assignments

Monday, February 17, 2020 9:58 PM

Homework 2

Due (in class) Wed., March 4 / Print and submit your codes if any

Note: **You are expected to write your own ODE solvers for all problems.** Do not use the Matlab's canned ODE solvers (e.g., `ode45`) unless required in problem statements. If you need to find roots of a nonlinear equation (e.g., find roots of $f(y) = y - sin(y) = 0$), you may find Matlab subroutines `roots` or `fzero` useful. If you are interested in learning these algorithms in detail, search Newton-Raphson method or secant method in Google.

1. Problem 4.6 from the text. (Estimation for the maximum time step should be obtained from stability analysis. You are encouraged to check the validity of this estimate by experimenting with Δt.)

2. Problem 4.7 from the text. (Assume that your ODE of interest has a purely oscillatory exact solution: $y' = i\omega y$, $y(0) = 1$.)

3. Problem 4.12 from the text.

4. Problem 4.16 from the text.

5. Problem 4.21 from the text.

6. Problem 4.26 from the text.

7. A popular version of third-order Runge-Kutta schemes is the one with a so-called strong-stability preserving (SSP) property, which is well suited for time advancing hyperbolic PDEs. Check out the article below from the web, and analyze the SSP RK3 scheme presented in Eq. (4.2) in the paper. Consider a linear model problem in the form of $\frac{du}{dt} = L(u) = \lambda u$. Discuss in detail order of accuracy, stability condition for general complex-valued λ, and amplitude/phase errors for purely oscillatory solution ($\lambda = i\omega$).

   Gottlieb, S., Shu, C.W. and Tadmor, E., 2001. Strong stability-preserving high-order time discretization methods. SIAM review, 43(1), pp.89-112,

8. In a single plot, draw the stability boundary curves of the following ODE solvers on the $\lambda_R h - \lambda_I h$ plane (use different colors): explicit Euler, RK2, RK3, RK4, second-order Adams-Bashforth method, and leapfrog. You should first identify points defining the stability boundary curves from the condition $|\sigma| = 1$ (Hint: read section 4.8 of text). For each scheme, present intersections of the stability boundary curves with real and imaginary axes (e.g., RK3 [-2.51, $1.75i$]).

4.6 Set up finding integrating factor

Wednesday, February 19, 2020 5:42 PM

6. A physical phenomena is governed by a simple differential equation:

$$\frac{dv}{dt} = -\alpha(t)v + \beta(t),$$

where

$$\alpha(t) = \frac{3t}{(1+t)} \qquad \beta(t) = 2(1+t)^3 e^{-t}.$$

Assume an initial value $v(0) = 1.0$, and solve the equation for $0 < t < 15$ using the following numerical methods

(a) Euler
(b) Backward Euler
(c) Trapezoidal method
(d) Second-order Runge–Kutta
(e) Fourth-order Runge–Kutta

Try time steps, $h = 0.2, 0.8, 1.1$. On separate plots, compare your results with the exact solution. Discuss the accuracy and stability of each method. For each scheme, *estimate* the maximum Δt for stable solution (over the given time domain *and* over a very long time).

$\frac{dv}{dt} = -\alpha(t)v + \beta(t)$ of the form of a linear ODE so we can multiply by the integrator and by product rule will get the following

$\frac{dv}{dt} + \alpha(t)v = \beta(t)$

$v \cdot e^{\int \alpha(t)dt} = \int e^{\int \alpha(t)dt} \cdot \beta(t)dt$

$\int \alpha(t)dt = \int \frac{3t}{(1+t)} dt$ $\quad u = 1+t \quad |du = dt \quad \frac{du}{dt} = 1$

$u - 1 = t$

$\int \frac{3u-3}{u} du \rightarrow \int 3 - \frac{3}{u} du \rightarrow 3u - 3\ln(u) \rightarrow 3(1+t) - 3\ln(1+t) + C$

$v \cdot e^{3(1+t)-3\ln(1+t)+C} = \int e^{3(1+t)-3\ln(1+t)+C} \cdot 2(1+t)^3 \cdot e^{-t}$

Part 1 integrating right hand side

Thursday, February 27, 2020 9:32 AM

$$V \cdot e^{3(1+t)-3\ln(1+t)+C} = \int e^{3(1+t)-3\ln(1+t)+C} \cdot 2(1+t)^3 \cdot e^{-t}\,dt$$

$$C_1 V \cdot e^{3(1+t)-3\ln(1+t)} = \int e^{3+3t-3\ln(1+t)-t+C} \cdot 2(1+t)^3\,dt$$

$$\int \frac{e^{3+2t+C}}{e^{\ln(1+t)^3}} \cdot 2(1+t)^3\,dt \rightarrow \int \frac{e^{3+2t+C}}{(1+t)^3} \cdot 2(1+t)^3\,dt \rightarrow \int 2e^{3+2t+C}\,dt$$

$u = 3+2t+C \quad \frac{du}{dt} = 2$

$\frac{1}{2}du = dt$

$$\int e^u\,du \rightarrow C_1 e^{3+2t} + C_2$$

$$e^u = e^{3+2t+C}$$

$$C_1 V \cdot e^{3(1+t)-3\ln(1+t)} = C_1 e^{3+2t} + C_2 \rightarrow V \cdot \frac{C_1 e^{2(1+t)}}{(1+t)^3} = C_1 e^{3+2t} + C \rightarrow$$

$$V = (1+t)^3 \cdot e^{3+2t-3-3t} + C \cdot e^{-3-3t} \cdot (1+t)^3$$

$$V = (1+t)^3 \cdot e^{-t} + C \cdot e^{(-3-3t)} \cdot (1+t)^3$$

Checking work

Thursday, February 27, 2020 2:41 PM

$$\frac{dv}{dt} = -\alpha(t)v + \beta(t),$$

$$\alpha(t) = \frac{3t}{(1+t)} \qquad \beta(t) = 2(1+t)^3 e^{-t}.$$

Exact Solution $V = (1+t)^3 \cdot e^{-t} + C \cdot e^{(-3-3t)} \cdot (1+t)^3$

$$\frac{d}{dx}(y) = -\left(\frac{3 \cdot x}{1+x}\right) \cdot y + \left(2 \cdot (1+x)^3 \cdot e^{-x}\right) \qquad y = \frac{e^{c_1+3+2x}\left(1+3x+3x^2+x^3\right) + c_1\left(1+3x+3x^2+x^3\right)}{e^{3+c_1+3x}}$$

$$V = (1+t)^3 \left(e^{-t} + C \cdot e^{(-3-3t)}\right)$$

$$\frac{dV}{dt} = 3(1+t)^2\left(e^{-t} + C \cdot e^{(-3-3t)}\right) - (1+t)^3\left(e^{-t} - 3Ce^{-3-3t}\right)$$

$$\frac{dV}{dt} = -\alpha \cdot V + \beta(t) = -\frac{3t}{(1+t)} \cdot V + \beta(t)$$

$$\frac{dV}{dt} = -3t(1+t)^2\left(e^{-t} + C \cdot e^{(-3-3t)}\right) + 2(1+t)^3 e^{-t}$$

finding C from initial conditions

Thursday, February 27, 2020 6:44 PM

$$V = (1+t)^3 \cdot e^{-t} + C \cdot e^{(-3-3t)} \cdot (1+t)^3$$

$$V(0) = 1$$

$$1 = (1+0)^3 \cdot e^{-0} + C \cdot e^{(-3-3\cdot 0)} \cdot (1+0)^3$$

$$1 = 1 + C \cdot e^{(-3)}$$

$$C = 0$$

Implicit euler

Thursday, February 27, 2020 7:40 PM

$$\frac{dv}{dt} = -\alpha(t)v + \beta(t),$$

$$\frac{dv}{dt} = \frac{-3t}{(1+t)} \cdot v + 2(1+t)^3 e^{-t}$$

$$\alpha(t) = \frac{3t}{(1+t)} \qquad \beta(t) = 2(1+t)^3 e^{-t}.$$

$$y_{n+1} = y_n + h\left(y_{n+1}, t_{n+1}\right)$$

$$y_{n+1} = y_n + h\left(\frac{-3t_{n+1}}{(1+t_{n+1})} \cdot y_{n+1} + 2(1+t_{n+1})^3 e^{-t_{n+1}}\right)$$

$$y_{n+1} = y_n + -h \cdot \frac{3t_{n+1}}{(1+t_{n+1})} \cdot y_{n+1} + h \cdot 2(1+t_{n+1})^3 e^{-t_{n+1}}$$

$$y_{n+1} + h \cdot \frac{3t_{n+1}}{(1+t_{n+1})} \cdot y_{n+1} = y_n + h \cdot 2(1+t_{n+1})^3 e^{-t_{n+1}}$$

$$y_{n+1}\left(1 + h \cdot \frac{3t_{n+1}}{(1+t_{n+1})}\right) = y_n + h \cdot 2(1+t_{n+1})^3 e^{-t_{n+1}}$$

$$y_{n+1} = \frac{y_n + h \cdot 2(1+t_{n+1})^3 e^{-t_{n+1}}}{\left(1 + h \cdot \frac{3t_{n+1}}{(1+t_{n+1})}\right)}$$

formula for implicit euler set this equal to your f

Trapezoidal Method

Friday, February 28, 2020 6:11 PM

$$y_{n+1} = y_n + \frac{h}{2}\left[f(y_{n+1}, t_{n+1}) + f(y_n, t_n)\right]$$

$$\frac{dv}{dt} = -\alpha(t)v + \beta(t),$$

$$\alpha(t) = \frac{3t}{(1+t)} \qquad \beta(t) = 2(1+t)^3 e^{-t}.$$

$$y_{n+1} = y_n + \frac{h}{2}\left[\frac{-3\cdot\epsilon_{n+1}}{1+\epsilon_{n+1}}\cdot y_{n+1} + 2(1+\epsilon_{n+1})^3 e^{-\epsilon_{n+1}} + \frac{-3\cdot\epsilon_n}{1+\epsilon_n}\cdot y_n + 2(1+\epsilon_n)^3 e^{-\epsilon_n}\right]$$

$$y_{n+1} = y_n + \frac{h}{2}\cdot\frac{-3\cdot\epsilon_{n+1}}{1+\epsilon_{n+1}}\cdot y_{n+1} + \frac{h}{2}2(1+\epsilon_{n+1})^3 e^{-\epsilon_{n+1}} + \frac{h}{2}\frac{-3\cdot\epsilon_n}{1+\epsilon_n}\cdot y_n + \frac{h}{2}2(1+\epsilon_n)^3 e^{-\epsilon_n}$$

