Numerical Analysis II: Homework Assignment 5
Spring 2018
Due April 20
Lindsay Eddy

This homework assignment covers materials on numerical methods for ODEs and revisits numerical integration and approximation theory.

1. [*10 points*] Using the following Runge–Kutta scheme, known as Heun's method [1, p. 338],

$$y_{n+1} = y_n + \frac{h}{2}\Big[f(x_n, y_n) + f\big(x_{n+1}, y_n + hf(x_n, y_n)\big)\Big],$$

solve the following problems,

$$y' = \frac{y}{4}\Big[1 - \frac{y}{20}\Big], \qquad y(0) = 1, \qquad Y(x) = \frac{20}{1 + 19e^{-x/4}},$$
$$y' = -y + 2\cos(x), \quad y(0) = 1, \qquad Y(x) = \cos(x) + \sin(x),$$

on interval $[0, 5]$; here $Y(x)$ is the true solution to the initial value problem (IVP). Use step sizes of $h = 5/(2^i)$, $i = 1, 2, \ldots, 7$. Report tables showing h, the errors $E_h = \max_n |Y(x_n) - y_n|$ and ratios E_{2h}/E_h. How do your results compare with theoretically expected rate of convergence? Discuss briefly.

Answer:

The errors and error ratios of the approximation to the IVP with true solution $Y(x) = \frac{20}{1+19e^{-x/4}}$:

h	E_h	E_{2h}/E_h
$5/(2^1)$	1.0569×10^{-01}	–
$5/(2^2)$	3.2321×10^{-02}	3.2700
$5/(2^3)$	8.9684×10^{-03}	3.6038
$5/(2^4)$	2.3636×10^{-03}	3.7943
$5/(2^5)$	6.0678×10^{-04}	3.8953
$5/(2^6)$	1.5372×10^{-04}	3.9472
$5/(2^7)$	3.8687×10^{-05}	3.9735

The errors and error ratios of the approximation to the IVP with true solution $Y(x) = \cos(x)+\sin(x)$:

h	E_h	E_{2h}/E_h
$5/(2^1)$	3.9252	–
$5/(2^2)$	6.5582×10^{-01}	5.9852
$5/(2^3)$	1.3557×10^{-01}	4.8374
$5/(2^4)$	2.9753×10^{-02}	4.5565
$5/(2^5)$	6.9298×10^{-03}	4.2935
$5/(2^6)$	1.6722×10^{-03}	4.1441
$5/(2^7)$	4.1077×10^{-04}	4.0709

Since Heun's method has order 2, we expect to see $E_h = O(h^2)$ as $h \to 0$. That is, we expect E_{2h}/E_h to approach 4 as $h \to 0$. We do observe this trend in the approximations to both IVPs.

2. [*20 points*] Consider the initial value problem $\boldsymbol{y}' = \boldsymbol{f}(\boldsymbol{y})$, where $\boldsymbol{y}(t) \in \mathbb{R}^4$ is the state vector, and the right hand side function and the initial state are given by

$$\boldsymbol{f}(\boldsymbol{y}) = \begin{bmatrix} -k_1 y_1 y_2 + k_2 y_3 \\ -k_1 y_1 y_2 + (k_2 + k_3) y_3 \\ k_1 y_1 y_2 - (k_2 + k_3) y_3 \\ k_3 y_3 \end{bmatrix}, \quad \boldsymbol{y}(0) = \begin{bmatrix} 5 \times 10^{-7} \\ 2 \times 10^{-7} \\ 0 \\ 0 \end{bmatrix}. \tag{1}$$

We consider the system in the time interval $[0, 100]$. Values of k_1, k_2, and k_3 are given by

$$k_1 = 10^8, \quad k_2 = 10^{-8}, \quad k_3 = 0.1.$$

In this problem, you will solve the system using the trapezoidal (Crank–Nicolson) method, which for a generic system $\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y})$ reads:

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{h}{2}\Big[\boldsymbol{f}(x_n, \boldsymbol{y}_n) + \boldsymbol{f}(x_{n+1}, \boldsymbol{y}_{n+1})\Big], \quad n \geq 0.$$

Note that this is an implicit method that requires solving a nonlinear system at each time-step. You will use Newton's method for this purpose.

