AMATH740/CM770/CS770
Fall 2020: Computational Assignment C

Instr.: Hans De Sterck e-mail: *hdesterck@uwaterloo.ca* Office hrs: Wed 9:00-9:45am, Thu 11am-12
TA.: Mohammad Aali e-mail: *mohammad.aali@uwaterloo.ca* Office hrs: Tue 11am-12, Thu 9-10am

Assignment due date: Monday December 7, 5:00pm (Waterloo time), Crowdmark (for the answer package) and LEARN (for the program code files)

Submission instructions: Dual submission

- **LEARN submission**: Please submit your program code files to the Dropbox on LEARN (only the program code files; no program output or answers to questions).
- **Crowdmark submission**: Please submit the answers to each assignment question on Crowdmark, including written or typed answers to the questions asked, with relevant computer output (plots, tables or number output) as requested in the assignment questions. Please also include, for each question, a pdf file or screenshot with your program code (to facilitate TA feedback on your code).
- one common submission per group
- MATLAB is the recommended computer language, but you can choose a different language like python or Julia or C or C++ (contact the instructor if you want to choose a language not listed before; Maple or Mathematica are not suitable).
- The questions are written assuming you use MATLAB, and some short MATLAB code fragments are provided on LEARN for download, but it should not be difficult to translate this to another language like python, and I don't think there should be much extra work when using another language; let me know if you encounter issues or have questions about this
- there are some potentially useful tools for programming in teams that you may consider to use, such as version control tools like git (e.g., you can create a git repository for your group on github or bitbucket) (but the computer code you will write for the programming questions will generally not be long or complicated, so it is by no means necessary to use that kind of tools); collaboration tools like slack could also be useful (or you can communicate and share files via tools from Google, or MS Teams, etc.)

1. Lorenz equations and chaos. (20 marks)

   You are asked to implement the Ralston 2-stage Runge-Kutta method for $y'(x) = f(x, y(x))$, given by

$$
\begin{aligned}
k_1 &= f(x_n, y_n), \\
k_2 &= f\left(x_n + \frac{2}{3}h, y_n + \frac{2}{3}h\, k_1\right), \\
y_{n+1} &= y_n + h\left(\frac{1}{4}k_1 + \frac{3}{4}k_2\right),
\end{aligned}
$$

for the numerical solution of ODE systems

$$U'(x) = F(x, U(x)),$$

and apply it to simulate chaotic solutions of the nonlinear Lorenz equation system.

Download the files `driver1.m`, `FESystem.m` and `EqSystemTest.m` from the class website. These files provide details about the required input and output variables for the routines and an example implementation of the Euler method (which is also called the Forward Euler method, or FE), and will allow you to test your implementation.

Study the implementation of the system Forward Euler method that is provided in Matlab m-file `FESystem.m`, with function heading

`function U=FESystem(F,a,b,U0,steps).`

Here `[a,b]` is the interval where the approximate solution is sought, and `steps` is the number of subintervals. For the n-dimensional system $U'(x) = F(x, U)$, `U` is an $n \times (steps + 1)$ result matrix with the discrete approximations, and `U0` is a column vector with the initial condition. The RHS function `F` can be specified in a separate m-file (see `driver1.m` and `EqSystemTest.m`).

(a) **Implement the 2-stage Ralston method in a Matlab m-file that you name `RalstonSystem.m`, with function heading**

`function U=RalstonSystem(F,a,b,U0,steps).`

Test your implementation with `driver1.m`, and include the plot it generates in your assignment package to be submitted on Crowdmark. What do you observe, which of FE or Ralston gives the most accurate solution? Explain why. Also, include a copy of your `RalstonSystem.m` in your assignment package.

(b) Compare the order of accuracy of the two methods applied to the test problem specified in `EqSystemTest.m`. (Extend `driver1.m` to do this.) Define the error

$$e_i = \|U_i - V_i\|_2$$

where U_i is the exact solution vector and V_i the numerical approximation at discrete location x_i. **For each of the FE and Ralston methods and for** $steps = 16, 32, 64, 128, 256$**, make a table of the error** e_i **at the endpoint of the interval (i.e.,** i **is the index of the endpoint of the interval). What can you conclude about the global order of accuracy for each of the methods?** Include a copy of your extended `driver1.m` in your assignment package.