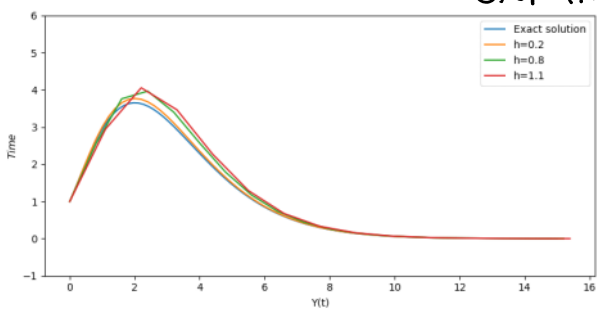
$$y_{n+1} + \frac{h}{2}\cdot\frac{3\cdot\epsilon_{n+1}}{1+\epsilon_{n+1}}\cdot y_{n+1} = y_n + \frac{h}{2}2(1+\epsilon_{n+1})^3 e^{-\epsilon_{n+1}} + \frac{h}{2}\frac{-3\cdot\epsilon_n}{1+\epsilon_n}\cdot y_n + \frac{h}{2}2(1+\epsilon_n)^3 e^{-\epsilon_n}$$

$$y_{n+1} = \frac{y_n + \frac{h}{2}2(1+\epsilon_{n+1})^3 e^{-\epsilon_{n+1}} + \frac{h}{2}\frac{-3\cdot\epsilon_n}{1+\epsilon_n}\cdot y_n + \frac{h}{2}2(1+\epsilon_n)^3 e^{-\epsilon_n}}{\left(1 + \frac{h}{2}\frac{3\cdot\epsilon_{n+1}}{1+\epsilon_{n+1}}\right)}$$

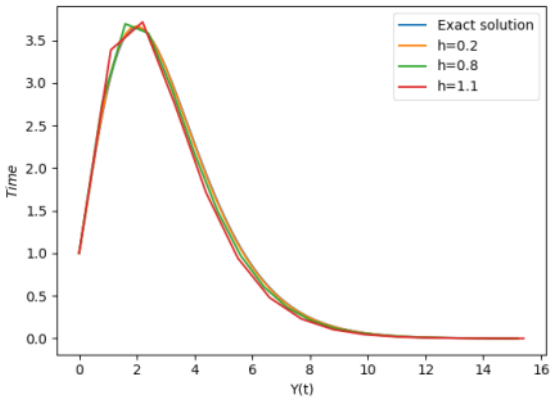
Graphs

Sunday, March 8, 2020 4:56 PM

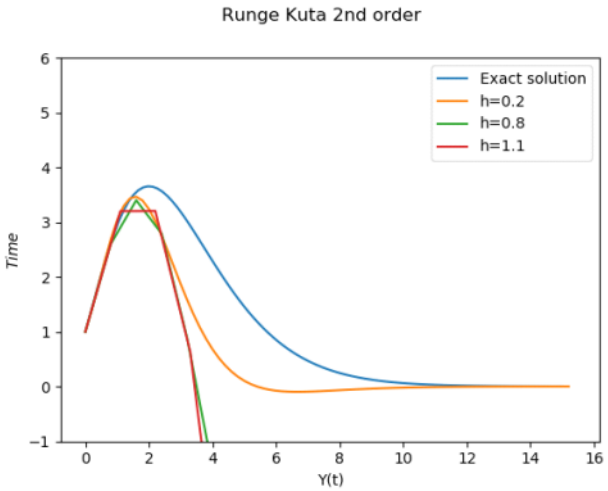
max stability step 1.1



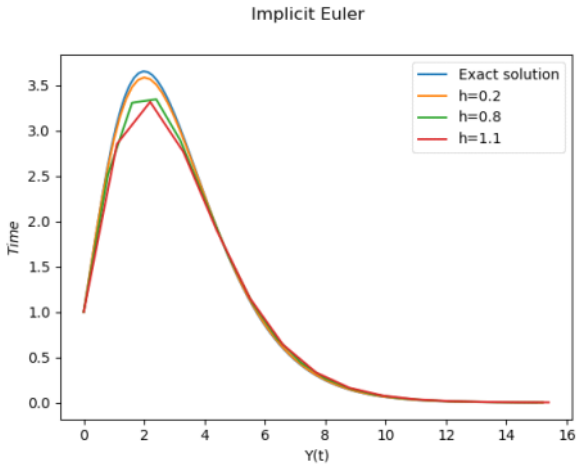
max stability step 1.1

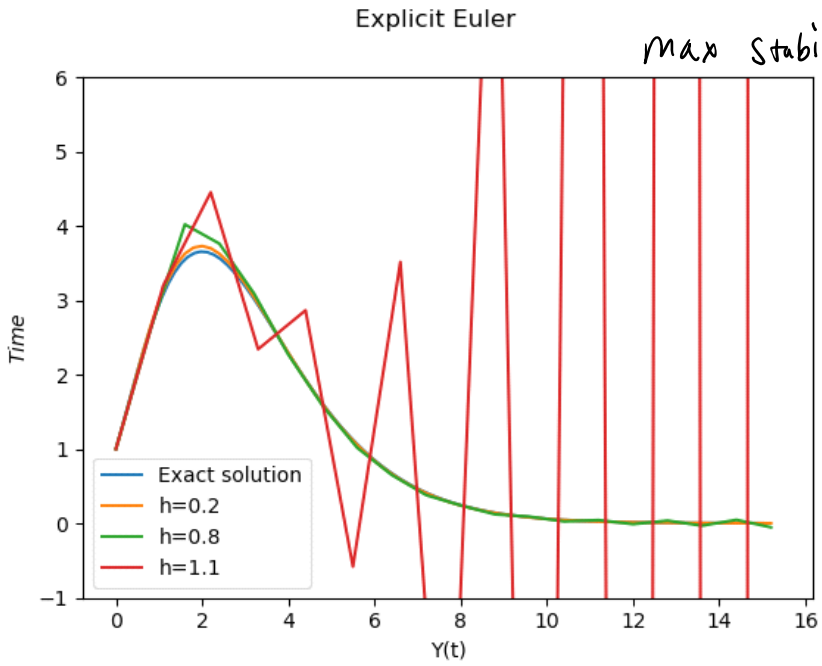


max stability 0.2



max stability 1.1





max stability 1.1

4.7

Wednesday, February 19, 2020 5:47 PM

7. *Choosing a method.*
The proper comparison of numerical methods should involve both the cost incurred as well as accuracy. Function evaluations are usually by far the most expensive part of the calculation. Let M be the total number of function evaluations allowed (reflecting a fixed computer budget) and suppose the calculation must reach time $t = T$. Given these two constraints the problem is to find the method that would maximize the accuracy (phase and amplitude) of the solution at time $t = T$. Occasionally, an additional constraint, that we do not consider here, related to storage requirements must also be included.

Note that a method which uses two evaluations/step must take $M/2$ steps of size $2h$ to reach T, in this case the expression for amplitude error is $1 - |\sigma|^{\frac{M}{2}}$ and the phase error is $\frac{M}{2}(2\omega h - \tan^{-1}(\frac{\sigma_I}{\sigma_R}))$. Let $T = 50$ and $\omega = 1$, plot these expressions for the following methods for M in the range 100–1000:

(i) Explicit Euler
(ii) RK2
(iii) RK4
(iv) Linearized trapezoidal
(v) Leapfrog

Which method would you most likely choose?

M = total number of function evaluations

T = t time you need to reach

Maximize accuracy phase and amplitude

	$\lvert\sigma\rvert$	PE	FE	step size	
Explicit E	$\sqrt{1+(\omega h)^2}$	ωh^2	1	M	h
RK2	$\sqrt{1+\frac{(\omega h)^4}{4}}$	$-\frac{\omega h^3}{6}$	2	$\frac{M}{2}$	$2h$
RK4	see 1	$\frac{(\omega h)^5}{120}$	4	$\frac{M}{4}$	$4h$
lin T	0	$\frac{\omega h^3}{12}$	3	$\frac{M}{3}$	$3h$
Leapfrog	0	$-\frac{\omega h^3}{6}$	1	M	h

1: $\left(0.00173611111111111h^8w^8 - 0.0138888888888889h^6w^6 - 2.22044604925031 \cdot 10^{-16}h^2w^2 + 1\right)^{0.5}$ rounding error

Runge kutta

Sunday, March 1, 2020 3:52 PM

$$K_1 = h\lambda y_n$$

$$K_2 = h(y_n + \tfrac{1}{2}K_1)\cdot\lambda = h(y_n + \tfrac{1}{2}K_1)\lambda = hy_n\lambda + \tfrac{1}{2}K_1\lambda = hy_n\lambda + \tfrac{1}{2}h^2y_n\lambda^2$$

$$K_3 = h(y_n + \tfrac{1}{2}K_2)\cdot\lambda = hy_n\lambda + \tfrac{1}{2}K_2\lambda \Rightarrow$$
$$= hy_n\lambda + \tfrac{1}{2}h^2y_n\lambda^2 + \tfrac{1}{4}h^3y_n\lambda^3$$

$$K_4 = h(y_n + K_3)\cdot\lambda = h\cdot y_n\cdot\lambda + K_3\lambda =$$
$$hy_n\cdot\lambda + h^2y_n\lambda^2 + \tfrac{1}{2}h^3y_n\lambda^3 + \tfrac{1}{4}h^4y_n\lambda^4$$

$$y_{n+1} = y_n + \frac{1}{6}k_1 + \frac{1}{3}(k_2 + k_3) + \frac{1}{6}k_4.$$

$$y_{n+1} = y_n + \tfrac{1}{6}(h\lambda y_n) + \tfrac{1}{3}\left(hy_n\lambda + \tfrac{1}{2}h^2y_n\lambda^2 + hy_n\lambda + \tfrac{1}{2}h^2y_n\lambda^2 + \tfrac{1}{4}h^3y_n\lambda^3\right)$$
$$+ \tfrac{1}{6}\left(hy_n\cdot\lambda + h^2y_n\lambda^2 + \tfrac{1}{2}h^3y_n\lambda^3 + \tfrac{1}{4}h^4y_n\lambda^4\right)$$
$$y_{n+1} = y_n\left(1 + \tfrac{1}{6}(h\lambda) + \tfrac{1}{3}\left(h\lambda + \tfrac{1}{2}h\lambda^2 + h\lambda + \tfrac{1}{2}h\lambda^2 + \tfrac{1}{4}h\lambda^3\right) + \left(h\lambda + h\lambda^2 + \tfrac{1}{2}h\lambda^3 + \tfrac{1}{4}h\lambda^4\right)\left(\frac{1}{60}\right)\right)$$