(a) Describe the nonlinear system that needs to be solved at each step and describe the Newton iterations for solving this system. If you need to refresh your memory of Newton's method see [2, Chapter 5] (for those of you who don't know, this book is available in NCSU library's online system).

Answer:

At each step, the non-linear system $F(\boldsymbol{y}_{n+1}) = \boldsymbol{y}_{n+1} - \boldsymbol{y}_n - \frac{h}{2}\Big[\boldsymbol{f}(x_n, \boldsymbol{y}_n) + \boldsymbol{f}(x_{n+1}, \boldsymbol{y}_{n+1})\Big] = 0$ needs to be solved. Note that $\boldsymbol{y}_n$ is determined in the preceeding step, and h, x_n, and x_{n+1} are known. Letting $\boldsymbol{x} = \boldsymbol{y}_{n+1}$, we have

$$F(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{y}_n - \frac{h}{2}\Big[\boldsymbol{f}(x_n, \boldsymbol{y}_n) + \boldsymbol{f}(x_{n+1}, \boldsymbol{x})\Big] = 0.$$

At each iteration of Newton's method, we have

$$\boldsymbol{x}_{\text{next}} = \boldsymbol{x}_{\text{current}} - F'(\boldsymbol{x}_{\text{current}})^{-1} F(\boldsymbol{x}_{\text{current}})$$

where F' is the Jacobian of F. These iterations continue until the norm $\|F(\boldsymbol{x})\|$ is below a given tolerance, or until the maximum number of iterations has been met.

(b) Implement a Crank-Nicolson solver for the initial value problem described above. I have provided a generic implementation of Newton's method for solving a nonlinear system $F(\boldsymbol{u}) = \boldsymbol{0}$ along with this homework in Moodle that you can use in your solver. In each step, use explicit Euler to compute the initial iterate for the Newton iterations.

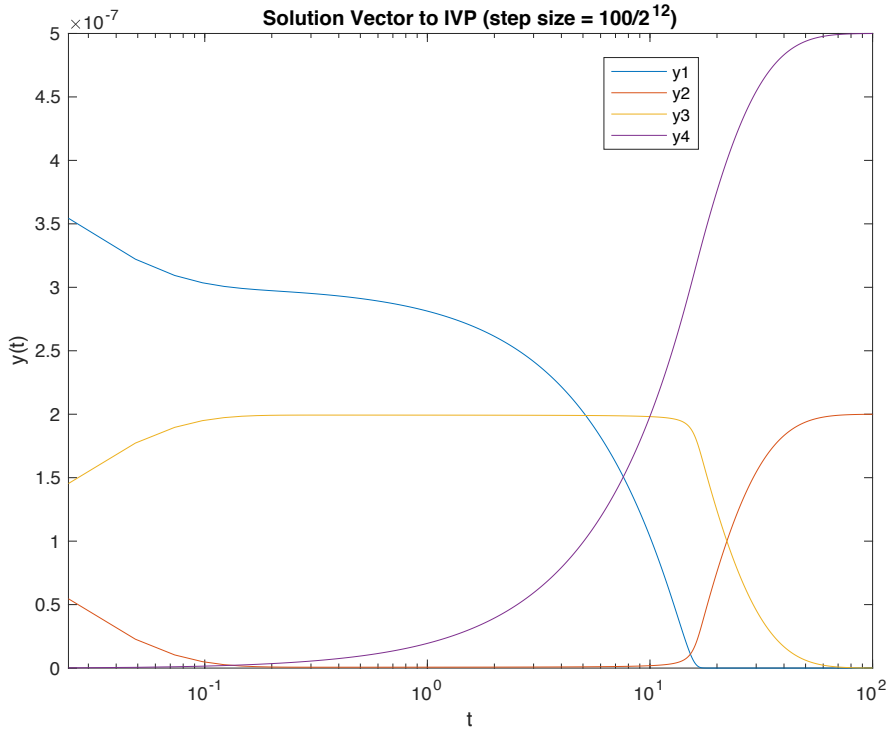
Answer:

See `CrankNicSolver.m`.