(c) Consider the Lorenz equations, which are given by

$$\begin{cases} x'(t) &= \sigma\,(y(t) - x(t)), \\ y'(t) &= r\,x(t) - y(t) - x(t)z(t), \\ z'(t) &= x(t)y(t) - b\,z(t), \end{cases}$$

where σ, b and r are fixed parameters $\in \mathbb{R}$ (assumed to be positive) and t is time.

These equations are derived from fluid mechanics, and are related to weather prediction. The equations are nonlinear (e.g., there are products $x(t)z(t)$ and $x(t)y(t)$) and there are no closed-form solutions, so numerical solutions are required.

- Implement the RHS function F of the Lorenz system in a Matlab m-file that you name `EqLorenz.m`, with function heading

  `function F=EqLorenz(t,U)`,

  where $U(1) = x$, $U(2) = y$, and $U(3) = z$ and x, y, z are functions of t. Use parameters $\sigma = 10, b = 8/3, r = 28$. Include a copy of your code in your assignment package to be submitted on Crowdmark.
- Write a Matlab pogram `driverLorenz.m` that uses the Ralston method (or the system Forward Euler method if you did not get Ralston to work) to calculate and plot solutions for the Lorenz system. (You can adapt `driver1.m`.) For initial condition $(x_0, y_0, z_0) = (1, 0, 0)$ and $t \in [0, 50]$, produce a plot of $x(t)$ and a plot of the solution trajectory $(x(t), z(t))$ in the xz-plane. A good value for the time step is $\Delta t = 0.002$. (With larger time steps, the method may become unstable, or the accuracy may be too low.) Include a copy of your code `driverLorenz.m` and the two plots (generated by your code) in your assignment package.
- Hmm, the solution you generated looks very strange. Just like Lorenz in the 1960s, you have re-discovered (mathematical) "Chaos"! (Lorenz also used the FE method when he first observed Chaos numerically.) You can look up "Chaos theory" on Wikipedia and have a look at some of the properties of this phenomenon as described there. Your plot $(x(t), z(t))$ in the xz-plane reflects a "strange attractor". The solution remains in a bounded region of space, but it does not converge to an equilibrium point or cycle, and it appears to fill up the space locally. The attractor has a fractal structure: it fills up the local 3D $(x, y, z)-$space more than a plane (dimension 2), but not entirely (dimension 3); its Haussdorf dimension has been measured to be about 2.06.

  One of the properties of chaos is sensitivity to initial conditions: small differences in initial conditions (such as those due to rounding errors in numerical computation) yield widely diverging outcomes (for a solution that remains bounded), rendering long-term prediction problematic. This has been summarized as: "Chaos: When the present determines the future, but the approximate present does not approximately determine the future." Following Lorenz' 1972 paper with title "Predictability: Does the Flap of a Butterfly's Wings in Brazil set off a Tornado in Texas?", sensitivity to initial conditions is now popularly known as the "butterfly effect".

  You are asked to illustrate the property of "sensitivity to initial conditions": change the x_0 initial condition to $x_0 = 1.0001$, and make a plot that overlays the values of $x(t)$ with $x(t)$ for the case $x_0 = 1$. (Use "hold on" and diffferent colours or linestyles.) Submit this plot as part of your assignment package, with a brief discussion (at most one paragraph) on how the plot illustrates sensitivity to initial conditions.

2. Adaptive RK45 method. (20 marks)

   Download the matlab m-files `adaptiveRK.m`, `driver2.m` and `FBrusselator.m` from the class website. You are asked to implement the adaptive RK45 method that was described in class, and apply it to a nonlinear ODE system called the 'Brusselator' ODE system