We plug in iw for λ and get an amplification of

$$\sqrt{0.00173611111111111h^8w^8 - 0.0138888888888889h^6w^6 - 2.2204460492503\cdot10^{-16}h^2w^2 + 1}$$

rounding error

$$\frac{-0.166666666666667h^3w^3 + 1.0hw}{0.0416666666666667h^4w^4 - 0.5h^2w^2 + 1} = Wh$$

$A = -0.16666$

$B = 0.041666$

$$\frac{Ax^3 + x}{Bx^4 - \frac{1}{2}x^2 + 1} = x$$

$X = 0.82734$

$$Ax^3 + x = Bx^5 - \frac{1}{2}x^3 + 1$$

$$Bx^5 + \left(-\frac{1}{2} - A\right)x^3 - x + 1 = 0$$

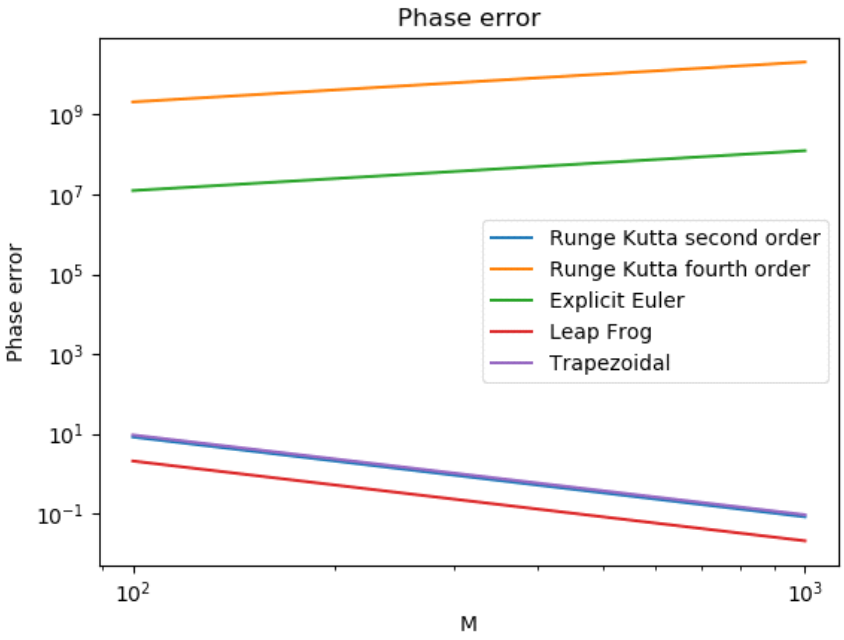
$$\frac{x^3}{1 - x^4}$$

$$Wh\ \frac{(Ax^2 + 1)}{Bx^4 - \frac{1}{2}x^2 + 1}$$

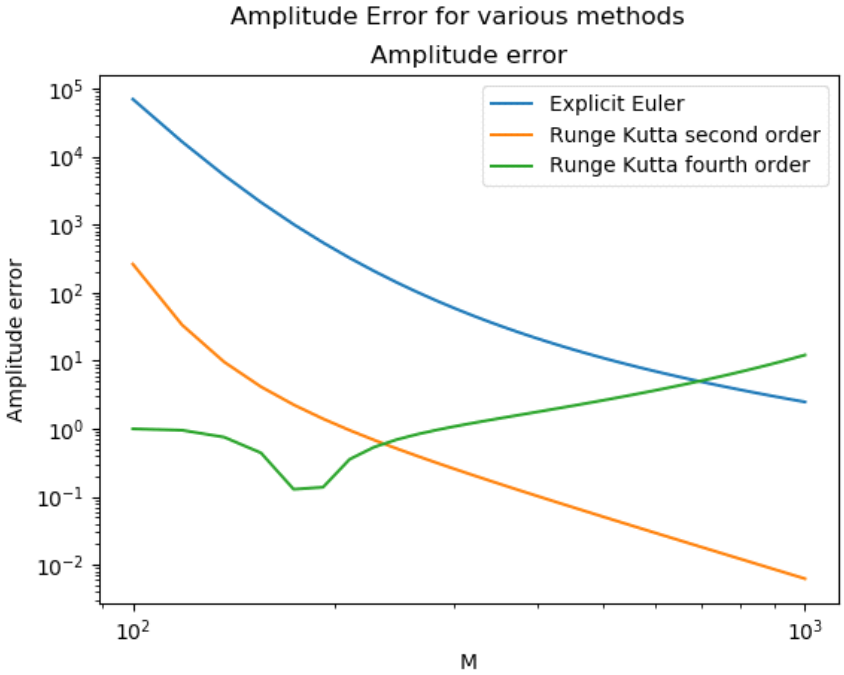
We Consider $\lambda = L = iw$

$$\frac{1}{4}w^4h - \frac{2}{3}iw^3h - \frac{7}{6}w^2h + \frac{11}{6}iwh + 1$$

$$\left(1 + \frac{1}{4}w^4h - \frac{7}{6}w^2h\right)^2 - \left(\frac{2}{3}w^3h + \frac{11}{6}wh\right.$$



Which I use would depend on the problem if I had an osccilating system I would use leap frog but if needed something fairly fast and accurate for a non oscilatin system. I would use RK4. While if I had a only a few ODE's and I could easily take their Derivatives I would use trapezoidal.



4.12

Wednesday, February 19, 2020 5:48 PM

12. *Fully implicit vs. linearized implicit.*
Consider the ODE

$$\frac{dy}{dt} = e^{y-t} \qquad y(0) = y_0.$$

Its analytical solution is

$$y(t) = -\ln(e^{-y_0} + e^{-t} - 1).$$

(a) Derive the linearized implicit Euler scheme.
(b) Use the analytic solution to derive exact expressions for the leading terms in the time discretization error and the linearization error.
(c) For $y_0 = -1 \times 10^{-5}$ and $h = 0.2$, plot the errors. Solve the system using the fully implicit and linearized implicit methods and plot their solutions against the analytical solution.
(d) Repeat part (c) with $y_0 = -1$. Comment on the sensitivity of the linearized solution to the initial condition.

a) $y_{n+1} = y_n + h f(y_{n+1}, t_{n+1}) + O(h^2)$ only globally first order

take a Taylor approx about (y_n, t_{n+1})

$$f(y_{n+1}, t_{n+1}) = f(y_n, t_{n+1}) + (y_{n+1} - y_n) \cdot f'(y_n, t_{n+1})$$

$$y_{n+1} = y_n + h \cdot \left(f(y_n, t_{n+1}) + (y_{n+1} - y_n) \cdot f'(y_n, t_{n+1}) \right)$$

$$y_{n+1} = y_n + h \cdot f(y_n, t_{n+1}) + h f'(y_n, t_{n+1}) \cdot y_{n+1} - h y_n f'(y_n, t_{n+1})$$

$$y_{n+1} - h f'(y_n, t_{n+1}) y_{n+1} = y_n + h \cdot f(y_n, t_{n+1}) - h y_n f'(y_n, t_{n+1})$$

$$y_{n+1} = \frac{y_n + h \cdot f(y_n, t_{n+1}) - h y_n f'(y_n, t_{n+1})}{(1 - h f'(y_n, t_{n+1})}$$

$$f'(y_n, t_{n+1}) = -e^{y-t}$$

Central difference justification

Wednesday, March 4, 2020 10:13 PM

Construct a finite difference scheme of the second order

$$f_j' + \sum_{j=-1}^{i=1} a_i \cdot f_i = O(h^2)$$

	f_j	f_j'	f_j''	f_j'''	f_j''''
f_j'	0	1	0	0	0
$a_0 f_j$	1	0	0	0	0
$a_1 f_{j-1}$	1	$-h$	$\frac{h^2}{2}$	$\frac{h^3}{6}$	$\frac{h^4}{24}$
$a_2 f_{j+1}$	1	h	$\frac{h^2}{2}$	$\frac{h^3}{6}$	$\frac{h^4}{24}$

$$(a_0 + a_1 + a_2) f_j + (1 - a_1 h + a_2 h) \cdot f_j' + (0 + a_1 \frac{h^2}{2} + a_2 \cdot \frac{h^2}{2}) f_j'' \quad O(h^3)$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & -h & h \\ 0 & \frac{h^2}{2} & \frac{h^2}{2} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$$

$$f_j' + f_j + f_{j-1} \quad f_{j+1} = O(h^3)$$

$$f_j' = \frac{f_{j+1} - f_{j-1}}{2h}$$

This construction is for $\frac{\partial f}{\partial y}\Big|_{y_n, t_{n+1}}$ to we must evaluate f before and after y_n keeping t_{n+1} constant this way the function ends to

$$y_{n+1} = \frac{y_n + h \cdot f(y_n, t_{n+1}) - h y_n f'(y_n, t_{n+1})}{(1 - h f'(y_n, t_{n+1}))}$$

h_y can be a different value than h

$$y_{n+1} = \frac{y_n + h \cdot f(y_n, t_{n+1}) - h y_n \cdot \left(\frac{f(y_n + h_y, t_{n+1}) - f(y_n - h_y, t_{n+1})}{2 h_y} \right)}{\left(1 - h \cdot \left(\frac{f(y_n + h_y, t_{n+1}) - f(y_n - h_y, t_{n+1})}{2 h_y} \right)\right)}$$