(c) Provide a plot of your computed solution vector over time in one figure; report your choice of step size, parameters for the Newton method, and overall computational time. Since there is a fast transient regime in this system, use logarithmic scale in the horizontal axis to be able to see the fast dynamics (use `semilogx`). Make sure to experiment with a few choices of steps sizes, but report only the plot corresponding to your most accurate solution. Comment on your results briefly.

Answer:

The figure below shows the solution vector obtained by running my Crank-Nicolson solver using a step size of $100/(2^{12})$ with the following Newton's parameters: absoulte tolerence $= 10^{-8}$, relative tolerance $= 10^{-8}$, and the maximum number of iterations equal to 10^3. With these parameters, the function ran for approximately 0.4 seconds. (The computational time varied somewhat.)



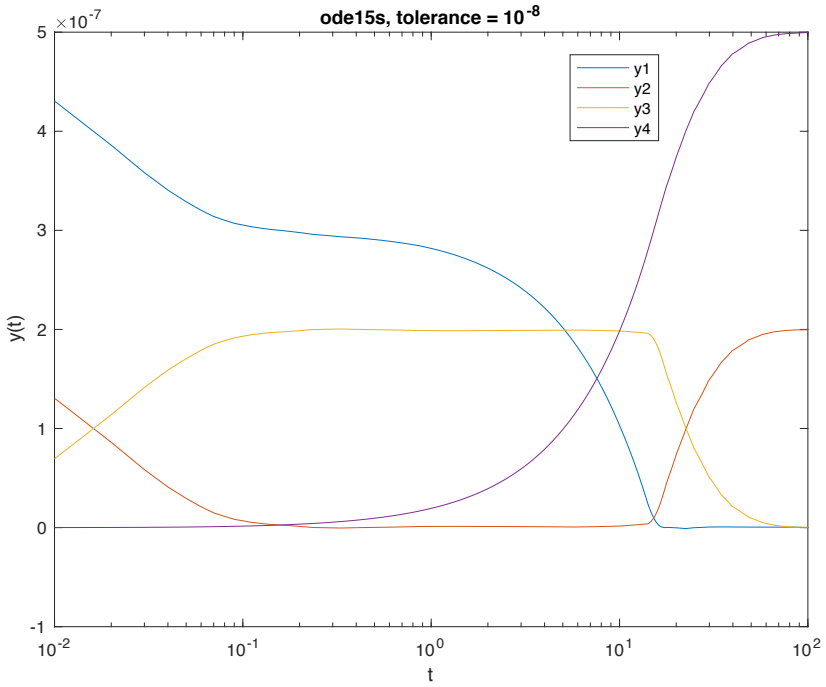
This small step size was necessary to obtain a reasonably accurate solution to the IVP. When the solver was run using larger step sizes, the solution differed visually a substantial amount from the solution obtained using `ode15s`. I noted that when `ode15s` was run with absolute and relative tolerances of 10^{-10}, the number of steps taken on the interval $[0, 1]$ (where most of the fast dynamics occur) was approximately 40. In this Crank-Nicolson solver, 2^{12} steps were taken. This corresponds to approximatly 40 steps per unit time, so it stands to reason that the approximation obtained using the Crank-Nicolson solver with 2^{12} steps would resember the approximation obtained using `ode15s` with absolute and relative tolerances of 10^{-10}.

3. [*10 points*] Consider the initial value problem in the previous problem. Use the MATLAB ODE solvers `ode45` and `ode15s` and two other MATLAB solvers of your choice to solve the system. Pass relative and absolute tolerances (`rtol` and `atol`, respectively) to the solvers as follows: use the same values for `atol` and `rtol`, and try tolerances of 10^{-6}, 10^{-8}, and 10^{-10}. Ask each of the solvers to report results at $10{,}000$ equally spaced points in the interval $[0, 100]$. Compare the results for the different solvers and for the difference tolerances. You have a lot of flexibility in reporting your findings, but make sure to comment on the solutions obtained with the given tolerances, number of the steps taken, and number of function evaluations. Provide representative plots as needed in your discussion. Also briefly comment on the performance of these methods as compared with that of the Crank–Nicolson method you implemented in the previous problem. Does your solution from the previous problem agree well at least visually with the solutions obtained in this problem? Can you suggest a simple strategy to speed up your Crank–Nicolson solver?