   $$U'(t) = F(t, U(t)),$$

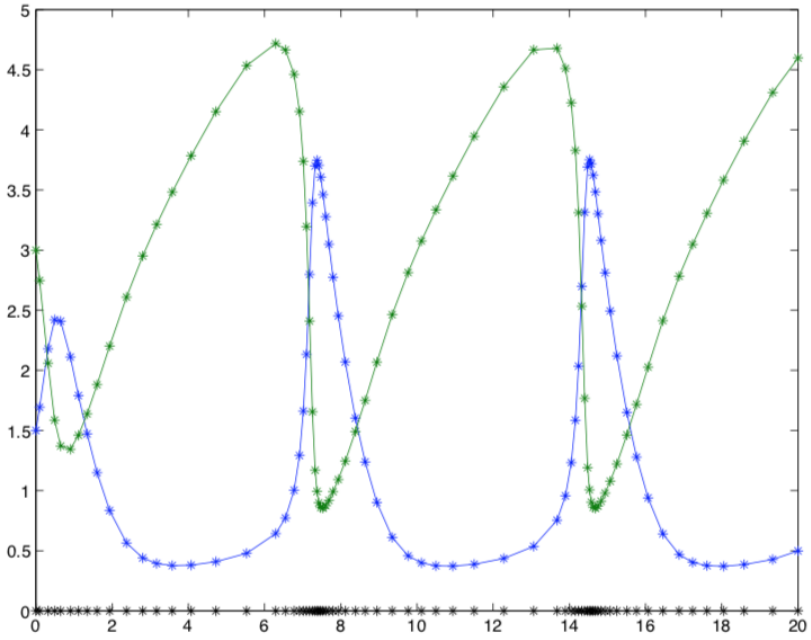
   given by

$$u_1'(t) = 1 + u_1(t)^2 u_2(t) - 4u_1(t),$$
$$u_2'(t) = 3u_1(t) - u_1(t)^2 u_2(t).$$

This system describes autocatalytic reactions in chemistry. The Brusselator IVP you will solve is given by

$$U'(t) = F(t, U(t)),$$
$$u_1(0) = 1.5 \text{ and } u_2(0) = 3,$$
$$t \in [0, 20],$$

and the numerical solution you will reproduce looks like this:



(a) The adaptive RK45 method uses a Runge-Kutta base method, method A, with global accuracy of order 4, and a secondary Runge-Kutta method, method B, with global accuracy of order 5. For efficiency, methods A and B each use 6 stages where the intermediate RK slopes $K_j,\ j = 1, \ldots, 6$ computed in time step n are shared between the two methods. Using the slopes $K_j,\ j = 1, \ldots, 6$ computed in step n with step length h_n, methods A and B update the solution at time t_{n+1} from the solution at time t_n using a final slope that is a linear combination of the slopes $K_j,\ j = 1, \ldots, 6$, as follows:

$$U_{n+1}^{(A)} = U_n + h_n \left(\sum_{j=1}^{6} w_j^{(A)} K_j \right),$$

and

$$U_{n+1}^{(B)} = U_n + h_n \left(\sum_{j=1}^{6} w_j^{(B)} K_j \right).$$

The coefficients $w_j^{(A)}$ for method A are chosen such that the global order for method A is 4, and the coefficients $w_j^{(B)}$ for method B are chosen to produce an approximation with global order 5. The slopes $K_j,\ j = 1, \ldots, 6$ in step n are computed by

$$K_j = F\left(t_n + h_n\alpha_j, U_n + h_n\left(\sum_{i=1}^{j-1}\beta_{ij}K_i\right)\right),$$

where the coefficients β_{ij} and α_j are shared between methods A and B. The specific coefficients β_{ij}, α_j, $w_j^{(A)}$ and $w_j^{(B)}$ we will use can be found in file `adaptiveRK.m`. All details about the input and output arguments required are also given in the file `adaptiveRK.m`, and the script `driver2.m` provides the input parameters that will allow you to reproduce the adaptive simulation of the so-called 'Brusselator' ODE system shown above.

You are asked to implement the adaptive RK45 code into file `adaptiveRK.m`, and submit the file to the LEARN drop box on the course webpage. Also, include a copy of the code in your assignment package to be submitted on Crowdmark.

(b) Run your adaptive RK45 code applied to the Brusselator IVP with initial timestep 0.1 and local truncation error tolerances $\delta = 10^{-2}$, $\delta = 10^{-4}$ and $\delta = 10^{-6}$.

- For all three values of δ, make plots of U as a function of t.
- For $\delta = 10^{-6}$, make a phase-space plot, i.e., plot $(u_1(t), u_2(t))$ in the (u_1, u_2)-plane. How does the solution behave for large t?
- For $\delta = 10^{-6}$, how many function evaluations (computations of a K_j) does the algorithm perform? What is the smallest step taken by the adaptive method?
- One way to compare the efficiency of the adaptive method with a fixed timestep method, is to count how many function evaluations would be needed in a fixed step method with the the same steplength as the smallest steplength in the adaptive simulation. (This would guarantee that local errors would not be much larger in the fixed step method than in the adaptive simulation.) Make this comparison for $\delta = 10^{-6}$, and conclude whether an adaptive or a fixed step calculation is most efficient for this IVP.