Part b

Friday, March 6, 2020 9:22 PM

The term we threw away was $\frac{1}{2}(y_{n+1}-y_n)^2 \frac{\partial^2 f}{\partial y^2}$ but in the Explicit Euler case

$$\frac{h}{2}(y_{n+1}-y_n)^2 e^{y_n - t_{n+1}}$$

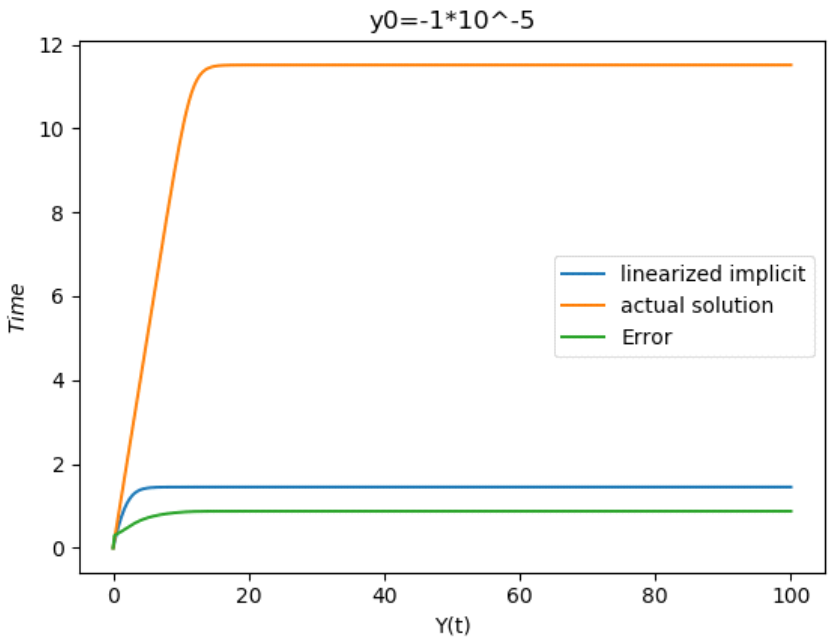
$$y_n = -\ln(e^{-y_0}+e^{-t_n}-1)$$

Part C IE = $y_{n+1} = y_n + f(y_{n+1}, t_{n+1})$

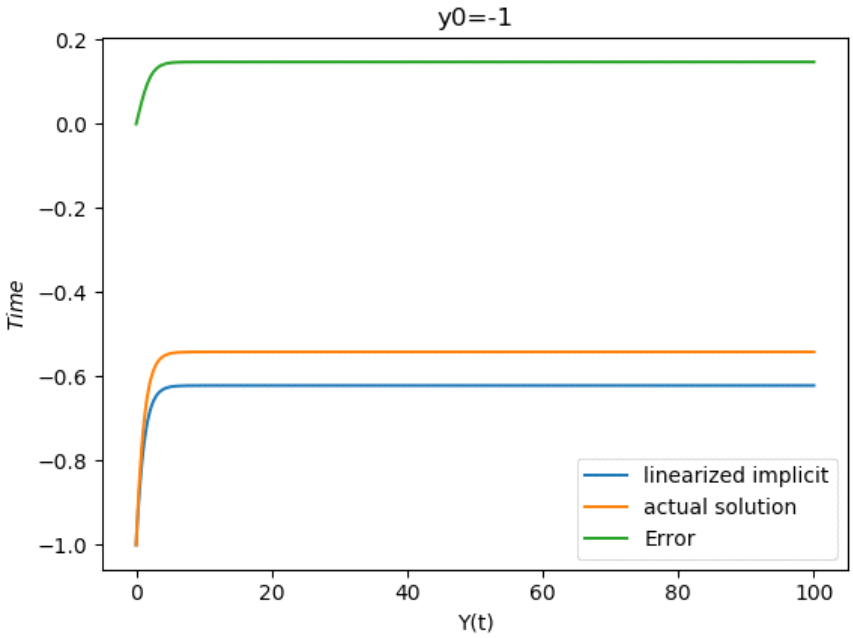
$$\frac{dy}{dt} = e^{y-t} \qquad y(0) = y_0.$$

$$y_{n+1} = y_n + e^{y_{n+1} - t_{n+1}}$$

$$0 = y_{n+1} + y_n + e^{(y_{n+1} - t_{n+1})}$$



The solution may very sensitive to initial condition. as can be seen in the plots.



4.16

Wednesday, February 19, 2020 5:48 PM

16. Non-linear differential equations with several degrees of freedom often exhibit chaotic solutions. Chaos is associated with sensitive dependence to initial conditions; however, numerical solutions are often confined to a so-called strange attractor, which attracts solutions resulting from different initial conditions to its vicinity in the phase space. It is the sensitive dependence on initial conditions that makes many physical systems (such as weather patterns) unpredictable, and it is the attractor that does not allow physical parameters to get out of hand (e.g., very high or low temperatures, etc.) An example of a strange attractor is the Lorenz attractor, which results from the solution of the following equations:

$$\frac{dx}{dt} = \sigma(y - x)$$
$$\frac{dy}{dt} = rx - y - xz$$
$$\frac{dz}{dt} = xy - bz.$$

The values of σ and b are usually fixed ($\sigma = 10$ and $b = 8/3$ in this problem) leaving r as the control parameter. For low values of r, the stable solutions are stationary. When r exceeds 24.74, the trajectories in xyz space become irregular orbits about two particular points.

(a) Solve these equations using $r = 20$. Start from point $(x, y, z) = (1, 1, 1)$, and plot the solution trajectory for $0 \leq t \leq 25$ in the xy, xz, and yz planes. Plot also x, y, and z versus t. Comment on your plots in terms of the previous discussion.

(b) Observe the change in the solution by repeating (a) for $r = 28$. In this case, plot also the trajectory of the solution in the three-dimensional xyz space (let the z axis be in the horizontal plane; you can use the MATLAB command `plot3(z,y,x)` for this). Compare your plots to (a).

(c) Observe the unpredictability at $r = 28$ by overplotting two solutions versus time starting from two initially nearby points: (6, 6, 6) and (6, 6.01, 6).

Part A

Wednesday, March 4, 2020 11:37 PM

$$\frac{dx}{dt} = \sigma(y - x)$$
$$\frac{dy}{dt} = rx - y - xz$$
$$\frac{dz}{dt} = xy - bz.$$

$r = 20$

$f_x(x, y, z, t) = 10y - 10x$

$f_y(x, y, z, t) = r \cdot x - y - xz$

$f_z(x, y, z, t) = x \cdot y - \frac{8}{3} z$

Start from $(1, 1, 1)$ $0 < t < 25$

To keep it simple lets use explicit euler

Pseudo code

$X = [1]$ $y = [1]$ $z = [1]$

$X_{n+1} = X_n + h \cdot f_x(X_n, y_n, z_n, t_n)$

$y_{n+1} = y_n + h \cdot f_y(X_n, y_n, z_n, t_n)$

$z_{n+1} = z_n + h \cdot f_z(X_n, y_n, z_n, t_n)$

$t_n = t_n + h$

append $X_{n+1}, y_{n+1}, z_{n+1}$ to their respective list

Rinse and Repeat.

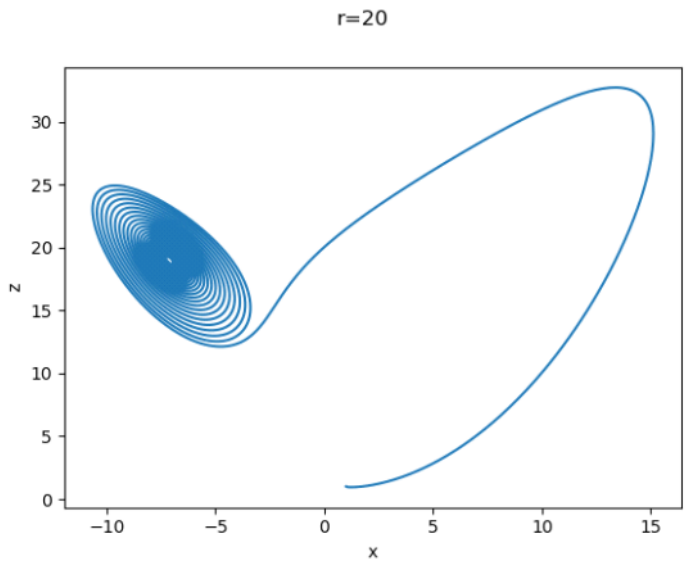
if we were to use RK4 I think you would add $\frac{1}{2} k_1$ to each variable in f_x, f_y, or f_z

$$y_{n+1} = y_n + \frac{1}{6}k_1 + \frac{1}{3}(k_2 + k_3) + \frac{1}{6}k_4. \quad (4.30a)$$

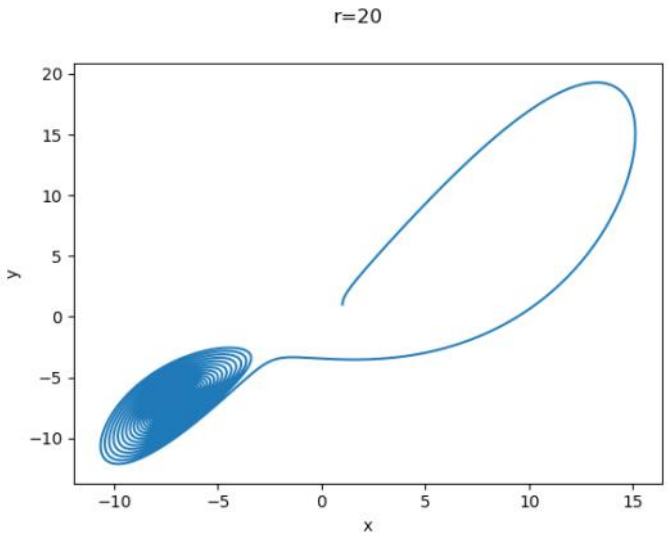
where

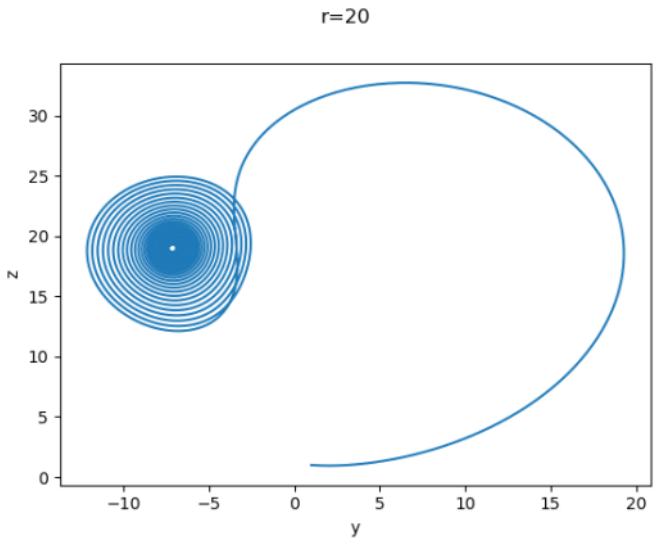
$$k_1 = hf(y_n, t_n) \quad (4.30b)$$
$$k_2 = hf\left(y_n + \frac{1}{2}k_1, t_n + \frac{h}{2}\right) \quad (4.30c)$$
$$k_3 = hf\left(y_n + \frac{1}{2}k_2, t_n + \frac{h}{2}\right) \quad (4.30d)$$
$$k_4 = hf(y_n + k_3, t_n + h). \quad (4.30e)$$