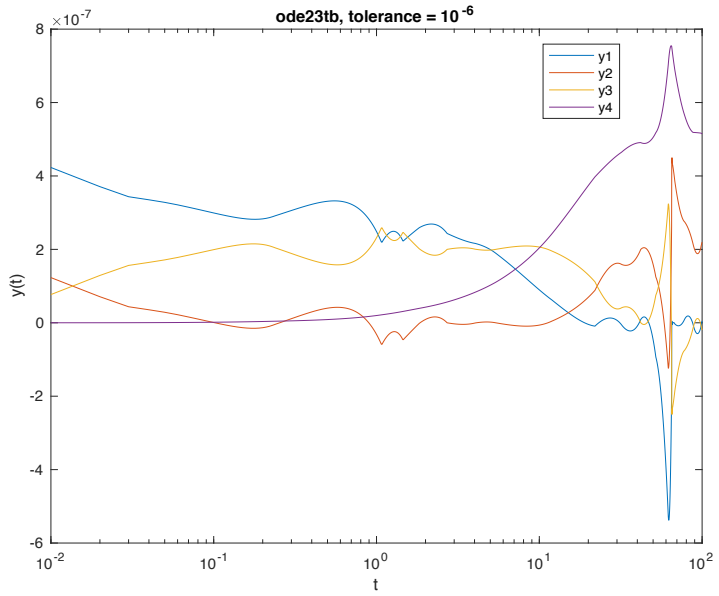
Answer:

In addition to `ode45` and `ode15s`, I used `ode23tb` and `ode23s`. All of the solvers except for `ode45` are billed as suited to solving stiff problems. In `ode45` with the tolerances equal to 10^{-6}, the solver did not finish, since it couldn't meet the tolerances without reducing the step size below the smallest value allowed.

With each solver, a sufficiently low tolerance was needed to obtain a reasonable looking solution vector. By a reasonable looking solution vector, I meaone which yields a plot that looks the same as plots produced by other solvers with sufficiently high tolerance. In addition, the reasonable plots resemble the plot reported in problem 2, generated using Crank–Nicolson with small step size. Such a "reasonable-looking plot" of the solution vector obtained using `ode15s` with tolerances of 10^{-8} is shown below.



For `ode25s`, `ode23tb`, and `ode23s`, tolerances of 10^{-8} were sufficient to product a reasonable solution vector, but for `ode45`, tolerances of 10^{-10} were necessary. As an example of a "unreasonable" looking plot, a plot of the approximation using `ode23tb` with tolerances of 10^{-6} is reported.



The number of function evaluations and the number of steps taken for each combination of solver and tolerances are reported in the following tables.

Number of function evaluations:

	10^{-6}	10^{-8}	10^{-10}
`ode45`	–	4573	3253
`ode15s`	140	116	268
`ode23tb`	137	139	276
`ode23s`	198	175	399

Number of steps taken:

	10^{-6}	10^{-8}	10^{-10}
`ode45`	–	593	530
`ode15s`	35	37	93
`ode23tb`	25	21	55
`ode23s`	26	23	54

Note that `ode45` both took more steps and had a larger number of function evaluations than the other methods did. This is becuase `ode45` is not suited to solving stiff problems, and thus it needed to take more steps in order to meet the tolerances.

These methods are much faster than the Crank–Nicolson method I implemented in the previous problem. Implementing all four methods with three times each (using the three different tolerances) took about 0.4 seconds, which is the same amount of time it took my Crank–Nicolson method to run once. However, with sufficiently small step size, my Crank–Nicolson solver was able to solve the IVP as well as the Matlab solvers. To make my Crank–Nicolson solver faster, I could increase the step size on the interval $[5, 100]$, where there are not fast dynamics, while leaving the step size on the interval $[0, 5)$ small.