As always, include all answers, plots and code in your Crowdmark submission.

3. Rootfinding methods. (20 marks)

(a) Implement each of the following four root finding algorithms discussed in class in MATLAB in a seperate m-file. You should use the function header given for each below.

(a1) Bisection method.

```
function root = bisection(f, a, b, maxit)
% Usage:  bisection(f, a, b, maxit)
% This method uses bisection to find roots of
% f in the interval [a,b] with maximum number of steps maxit.
```

(a2) Fixed point iteration.

```
function root = fixedpoint(g, x0, maxit)
% Usage:  fixedpoint(g, x0, maxit)
% This method uses fixed point iteration to
% find fixed points of g with maximum number of steps maxit
% using initial estimate x0.
```

(a3) Newton's method.

```
function root = newton(f, fprime, x0, maxit)
% Usage:  newton(f, fprime, x0, maxit)
% This method uses Newton's method to
% find roots of f (with derivative function fprime)
% with maximum number of steps maxit
% using initial estimate x0.
```

(a4) Secant method.

```
function root = secant(f, x0, x1, maxit)
% Usage:  secant(f, x0, x1, maxit)
% This method uses the secant method to
% find roots of f with maximum number of
% steps maxit using initial estimates x0 and x1.
```

Write an m-file driver script, `question3.m`, in which you use the four functions to find the root of

$$f(x) = x^2 - \exp(-x)/2.$$

The script should produce a table of the approximate solution for successive steps of each of the four methods, and a table of the absolute errors for all steps. Use `format long`. Use enough steps to reach full accuracy for all methods. You can use the `inline` function to construct f, $fprime$ and g in the driver script (or you can use separate m-files). For the fixed point method, use $g(x) = x - f(x)$. For bisection, use $[a, b] = [0, 2]$ as the initial interval. Use initial guess $x_0 = 1$ for the fixed point method, and $x_0 = 2$ for Newton. For the Secant method, use $x_0 = 2$ and $x_1 = 0$.

Investigate the convergence order and convergence constants for the Newton, Secant and fixed point methods. Make tables of the error ratios c_k for each step for each of these methods, and verify that c_k asymptotically tends to a constant. Verify that this is the theoretically predicted constant for the Newton and fixed point methods.

Submit your driver and function m-files to the LEARN drop box on the course webpage. Also, include a copy of the code in your assignment package submitted on Crowdmark. Your driver should produce the tables requested. Also provide printouts of your tables in your Crowdmark submission.

You will next apply your methods to a few more examples.

(b) Consider $f(x) = (x - \pi)|x - \pi|$. This function has one root, located at $x = \pi$. Plot this function using the `ezplot` command on the interval $[2, 4]$. Ensure that your plot has appropriate title and axis labels. Using the initial estimate $x_0 = 2$ apply Newton's method and the Secant method (with $x_1 = 4$) to find the root at $x = \pi$. What do you notice about the converge orders of each method? Explain why this behaviour does not contradict what you know about convergence of these methods.

(c) Consider

$$f(x) = -\frac{\exp(3x)}{9} - \frac{x^2}{2} + \frac{1}{9}.$$

Plot $f(x)$ and $g(x) = x - f(x)$ using `ezplot` and provide the resulting plot. Using fixed point iteration on $g(x)$, create a table of the results you get with initial condition $x_0 = 1$. Why are we unable to achieve convergence? Suggest a different $g(x)$ and x_0 that gives convergence with fixed point iteration.

(d) Consider $f(x) = -|x|^{1/3}$. Plot the function using `ezplot` and provide the resulting plot. Using Newton's method, create a table of the results you get with initial condition $x_0 = 1$. Explain graphically why we are unable to achieve convergence. Show analytically that Newton's method diverges for any initial guess. Does this contradict what you know about convergence of Newton's method?

(e) Consider $f(x) = x/(x^2 + 1)$. Plot the function using `ezplot` and provide the resulting plot. Using Newton's method, create a table of the results you get with the initial condition $x_0 = 0.75$. What do you observe? Does this contradict what you know about convergence of Newton's method? How can you fix this?