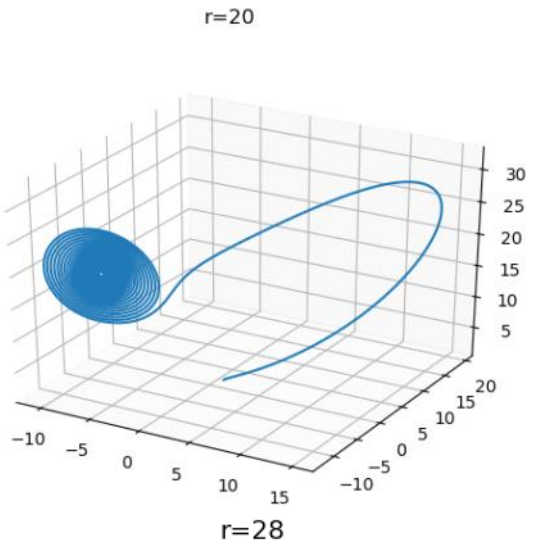
Sunday, March 8, 2020 5:11 PM



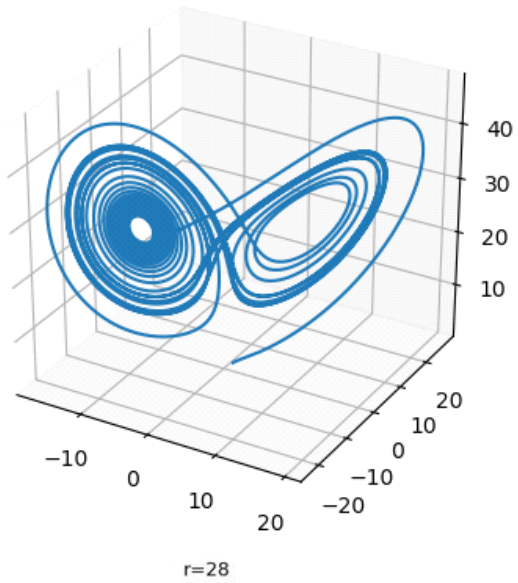
These look very chaotic and seem to center around a single point.

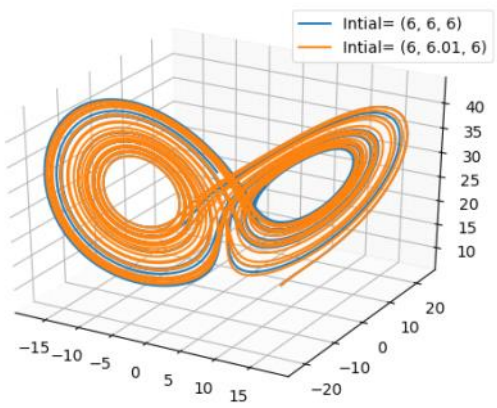






increasing r dramatically changed the solution.





the sensitivity to initial condition can be seen clearly in this over plot.

4.21

Wednesday, February 19, 2020 5:48 PM

21. Chemical reactions often give rise to stiff systems of coupled rate equations. The time history of a reaction of the following form:

$$\begin{aligned} &A_1 \rightarrow A_2 \\ &A_2 + A_3 \rightarrow A_1 + A_3 \\ &2A_2 \rightarrow 2A_3 \end{aligned}$$

is governed by the following rate equations

$$\begin{aligned} \dot{C}_1 &= -k_1 C_1 + k_2 C_2 C_3 \\ \dot{C}_2 &= k_1 C_1 - k_2 C_2 C_3 - 2k_3 C_2^2 \\ \dot{C}_3 &= 2k_3 C_2^2 \end{aligned}$$

where k_1, k_2, and k_3 are reaction rate constants given as

$$k_1 = 0.04, \qquad k_2 = 10.0, \qquad k_3 = 1.5 \times 10^3,$$

and the C_i are the concentrations of species A_i. Initially, $C_1(0) = 0.9$, $C_2(0) = 0.1$, and $C_3(0) = 0$.

(a) What is the analytical steady state solution? Note that these equations should conserve mass, that is, $C_1 + C_2 + C_3 = 1$.

(b) Evaluate the eigenvalues of the Jacobian matrix at $t = 0$. Is the problem stiff?

(c) Solve the given system to a steady state solution ($t = 3000$ represents steady state in this problem) using

(i) Fourth-order Runge–Kutta (use (b) to estimate the maximum time step).

(ii) A stiff solver such as *Numerical Recipes*' `stifbs`, `lsode`, or MATLAB's `ode23s`.

Make a log–log plot of the concentrations C_i vs. time. Compare the computer time required for these two methods.

(d) Set up the problem with a linearized trapezoidal method. What advantages would such a scheme have over fourth-order RK?

Part a and b

Thursday, March 5, 2020 1:01 PM

is governed by the following rate equations

$$\dot{C}_1 = -k_1 C_1 + k_2 C_2 C_3$$
$$\dot{C}_2 = k_1 C_1 - k_2 C_2 C_3 - 2k_3 C_2^2$$
$$\dot{C}_3 = 2k_3 C_2^2$$

where k_1, k_2, and k_3 are reaction rate constants given as

$$k_1 = 0.04, \qquad k_2 = 10.0, \qquad k_3 = 1.5 \times 10^3,$$

and the C_i are the concentrations of species A_i. Initially, $C_1(0) = 0.9$, $C_2(0) = 0.1$, and $C_3(0) = 0$.

(a) What is the analytical steady state solution? Note that these equations should conserve mass, that is, $C_1 + C_2 + C_3 = 1$.

(b) Evaluate the eigenvalues of the Jacobian matrix at $t = 0$. Is the problem stiff?

$0 = -K_1 C_1 + K_2 C_2 C_3$

$0 = K_1 C_1 - K_2 C_2 C_3 - 2K_3 C_2^2$

$0 = 2K_3 C_2^2$

$C_1 + C_2 + C_3 = 1$

to when the reaction stops this must be true $C_2 = 0$ to satisfy $0 = 2K_3 C_2$

$C_1 = 0$ to satisfy $0 = -K_1 C_1 + K_2 C_2 C_3 \Rightarrow 0 = -K_1 C_1$

$C_3 = 1$ to satisfy $C_1 + C_2 + C_3 = 1$

b) Jacobian =

$$\begin{bmatrix} \frac{dF_{C_1}}{C_1} & \frac{dF_{C_1}}{C_2} & \frac{dF_{C_1}}{C_3} \\ \frac{dF_{C_2}}{2C_1} & \frac{2F_{C_2}}{2C_2} & \frac{2F_{C_2}}{C_3} \\ \frac{2F_{C_3}}{2C_1} & \frac{2F_{C_3}}{2C_2} & \frac{2F_{C_3}}{2C_3} \end{bmatrix}$$

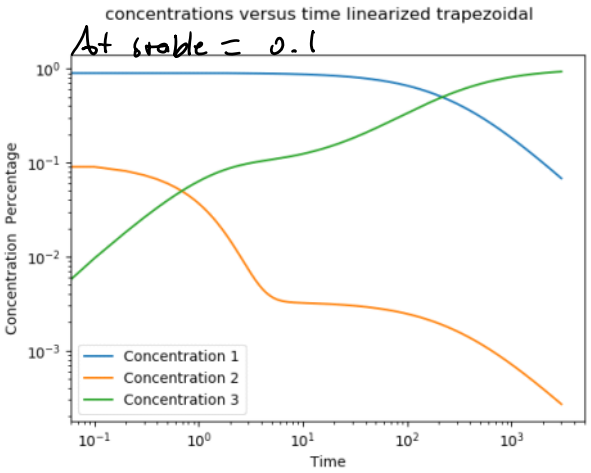
This is a stiff problem

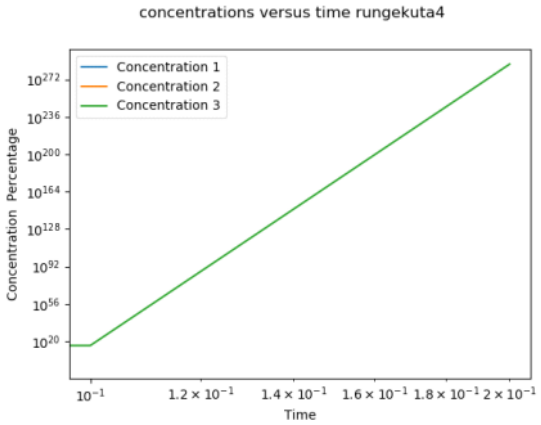
Jacobian =

$-K_1$	$K_2 C_3$	$K_2 C_2$
K_1	$-K_2 C_3 - 4K_3 C_2$	$-K_2 C_2$
0	$4K_3 C_2$	0

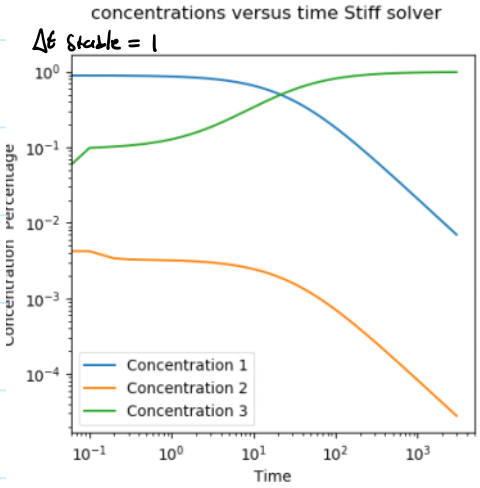
```
array([-5.98998261e+02,  1.77794334e-16, -1.0417
3925e+00])
```