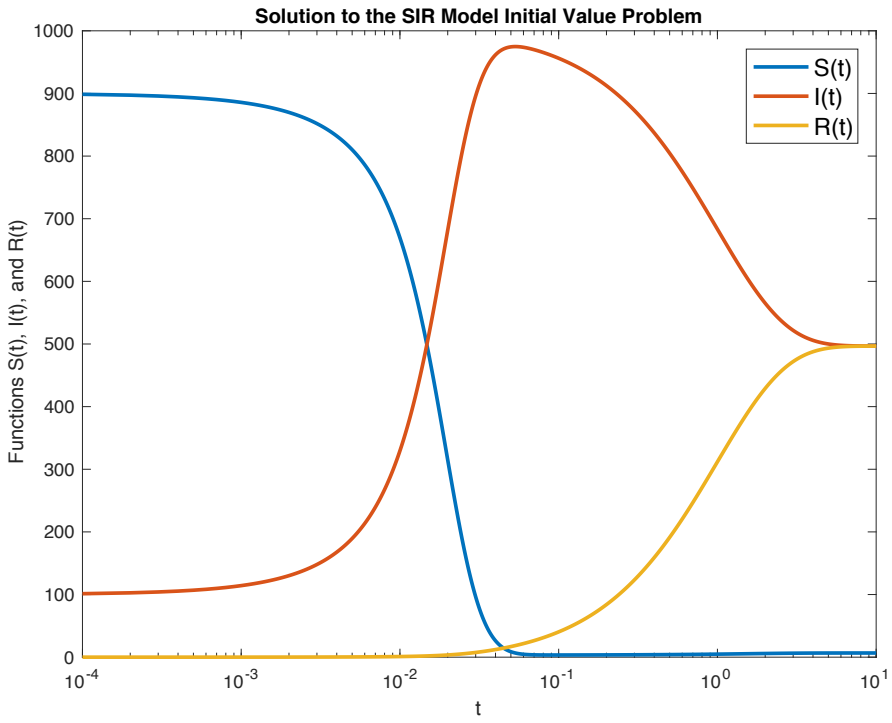
4. [*20 points*] The following ODE system describes a compartmental disease model of time evolution of susceptible, infected, and recovered populations in an environment. This is known as an SIR model.

$$\begin{aligned}\frac{dS}{dt} &= \delta(N - S) - \gamma IS,\\ \frac{dI}{dt} &= \gamma IS - (r + \delta)I,\\ \frac{dR}{dt} &= rI - \delta R.\end{aligned}$$

The initial state is given by $(S(0), I(0), R(0)) = (900, 100, 0)$, and $N = 1{,}000$ in the system. (This problem has been adapted from [3, Example 15.18.].)

(a) Solve the system using MATLAB's ode45 and compute and plot the solution on a uniform grid, with a reasonably small step size, in the interval $[0, 10]$. Use parameter values $(\delta, \gamma, r) = (0.5, .015, 0.5)$.

Answer:

