?
   - Use left division \ to solve the linear system at every iteration (do not use `inv(J)`!)
   - Let `info` be a `struct` you can use to return additional information, like reason of termination and number of steps needed.
3. Use your Newton algorithm to solve the steady state CSTR numerically.
   - Create a main file and two function files; one that calculates the CSTR equations (1) as functions of x, and one that calculates the analytical Jacobian as a function of x.
   - Use $k_1 = 0.5$, $k_2 = 10$, $x_1^{in} = 1.0$, $x_2^{in} = 1.5$ and $\tau = 5$
   - What is the total conversion of A (x_1) to D (x_4)?
   - Compare your result to what `fsolve()` finds. Try different starting guesses. Can you find more than one solution?
4. (Optional) Find online (same place where you found the exercise sheet) the function `jacobianest`. It is part of a user-made toolbox for estimating derivatives numerically, the DERIVEST suite which can be found on the MatlabCentral.
   - Modify your Newton algorithm so that it uses `jacobianest` to approximate the Jacobian if the input **J** is empty (use `isempty(J)` to check). To provide an empty input, use [] in the call.
   - How many steps are required with the analytical Jacobian for this specific case compared to the numerical Jacobian? Which algorithm takes longer?

2 Concentration evolution via ODE (core)

A decaying radioactive element changes its concentration according to the ODE:

$$\frac{dy}{dt} = -ky \tag{2}$$

The analytical solution reads:

$$y(t) = y_0 \exp(-kt) \tag{3}$$

1. (Optional) Plot the behavior of the radioactive decay problem as a vector field in the y vs t plane
   - Find online the function vector_field.m. It plots the solutions and derivatives of first order initial value problems (IVPs) for different initial conditions
   - Plot the vector field for different initial values between 0 and 1, time between 0 and 10 and $k = 1$. Can you see what a solver has to do?
   - Is it possible to switch from one trajectory in the vector field to another? What follows for the uniqueness of the solutions?
2. The forward Euler method reads for this problem

$$y_{n+1} = y_n + h * f(t_n, y_n) = y_n - h * k * y_n \tag{4}$$

   - First define a function defining the successor of yn with a header like function ynp = Forward_stepper(k, yn, h)
   - Use the conditions y0 = 1, k = 1 and h = 0.1 to solve the radioactive decay problem from t0 = 0 to tEnd = 10 by defining an integrating function of the form function [T,Y] = stepper_integrate(@stepper,k,t0,tEnd,y0,h)
3. The backward Euler method uses the following step formula

$$y_{n+1} = y_n + h * f(t_{n+1}, y_{n+1}) \tag{5}$$

   - Rearrange this equation so that you can define a function defining the successor of yn with a header like function ynp = Backward_stepper(k, yn, h)
   - Use the prior defined integration function to solve the problem with the backward method.
4. Plot the obtained solutions comparing them to the analytical solution. Use the subplot method to produce two subplots in a single figure. What happens if you increase the step size h? What happens if you increase the step size above 2?

3 High order ODEs (core)

Convert the following fourth order initial value problem

$$y^{(4)}(t) = \cos(\ddot{y}(t)) + \dot{y}(t)e^{-5t} \tag{6}$$

with initial conditions

$$y(t_0) = 0,\ \dot{y}(t_0) = 3,\ \ddot{y}(t_0) = -1,\ \ddot{y}(t_0) = -1,\ \dddot{y}(t_0) = 0 \tag{7}$$

into a first order initial value problem.