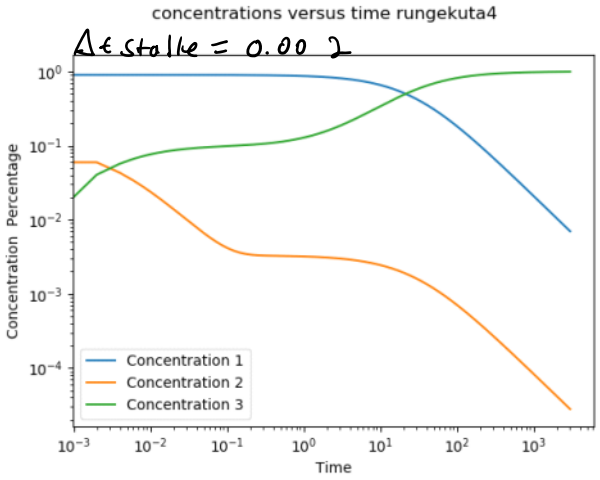
Thursday, March 5, 2020 6:27 PM





The stiff solver was far faster than the RK4 method and it required far larger steps to remain stable. The lin trav method was also a lot faster than RK4 it was also stable at much larger time steps

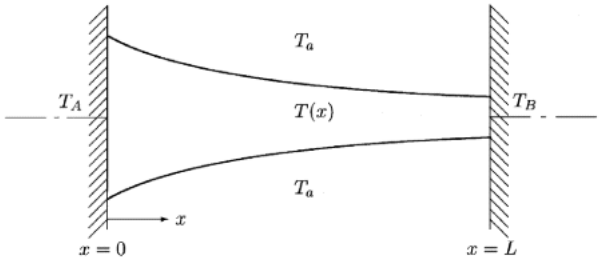




4.26

Wednesday, February 19, 2020 5:48 PM

The diagram shows a body of conical section fabricated from stainless steel immersed in air at a temperature $T_a = 0$. It is of circular cross section that varies with x. The large end is located at $x = 0$ and is held at temperature $T_A = 5$. The small end is located at $x = L = 2$ and is held at $T_B = 4$.



Conservation of energy can be used to develop a heat balance equation at any cross section of the body. When the body is not insulated along its length and the system is at a steady state, its temperature satisfies the following ODE:

$$\frac{d^2T}{dx^2} + a(x)\frac{dT}{dx} + b(x)T = f(x), \tag{1}$$

where $a(x)$, $b(x)$, and $f(x)$ are functions of the cross-sectional area, heat transfer coefficients, and the heat sinks inside the body. In the present example, they are given by

$$a(x) = -\frac{x+3}{x+1}, \qquad b(x) = \frac{x+3}{(x+1)^2}, \quad \text{and} \quad f(x) = 2(x+1) + 3b(x).$$

(a) In this part, we want to solve (1) using the shooting method.

(i) Convert the second-order differential equation (1) to a system of 2 first-order differential equations.

(ii) Use the shooting method to solve the system in (i). Plot the temperature distribution along the body.

(iii) If the body is insulated at the $x = L$ end, the boundary condition becomes $dT/dx = 0$. In this case use the shooting method to find $T(x)$ and in particular the temperature at $x = L$. Plot the temperature distribution along the body.

(b) We now want to solve (1) directly by approximating the derivatives with finite difference approximations. The interval from $x = 0$ to $x = L$ is discretized using N points (including the boundary points):

$$x_j = \frac{j-1}{N-1}L \quad j = 1, 2, \ldots, N.$$

The temperature at point j is denoted by T_j.

(i) Discretize the differential equation (1) using the central difference formulas for the second and first derivatives. The discretized equation is valid for $j = 2, 3, \ldots, N-1$ and therefore yields $N-2$ equations for the unknowns $T_1, T_2, \ldots, T_N$.

(ii) Obtain two additional equations from the boundary conditions ($T_A = 5$ and $T_B = 4$) and write the system of equations in matrix form $AT = f$. Solve this system with $N = 21$. Plot the temperature using symbols on the same plot of part (a)(ii).

Part A

Thursday, March 5, 2020 6:51 PM

$$\frac{d^2T}{dx^2} + a(x)\frac{dT}{dx} + b(x)T = f(x), \qquad (1)$$

where $a(x)$, $b(x)$, and $f(x)$ are functions of the cross-sectional area, heat transfer coefficients, and the heat sinks inside the body. In the present example, they are given by

$$a(x) = -\frac{x+3}{x+1}, \qquad b(x) = \frac{x+3}{(x+1)^2}, \quad \text{and} \quad f(x) = 2(x+1) + 3b(x).$$

Convert into a system of ODE's

$T_2 = T_1' = \frac{dT}{dx}$ $\quad T_1 = T$ $\quad T_1' = T_2$

$T_2' = -a(x)T_2 - b(x)T_1 + f(x)$

$\frac{d^2T}{dx} = \underbrace{-a(x)\frac{dT}{dx} - b(x)T + f(x)}_{g(x)} \rightarrow$

$T_2 = T_2 + h \cdot \frac{dT_2}{dx}(x, T_1, T_2)$

$T_1 = T_1 + h \cdot T_2$

2 queries for $T_0^{(1)}$

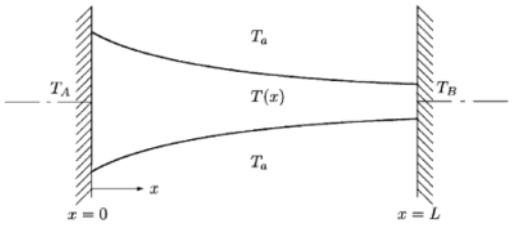
$T_0 = 5$ $\quad T_L =$

Guess 1 = G_1

Guess 2 = G_2

$$\begin{bmatrix} 1 & 1 \\ G_1 & G_2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 1 \\ T_B \end{bmatrix}$$

The diagram shows a body of conical section fabricated from stainless steel immersed in air at a temperature $T_a = 0$. It is of circular cross section that varies with x. The large end is located at $x = 0$ and is held at temperature $T_A = 5$. The small end is located at $x = L = 2$ and is held at $T_B = 4$.



$T_{(0)}^{(1)} = T_A = 5$ $\quad T_L = T_B = 4$

Guess 1 $T_0^{(2)} = -1$

Guess 2 $T_0^{(2)} = 1$

$T_n^{(2)} = T_n^{(2)} + h \cdot \frac{dT_a}{dx}(x, T_n^{(1)}, T_n^{(2)})$

$T_n^{(1)} = T_n^{(1)} + h \cdot T_n^{(2)}$

2 guesses for $T_0^{(1)}$

$T_0 = 5$ $\quad T_L =$

Guess 1 $= G_1$

Guess 2 $= G_2$

$$\begin{bmatrix} 1 & 1 \\ G_1 & G_2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 1 \\ T_B \end{bmatrix}$$

$$\frac{d^2T}{dx^2} + a(x)\frac{dT}{dx} + b(x)T = f(x), \qquad (1)$$

$$\frac{y_{j+1} - 2y_j + y_{j-1}}{h^2} + a(x)\frac{y_{j+1} - y_{j-1}}{2h} + b(x)\, y_j = f(x)$$

$$y_{j+1} - 2y_j + y_{j-1} + \frac{h}{2}a(x)(y_{j+1} - y_{j-1}) + b(x)\cdot h\, y_j = h^2 f(x)$$

$$y_{j+1} - 2y_j + y_{j-1} + \frac{h}{2}a(x)\, y_{j+1} - \frac{h}{2}a(x)\, y_{j-1} + b(x)\cdot h\, y_j = h^2 f(x)$$

$$-2y_j + \quad + \quad\quad + \quad = h^2 f(x)$$

$$\left(y_{j+1} + \frac{h}{2}a(x)\, y_{j+1}\right) + \left(y_{j-1} - \frac{h}{2}a(x)\, y_{j-1}\right) + b(x)\cdot h\, y_j - 2y_j = h^2 f(x)$$

$$a(x) = -\frac{x+3}{x+1}, \qquad b(x) = \frac{x+3}{(x+1)^2}, \quad \text{and} \quad f(x) = 2(x+1) + 3b(x).$$

Fill matrix and solve TDMA algorithm

$$y_{j-1}(A) + y_j(B) + y_{j+1}(C) = h^2 f(x)$$

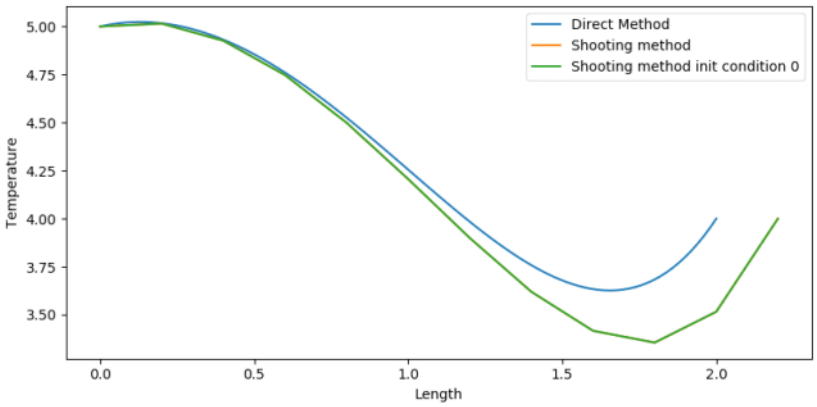
(1) $B y_1 + C\cdot y_2 = h^2 f(x) - y_0 \cdot A \longrightarrow 1y_1 + 0y_2 = T_A$

(2 : N-1) $y_{j-1}(A) + y_j(B) + y_{j+1}(C) = h^2 f(x)$

(N) $y_{N-2}A + y_{N-1}B = h^2 f(x) - y_N \cdot C \rightarrow 0y_{N-1} + y_{N-1} = T_B$

Graphs

Sunday, March 8, 2020 5:19 PM



Direct method solved ten system faster

Question 7

Sunday, March 1, 2020 9:13 PM

7. A popular version of third-order Runge-Kutta schemes is the one with a so-called strong-stability preserving (SSP) property, which is well suited for time advancing hyperbolic PDEs. Check out the article below from the web, and analyze the SSP RK3 scheme presented in Eq. (4.2) in the paper. Consider a linear model problem in the form of $\frac{du}{dt} = L(u) = \lambda u$. Discuss in detail order of accuracy, stability condition for general complex-valued λ, and amplitude/phase errors for purely oscillatory solution ($\lambda = i\omega$).