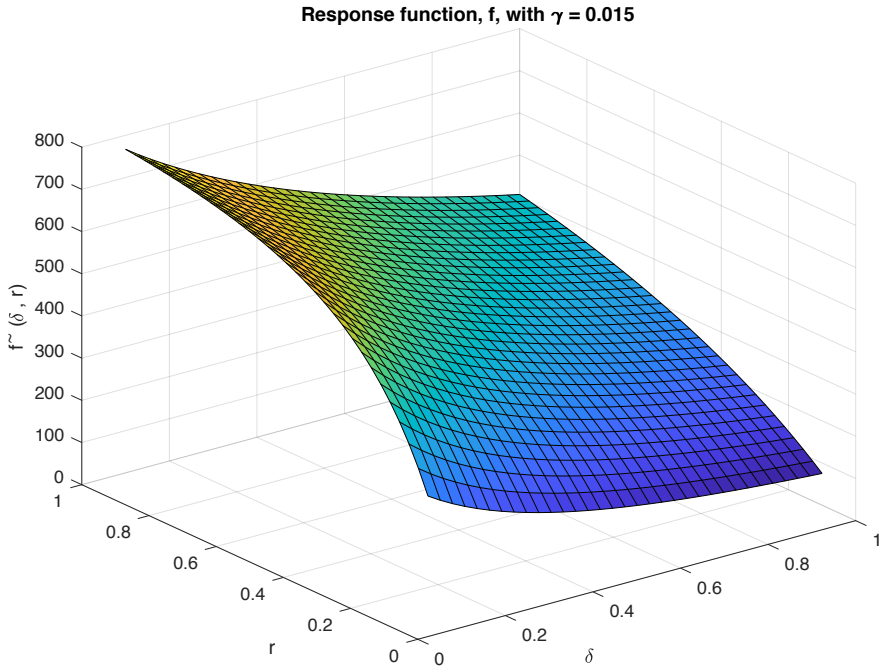


(b) Now suppose the parameters δ, γ, and r, each take values in the interval $[.01, 1]$; i.e., $(\delta, \gamma, r) \in X := [.01, 1]^3$. This way, the solution of the differential equation will depend on the choice of δ, γ, and r. Define the response function $f : X \to \mathbb{R}$ by,

$$f(\delta, \gamma, r) = \frac{1}{T} \int_0^T R(t; \delta, \gamma, r)\, dt, \quad T = 10.$$

Note that every evaluation of f requires solving the SIR model defined above. In your computations, compute the solution on a uniform grid with a reasonably small step size in the interval $[0, T]$, and compute the integral with composite trapezoidal rule, over your uniform time grid. Report a surface plot of $\tilde{f}(\delta, r) = f(\delta, 0.015, r)$.

Answer:



(c) Standardize the domain of the response function to $[-1,1]^3$ as follows. Find a simple change of variable $x \mapsto \alpha(x)$ that maps a point in $[-1,1]$ to $[.01,1]$, and then define,

$$g(\boldsymbol{x}) = f(\alpha(x_1), \alpha(x_2), \alpha(x_3)), \quad \boldsymbol{x} \in D.$$

Implement this via a MATLAB function that takes $\boldsymbol{x} \in D$ and evaluates $g(\boldsymbol{x})$.

Answer:

The following function $\alpha(x)$ maps a point in $[-1,1]$ to $[.01,1]$:

$$\alpha(x) = \frac{1-0.01}{2}x + \frac{1-0.01}{2} + 0.01 = \frac{0.99}{2}x + \frac{1.01}{2}$$

(d) Using the computer codes from the previous homework, compute an approximation of $g(\boldsymbol{x})$ using multivariate Legendre polynomials. Compute approximations φ_n of g, with orders $n = 1, \ldots, 6$. To examine the accuracy of the approximations, compute the relative L_2 error, $E_n := \|g - \varphi_n\|_2 / \|g\|_2$. (You can use an approach similar to the one in the previous homework.) Report a table and a plot (`semilogy`) showing the errors as you increase the order. Also, provide a plot of the coefficients of φ_n for $n = 6$. Comment on your results briefly.

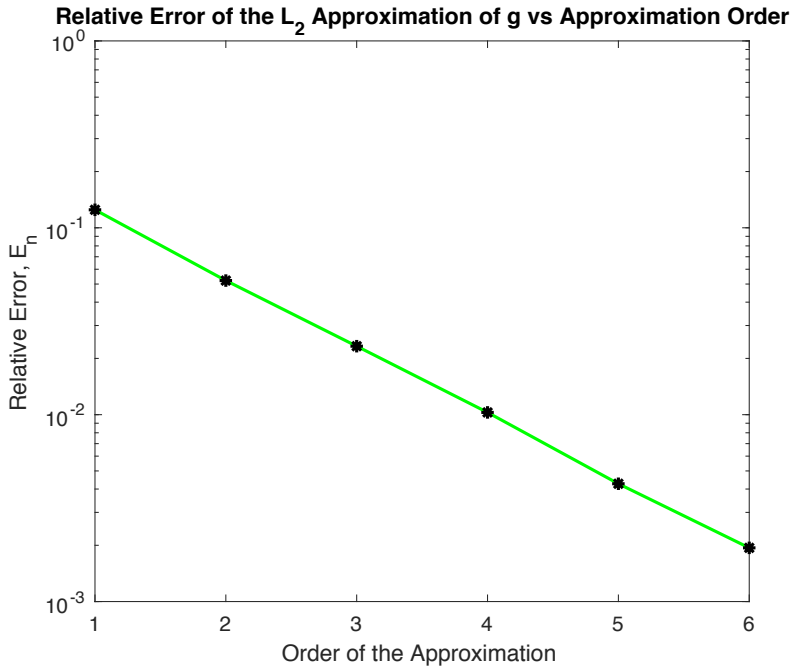
Note: this exercise combines the following topics covered throughout the course: numerical integration, ODE solvers, and approximation theory.

Answer:

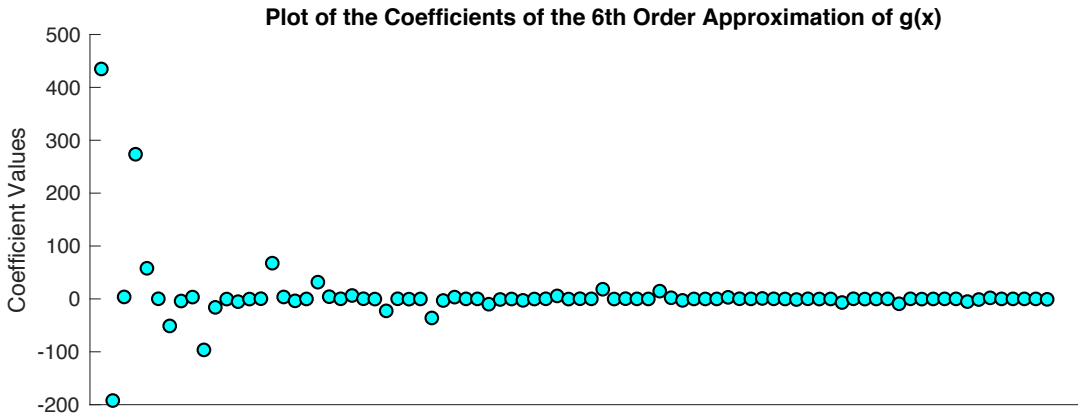
Table of the errors as the order of the L_2 approximation increases:

Order	E_n
1	1.2481×10^{-01}
2	5.2193×10^{-02}
3	2.3253×10^{-02}
4	1.0282×10^{-02}
5	4.2692×10^{-03}
6	1.9383×10^{-03}

Plot of the errors as the order of the L_2 approximation increases:



Plot of the coefficients of the 6th order approximation of $g(x)$:



The relative error of the least squared approxiation of $g(x)$ decreases logarithmically as n, the order of the approximation, increases. Computing the approximations took a fair amount of computational time, since evaluating g is computationally intensive and the number of evaulations of g has high for the higher order approximations.

5. [*10 points*] Consider the initial value problem $y' = f(x, y)$, $x \in [a, b]$, $y(a) = y_0$. The following is a three-stage explicit Runge–Kutta method for solving this IVP:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 4k_2 + k_3),$$
$$k_1 = f(x_n, y_n), \quad k_2 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1), \quad k_3 = f(x_n + h, y_n - hk_1 + 2hk_2).$$

(a) Write down the Butcher array for this method.

Answer:

$$\begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \\ -1 & 2 & 0 \end{bmatrix}$$

(b) By applying the method to the model ODE $y' = \lambda y$, $\lambda \in \mathbb{C}$, find the stability function of the method and define its region of absolute stability.

Answer:

$$\begin{aligned} y_{n+1} &= y_n + \frac{h}{6}(k_1 + 4k_2 + k_3) \\ &= y_n + \frac{h}{6}\Big[\lambda y_n + 4\lambda\Big(y_n + \frac{1}{2}h\lambda y_n\Big) + \lambda\Big(y_n - h\lambda y_n + 2h\lambda\Big(y_n + \frac{1}{2}h\lambda y_n\Big)\Big)\Big] \\ &= y_n\Big(1 + \frac{1}{6}z + \frac{4}{6}z + \frac{2}{6}z^2 + \frac{1}{6}z - \frac{1}{6}z^2 + \frac{2}{6}z^2 + \frac{1}{6}z^3\Big) \\ &= y_n\Big(\frac{1}{6}z^3 + \frac{1}{2}z^2 + z + 1\Big) \end{aligned}$$

Therefore the stability function $\varphi(z) = \frac{1}{6}z^3 + \frac{1}{2}z^2 + z + 1$. The region of absolute stability, D_A, is

$$D_A = \{z \in \mathbb{C} : \Big|\frac{1}{6}z^3 + \frac{1}{2}z^2 + z + 1\Big| < 1\}.$$

(c) Is this method A-stable? Explain briefly.

Answer:

This method is not A-stable. Consider $z = -3$.

$$|\varphi(-3)| = \Big|\frac{1}{6}(-3)^3 + \frac{1}{2}(-3)^2 + (-3) + 1\Big| = 2 \not< 1$$

Note that since h is a positive real number, $\mathrm{Re}(\lambda) < 0$. Therefore, since $|\varphi(z)| < 1$ does not hold for $z = -3$, the method is not A-stable.

6. [*10 points*] Derive the local truncation error of the following ODE scheme, used for solving an IVP, $y' = f(x, y)$, $x \in [a, b]$, $y(a) = y_0$. (note: here we consider a scalar ODE; i.e., $y(x) \in \mathbb{R}$.)

$$y_{n+1} = \frac{1}{2}(y_n + y_{n-1}) + \frac{h}{4}\big(4f(x_{n+1}, y_{n+1}) - f(x_n, y_n) + 3f(x_{n-1}, y_{n-1})\big).$$

Your answer should be in the following form:

$$T_n(y; h) = Ch^p y^{(q)}(x_n) + O(h^r).$$

What is the order of this method?

Answer:

Note that

$$T_n(y; h) = \frac{1}{h} T_n^L(y; h)$$

where

$$T_n^L(y; h) = y(x_{n+1}) - y_{n+1}.$$

and where $y(x_{n+1})$ denotes the exact solution to the IVP at x_{n+1} and y_{n+1} denotes the approximation of $y(x_{n+1})$ obtained using the above method.

Observe that

$$\begin{aligned} T_n^L(y; h) &= y(x_{n+1}) - \frac{1}{2}[y(x_n) + y(x_{n-1})] - \frac{h}{4}[4y'(x_{n+1}) - y'(x_n) + 3y'(x_{n-1})] \\ &= y(x_{n+1}) - \frac{1}{2}y(x_n) - \frac{1}{2}y(x_{n-1}) - hy'(x_{n+1}) + \frac{h}{4}y'(x_n) - \frac{3h}{4}y'(x_{n-1}) \end{aligned}$$

We simplify $T_n^L(y; h)$ by Taylor expanding $y(x_{n+1})$, $y(x_{n-1})$, $y'(x_{n+1})$, and $y'(x_{n-1})$ about x_n:

$$\begin{array}{rcccccccccc} T_n^L(y;h) & = & y(x_n) & +h & y'(x_n) & +\frac{h^2}{2} & y''(x_n) & +\frac{h^3}{6} & y'''(x_n) & + & O(h^4) \\ & -\frac{1}{2} & y(x_n) & & & & & & & & \\ & -\frac{1}{2} & y(x_n) & +\frac{h}{2} & y'(x_n) & -\frac{h^2}{4} & y''(x_n) & +\frac{h^3}{12} & y'''(x_n) & + & O(h^4) \\ & & & -h & y'(x_n) & -h^2 & y''(x_n) & -\frac{h^3}{2} & y'''(x_n) & + & O(h^4) \\ & & & +\frac{h}{4} & y'(x_n) & & & & & & \\ & & & -\frac{3h}{4} & y'(x_n) & +\frac{3h^2}{4} & y''(x_n) & -\frac{3h^3}{8} & y'''(x_n) & + & O(h^4) \\ \hline & = & & & & & & -\frac{5}{8}h^3 & y'''(x_n) & + & O(h^4) \end{array}$$

Finally, $T_n(y; h) = \frac{1}{h} T_n^L(y; h) = -\frac{5}{8}h^2 y'''(x_n) + O(h^3)$.

The order of this method is 2.

References

[1] Gautschi, Walter. Numerical analysis. Springer Science & Business Media, 2011.

[2] C.T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, 1995.

[3] Ralph C. Smith. *Uncertainty Quantification: Theory, Implementation, and Applications*, volume 12. SIAM, 2013.