   Gottlieb, S., Shu, C.W. and Tadmor, E., 2001. Strong stability-preserving high-order time discretization methods. SIAM review, 43(1), pp.89-112,

with a CFL coefficient $c = 1$ in (2.10). An optimal third-order SSP Runge–Kutta method (2.9) is given by

$$
\begin{aligned}
u^{(1)} &= u^n + \Delta t L(u^n), \\
u^{(2)} &= \frac{3}{4}u^n + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t L(u^{(1)}), \\
u^{n+1} &= \frac{1}{3}u^n + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t L(u^{(2)}),
\end{aligned}
\tag{4.2}
$$

with a CFL coefficient $c = 1$ in (2.10).

$L(u) = \lambda u$

apply model equation

$u^{(1)} = y_n + h \cdot \lambda y_n$

$u^{(2)} = \frac{3}{4} y_n + \frac{1}{4}(y_n + h \cdot \lambda y_n) + \frac{1}{4} h\lambda (y_n + h \cdot \lambda y_n)$

$u^{n+1} = \frac{1}{3} y_n + \frac{2}{3}\left(\frac{3}{4} y_n + \frac{1}{4}(y_n + h \cdot \lambda y_n) + \frac{1}{4} h\lambda (y_n + h \cdot \lambda y_n)\right)$

$\frac{2}{3} h \cdot \lambda \left(\frac{3}{4} y_n + \frac{1}{4}(y_n + h \cdot \lambda y_n) + \frac{1}{4} h\lambda (y_n + h \cdot \lambda y_n)\right)$

$$u^{n+1} = \frac{1}{3} y_n + \frac{2}{3}\left(\frac{3}{4} y_n + \frac{1}{4}(y_n + h\cdot\lambda y_n) + \frac{1}{4}h\lambda(y_n + h\cdot\lambda y_n)\right) + \frac{2}{3}h\cdot\lambda\left(\frac{3}{4} y_n + \frac{1}{4}(y_n + h\cdot\lambda y_n) + \frac{1}{4}h(y_n + h\cdot\lambda y_n)\right)$$

$$y_n\left(\frac{1}{3} + \frac{2}{3}\left(\frac{3}{4} + \frac{1}{4}(1 + h\lambda) + \frac{1}{4}h\lambda(1 + h\cdot\lambda)\right) + \frac{2}{3}h\lambda\left(\frac{3}{4} + \frac{1}{4}(1 + h\cdot\lambda) + \frac{1}{4}h\lambda(1 + h\cdot\lambda)\right)\right)$$

$$y_n\left(\frac{1}{3} + \frac{2}{3}\left(\frac{3}{4} + \frac{1}{4}(1 + h\lambda) + \frac{1}{4}h\lambda(1 + h\cdot\lambda)\right) + \frac{2}{3}h\lambda\left(\frac{3}{4} + \frac{1}{4}(1 + h\cdot\lambda) + \frac{1}{4}h\lambda(1 + h\cdot\lambda)\right)\right)$$

$$0 = \left(\frac{1}{3} + \frac{2}{3}\left(\frac{3}{4} + \frac{1}{4}(1 + h\lambda) + \frac{1}{4}h\lambda(1 + h\cdot\lambda)\right) + \frac{2}{3}h\lambda\left(\frac{3}{4} + \frac{1}{4}(1 + h\cdot\lambda) + \frac{1}{4}h\lambda(1 + h\cdot\lambda)\right)\right)$$

$$0.166666666666667L^3h^3 + 0.5L^2h^2 + 1.0Lh + 1.0$$

3rd order accurate

Imaginary

$$LIh\left(-0.166666666666667LI^2h^2 + 0.5LR^2h^2 + 1.0LRh + 1.0\right)$$

Real:

$$-0.5LI^2LRh^3 - 0.5LI^2h^2 + 0.166666666666667LR^3h^3 + 0.5LR^2h^2 + 1.0LRh + 1.0$$

$$\sqrt{\lambda_{real}^2 + \lambda_{Imaginary}^2} = 1$$

Phase Error

$$hw - \frac{-0.166666666666667h^3w^3 + 1.0hw}{-0.5h^2w^2 + 1.0}$$

taylor expand $\frac{1}{1-x^2} = 1 + \frac{x}{2} + \frac{x^2}{4} + \ldots$

$$hw - \left(-0.166\,h^3w^3 + hw\right) = \frac{h^3w^3}{6}$$

has the exact same phase error as RK2

8

Thursday, March 5, 2020 12:09 AM

8. In a single plot, draw the stability boundary curves of the following ODE solvers on the $\lambda_R h - \lambda_I h$ plane (use different colors): explicit Euler, RK2, RK3, RK4, second-order Adams-Bashforth method, and leapfrog. You should first identify points defining the stability boundary curves from the condition $|\sigma| = 1$ (Hint: read section 4.8 of text). For each scheme, present intersections of the stability boundary curves with real and imaginary axes (e.g., RK3 [-2.51, $1.75i$]).

EE(|σ|) = $A = \sqrt{(1-\lambda_R h)^2 + \lambda_I^2 h^2}$,

RK2(|σ|) = $\sigma = \left(1 + \lambda h + \frac{\lambda^2 h^2}{2}\right) = 1 + (\lambda_R + i\lambda_I) + \frac{(\lambda_R + i\lambda_I)^2 h^2}{2}$

$$\sqrt{\left(1 + \lambda_R h + \frac{h^2\lambda_R^2}{2} - \frac{\lambda_I^2 h^2}{2}\right)^2 + \left(i\lambda_I h + 2i\lambda_I\lambda_R\frac{h^2}{2}\right)^2}$$

RK3(|σ|) = from Q 7

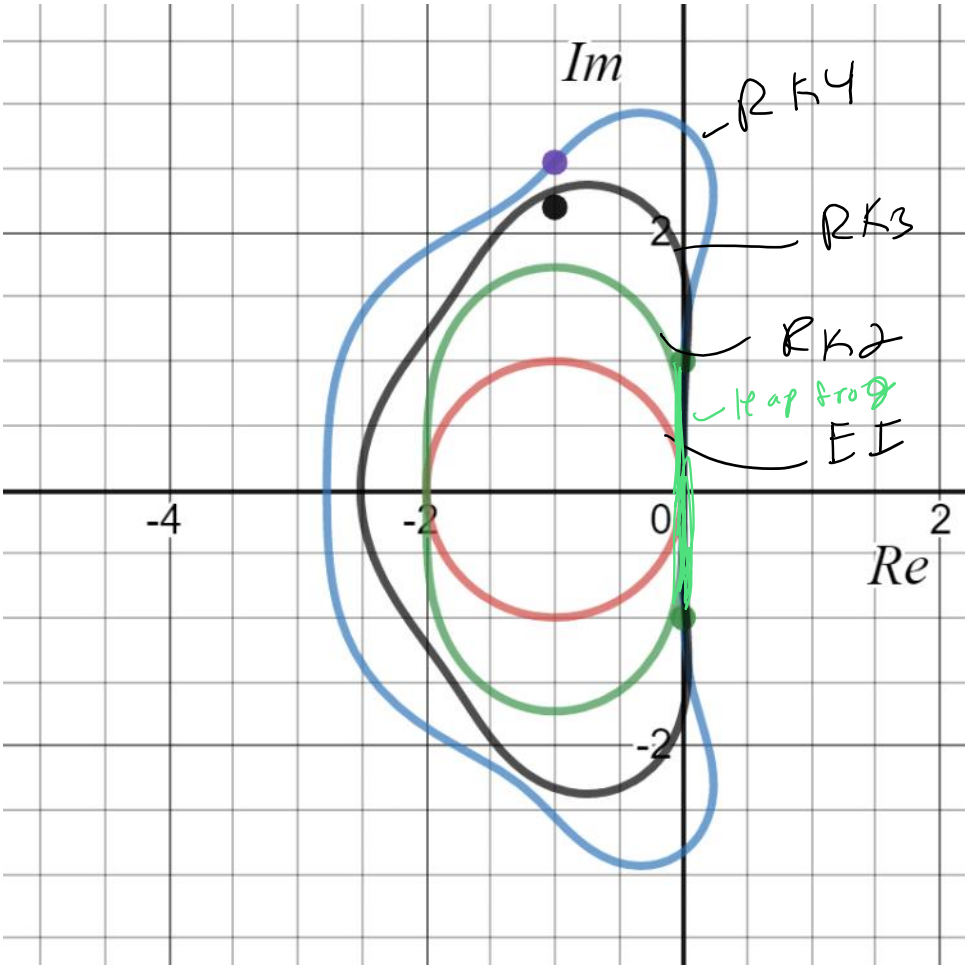
RK4(|σ|) = $\left(0.00173611111111111h^8w^8 - 0.0138888888888889h^6w^6 - 2.22044604925031 \cdot 10^{-16}h^2w^2 + 1\right)^{0.5}$

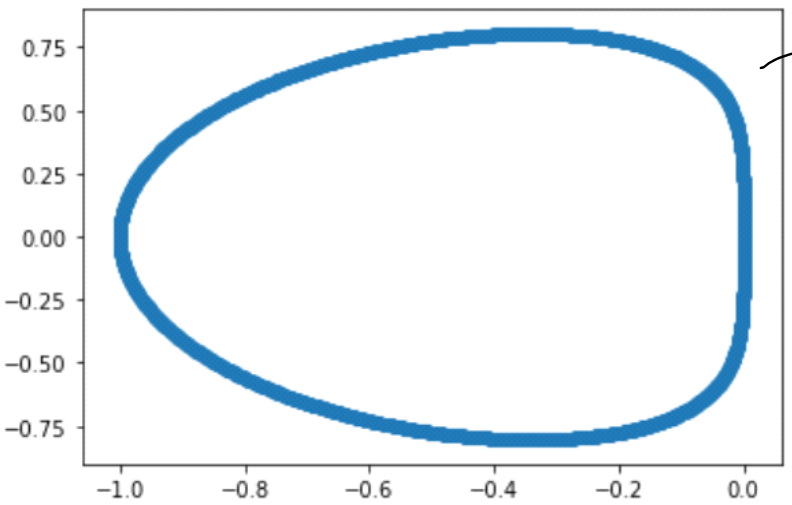
Adam Bashforth(σ) = $\frac{e^{i\theta} - 1}{\left(\frac{3}{2} - \frac{1}{2e^{i\theta}}\right)} \rightarrow \frac{\lambda_R + i\lambda_I - 1}{\frac{3}{2} - \frac{1}{2(\lambda_R + i\lambda_I)}}$

$$\frac{\lambda - 1}{\frac{3}{2} - \frac{1}{2\lambda}}$$

Leapfrog(σ) = $\sigma_1 = \lambda h + \sqrt{\lambda^2 h^2 + 1}$ If $|\omega h| \le 1$, then $\sigma_{1,2} = i\omega h \pm \sqrt{1 - \omega^2 h^2}$. $|\sigma_{1,2}| = 1$.

Run code labled Question 8 to see plot





→ Adam Bashforth

Jupyter notebooks

Sunday, March 8, 2020 5:59 PM

4.12

In [1]:
```
import numpy as np
import sympy as sym
from sympy import *
from IPython.display import display
x, y, z, t = symbols('x y z t')
init_printing(use_latex='mathjax')
```

In [7]:
```
h=symbols('h')
row1=[1,1,1,0]
row2=[0,-h,-2*h,-1]
row3=[0,(h**2)/2,2*(h**2),0]
A=Matrix([row1,row2,row3])
display(A)
S=solve_linear_system(A,'a0','a1','a2')
display(S)
```

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

```
{'a0': -3/(2*h), 'a1': 2/h, 'a2': -1/(2*h)}
```

In [ ]:
```
```

4.21

Sunday, March 8, 2020 6:00 PM

In [2]:
```
import numpy as np
import sympy as sym
from sympy import *
from IPython.display import display
x, y, z, t = symbols('x y z t')
init_printing(use_latex='mathjax')
```

In [14]:
```
k1=0.04
k2=10.0
k3=1.5*(10**3)
c1=0.9
c2=0.1
c3=0
row1=[-k1,k2*c3,k2*c2]
row2=[k1,-k2*c3-4*k3*c2,-k2*c2]
row3=[0,4*k3*c2,0]
A=np.array([row1,row2,row3])
eigval,eigvec=np.linalg.eigh(A)
display(eigval)
```

```
array([-9.70820394e+02, -3.99999998e-02,  3.70820394e+02])
```

In [ ]:
```
```

4.7

Sunday, March 8, 2020 6:01 PM

In [2]:
```
import numpy as np
import sympy as sym
from sympy import *
from IPython.display import display
x, y, z, t = symbols('x y z t')
init_printing(use_latex='mathjax')
```

In [9]:
```
a= symbols('LR', real=True)
b=symbols('LI', real=True)
h=symbols('h', real=True)
L=a+I*b

rk4=1+(1/6)*(L*h)+(1/3)*(h*L+(1/2)*h**2*L**2+h*L+(1/2)*h**2*L**2+(1/4)*h**3*L
reals=collect(rk4.expand(complex=True),I,evaluate=false)
t=reals[I]/reals[1]
print("Imaginary")
display(simplify(reals[I]))
print("Real:")
display(simplify(reals[1]))
#tani=t-((t**3)/3)+((t**5)/5)-((t**7)/7)+((t**9)/9)-((t**11)/11)

#display((simplify((reals[1]**2).expand()+(reals[I]**2).expand())**0.5))
#display(collect(simplify(tani),,evaluate=false))
```

```
Imaginary
```

$$LIh\left(-0.166666666666667LI^2LRh^3 - 0.166666666666667LI^2h^2 + 0.16666666666(\right.$$

```
Real:
```

$$0.0416666666666667LI^4h^4 - 0.25LI^2LR^2h^4 - 0.5LI^2LRh^3 - 0.5LI^2h^2 + 0.04$$
$$+ 0.166666666666667LR^3h^3 + 0.5LR^2h^2 + 1.0LRh + 1$$

In [ ]:
```
```

Question 7

Sunday, March 8, 2020 6:03 PM

In [2]:
```
import numpy as np
import sympy as sym
import matplotlib.pyplot as plt
from sympy import *
from IPython.display import display
x, y, z, t = symbols('x y z t')
init_printing(use_latex='mathjax')
```

In [5]:
```
a= symbols('LR', real=True)
b=symbols('LI', real=True)
w=symbols('w', real=True)
h=symbols('h', real=True)

#L=a
P1=((1/3)+(2/3)*((3/4)+(1/4)*(1+h*L)+(1/4)*h*L*(1+h*L)))
P2=(2/3)*h*L*((3/4)+(1/4)*(1+h*L)+(1/4)*h*L*(1+h*L))
rk3=simplify(P1+P2)
```

This is the amplification factor

$$y_{n+1} = \sigma^n * y_0$$

In [6]:
```
L=symbols('L', real=True)
print("amplification factor:")
display(rk3)
```

```
amplification factor:
```

$$0.166666666666667L^3h^3 + 0.5L^2h^2 + 1.0Lh + 1.0$$

Stability Region

In [7]:

```
L=a+I*b
P1=((1/3)+(2/3)*((3/4)+(1/4)*(1+h*L)+(1/4)*h*L*(1+h*L)))
P2=(2/3)*h*L*((3/4)+(1/4)*(1+h*L)+(1/4)*h*L*(1+h*L))
rk3=simplify(P1+P2)
reals=collect(rk3.expand(complex=True),I,evaluate=false)
print("Imaginary")
display(simplify(reals[I]))
#print(simplify(reals[I]))
print("Real:")
display(simplify(reals[1]))
#print(simplify(reals[1]))
```

```
Imaginary
```

$$LIh\left(-0.1666666666666667LI^2h^2 + 0.5LR^2h^2 + 1.0LRh + 1.0\right)$$

```
Real:
```

$$-0.5LI^2LRh^3 - 0.5LI^2h^2 + 0.1666666666666667LR^3h^3 + 0.5LR^2h^2 + 1.0LRh + 1.($$

$$\sqrt{\lambda_{real}^2 + \lambda_{Imaginary}^2} = 1$$

Is the stability region

In [50]:

```
L=I*w
P1=((1/3)+(2/3)*((3/4)+(1/4)*(1+h*L)+(1/4)*h*L*(1+h*L)))
P2=(2/3)*h*L*((3/4)+(1/4)*(1+h*L)+(1/4)*h*L*(1+h*L))
rk3=simplify(P1+P2)
reals=collect(rk3.expand(complex=True),I,evaluate=false)
t=reals[I]/reals[1]
display(t)
tani=t-((t**3)/3)+((t**5)/5)-((t**7)/7)+((t**9)/9)-((t**11)/11)
phase=w*h-tani

print("Phase Error")
display(phase)
```

$$\frac{-0.1666666666666667h^3w^3 + 1.0hw}{-0.5h^2w^2 + 1.0}$$

Phase Error

$$hw - \frac{-0.1666666666666667h^3w^3 + 1.0hw}{-0.5h^2w^2 + 1.0} + \frac{\left(-0.1666666666666667h^3w^3 + 1.0hw\right)^3}{3\left(-0.5h^2w^2 + 1.0\right)^3} - \frac{\left(-0.1666666666666667h^3w^3 + 1.0hw\right)^5}{5\left(-0.5h^2w^2 + 1.0\right)^5} + \frac{\left(-0.1666666666666667h^3w^3 + 1.0hw\right)^7}{7\left(-0.5h^2w^2 + 1.0\right)^7} - \frac{\left(-0.1666666666666667h^3w^3 + 1.0hw\right)^9}{9\left(-0.5h^2w^2 + 1.0\right)^9} + \frac{\left(-0.1666666666666667h^3w^3 + 1.0hw\right)^{11}}{11\left(-0.5h^2w^2 + 1.0\right)^{11}}$$

In [ ]:

Question 8

Sunday, March 8, 2020 6:03 PM

In [21]:
```
import numpy as np
import sympy as sym
import matplotlib.pyplot as plt
from sympy import *
from IPython.display import display
x, y, z, t = symbols('x y z t')
init_printing(use_latex='mathjax')
```

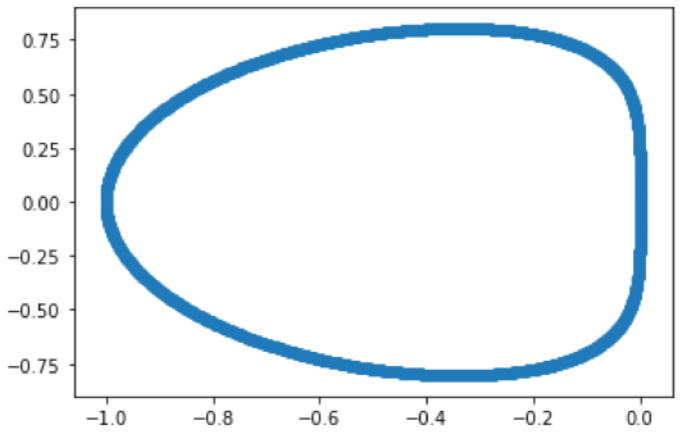
In [33]:
```
a= symbols('LR', real=True)
b=symbols('LI', real=True)
h=symbols('h', real=True)
L=a+I*b

rk4=1+(1/6)*(L*h)+(1/3)*(h*L+(1/2)*h**2*L**2+h*L+(1/2)*h**2*L**2+(1/4)*h**3*L
reals=collect(rk4.expand(complex=True),I,evaluate=false)
t=reals[I]/reals[1]
print("Imaginary")
#display(simplify(reals[I]))
print(simplify(reals[I]))
print("Real:")
#display(simplify(reals[1]))
print(simplify(reals[1]))
#tani=t-((t**3)/3)+((t**5)/5)-((t**7)/7)+((t**9)/9)-((t**11)/11)

#display((simplify((reals[1]**2).expand()+(reals[I]**2).expand())**0.5))
#display(collect(simplify(tani),,evaluate=false))
```

```
Imaginary
LI*h*(-0.166666666666667*LI**2*LR*h**3 - 0.166666666666667*LI**2*h**2 + 0.1
66666666666667*LR**3*h**3 + 0.5*LR**2*h**2 + 1.0*LR*h + 1.0)
Real:
0.0416666666666667*LI**4*h**4 - 0.25*LI**2*LR**2*h**4 - 0.5*LI**2*LR*h**3 -
0.5*LI**2*h**2 + 0.0416666666666667*LR**4*h**4 + 0.166666666666667*LR**3*h*
*3 + 0.5*LR**2*h**2 + 1.0*LR*h + 1
```

Adam Bashforth

In [31]:
```
t=np.linspace(0,2*np.pi,1000)
theta=0+1j*t
z=np.e**(theta)
Amp=(z-1)/((3/2)-(1/(2*z)))
plt.scatter(Amp.real,Amp.imag)
```

Out[31]: <matplotlib.collections.PathCollection at 0x233e91eed08>



In [ ]:

In [ ]: