ESC 407
Computational Methods in Engineering Science
Homework 6
Numerical Solution of Ordinary Differential Equations

Due Date: Friday, December 4 at 11:59 p.m.*

Problem 1 (35 pts)

Write three MATLAB functions to approximate the solutions to ordinary differential equation initial value problems of the form

$$\frac{d\vec{y}}{dt} = \dot{\vec{y}} = f(t, \vec{y}).$$

The first function, called `heun`, should implement Heun's method (without iteration), the second one, called `rk4`, should implement fourth-order Runge-Kutta, and the third one, called `rkf45`, should implement the Runge-Kutta Fehlberg algorithm that we discussed in class. The `heun` and `rk4` functions should take as inputs (in this order)

- the function f as a vector function handle,
- the time span over which the solution is to be approximated, as a row vector,
- the initial conditions $\vec{y}(0)$, as a row vector,
- the step size h.

The outputs from these functions should be

- a vector of the times t at which the solution has been approximated
- an array containing the approximations to the dependent variables $\vec{y}$.

For the `rkf45` function, the inputs should be (in this order)

- the function f as a vector function handle,
- the time span over which the solution is to be approximated, as a row vector,
- the initial conditions $\vec{y}(0)$, as a row vector,
- the error tolerance on the difference between the 4th- and 5th-order approximations,
- the maximum step-size allowed,
- the minimum step-size allowed.

The outputs from this function should be

- a vector of the times t at which the solution has been approximated

*You should definitely start on this assignment during Thanksgiving break! I can have office hours if needed.

- an array containing the approximations to the dependent variables $\vec{y}$,
- a vector containing the step-size used for each time step.*

So, for example, your `rk4` function might look like

```
function [t,y] = rk4(f,tspan,y0,h)
```

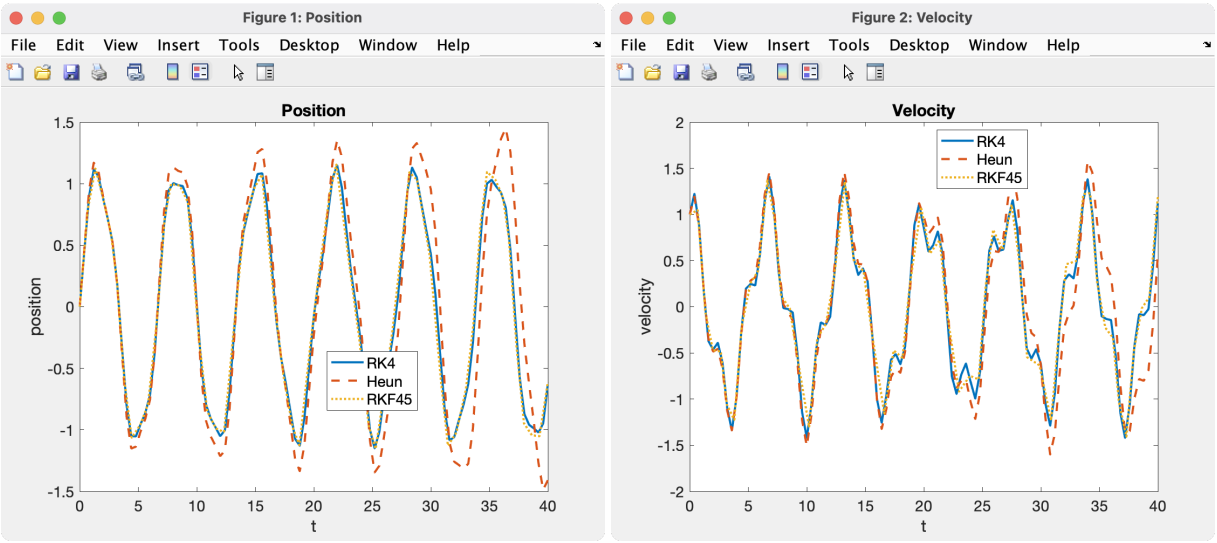
and your `rkf45` might look like

```
function [t,y,h] = rkf45(f,tspan,y0,tol,hmax,hmin)
```

Note that `heun` and `rk4` will only differ in the k's that are computed and in the increment function—they will otherwise be identical. No error checking is needed for these functions. To test your functions, solve

$$\ddot{y} + \sin y = \cos 3t, \quad y(0) = 0, \quad \dot{y}(0) = 1,$$

from $t = 0$ to $t = 40$. For `heun` and `rk4`, using a step-size of $h = 0.4$; and for `rkf45` using a tolerance of 1×10^{-3}, a minimum step-size of 0.4, and a maximum step-size of 2, your results should like this



Problem 2

(35 pts)

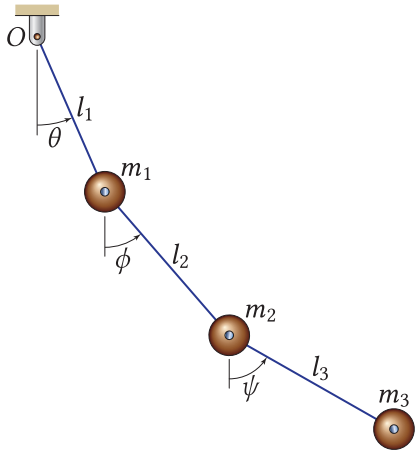
The equations of motion for the triple pendulum in Homework 5 can be written as

$$(m_1l_1 + m_2l_2 + m_3l_3)\ddot{\theta} + (m_2 + m_3)l_2\cos(\theta - \phi)\ddot{\phi} + m_3l_3\cos(\theta - \psi)\ddot{\psi} + (m_2 + m_3)l_2\sin(\theta - \phi)\dot{\phi}^2 + m_3l_3\sin(\theta - \psi)\dot{\psi}^2 + (m_1 + m_2 + m_3)g\sin\theta = 0 \quad (1)$$

$$(m_2 + m_3)l_1\cos(\theta - \phi)\ddot{\theta} + (m_2 + m_3)l_2\ddot{\phi} + m_3l_3\cos(\phi - \psi)\ddot{\psi} - (m_2 + m_3)l_1\sin(\theta - \phi)\dot{\theta}^2 + m_3l_3\sin(\phi - \psi)\dot{\psi}^2 + (m_2 + m_3)g\sin\phi = 0 \quad (2)$$

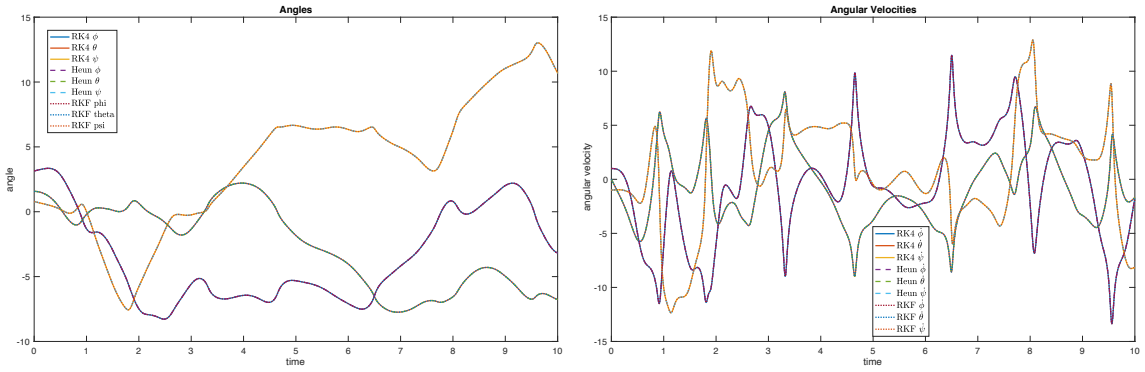
$$l_1\cos(\theta - \psi)\ddot{\theta} + l_2\cos(\phi - \psi)\ddot{\phi} + l_3\ddot{\psi} - l_1\sin(\theta - \psi)\dot{\theta}^2 - l_2\sin(\phi - \psi)\dot{\phi}^2 + g\sin\psi = 0 \quad (3)$$

*Of course, this could also be calculated from t.

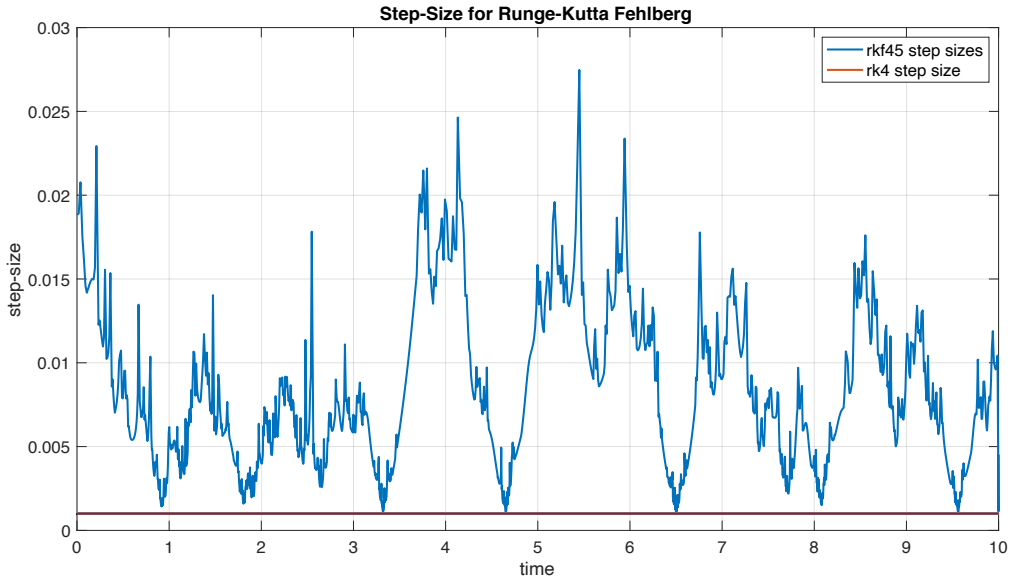


(a) Use MATLAB's `odeToVectorField` and `matlabFunction` commands, convert these equations to first-order form and solve them from $t = 0$ to $t = 10$ using all three of your two functions from Problem 1. Use the same initial conditions that you used in Homework 5. Use $l_1 = l_2 = l_3 = 1$ and $m_1 = 3$, $m_2 = 2$, and $m_3 = 1$. For each function, adjust the integration parameters until the three methods visually look the same over then entire time interval for both angles and angular velocities. For `heun` and `rk4`, this will mean reducing the step size, and for `rkf45` this will mean adjusting the error tolerance, minimum step-size, and maximum step-size. As you reduce the step-size for `heun` and `rk4`, keep in mind that the global error of each of the two methods is not the same, that is, you won't have to reduce the step-size as much for 4th-order Runge-Kutta as you will for Heun. In fact, using the global error of each, you should be able to figure out the ratio of the two step-sizes. It will be a little more difficult for the Runge-Kutta Fehlberg function, but you should be able to fairly quickly find parameters that work.

This is what your final result should look like.



The step-size needs to get pretty small for the Heun function, so be patient when executing it for small h. On my laptop, it took almost 70 seconds to execute all three functions and to plot the results. The Heun plot has *a lot* of points on it! Notice how much more efficient Runge-Kutta Fehlberg is compared to Heun's method, which is a second-order method, and fourth-order Runge-Kutta. To get agreement, I needed 10,000 steps in `fk4`, 10,000,000 in `heun` and only 1665 in `rkf45`. Plot h versus t to see the changing step-size in `rkf45`. Here is what I see when I do that:



So cool!

(b) Since it has a lot of points, use your fourth-order Runge-Kutta solution to plot the trajectories of masses m_1, m_2, and m_3 from $t = 0$ to $t = 10$ to see if they agree with what you found in Homework 5.*

(c) Use your fourth-order Runge-Kutta function to find the first time that $\psi = 2\pi$ to within 10^{-5} s.† Report the time at which the crossing occurs and the values of the dependent variables at that time. Make sure you allow ψ to approach 2π from either direction.

(d) Use MATLAB's `events` capability with MATLAB's `ode45` function to solve the same problem posed in Part (c). Again, make sure you allow ψ to approach 2π from either direction. Because these ODEs are so nonlinear, you will need to set more stringent tolerances on `AbsTol` and `RelTol` than the defaults for the time to agree with that found in Part (c).

Problem 3

(30 pts)

The Duffing equation, which is typically written as

$$\ddot{x} + \delta\dot{x} + \alpha x + \beta x^3 = \gamma \cos \omega t,$$

is a famous nonlinear differential equation that can exhibit a wide range of behaviors, depending on the parameters.‡ These constant parameters can be interpreted as follows:

*They should!

†**Hint:** Make new call to `rk4` for each time step of the solution.

‡See J. Guckenheimer and P. Holmes (1983) *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, Springer-Verlag, New York. For a simpler treatment, see F. C. Moon (1987) *Chaotic Vibrations: An Introduction for Applied Scientists and Engineers*, John Wiley & Sons, New York. For applications to spacecraft attitude dynamics, see G. L. Gray, I. Dobson, and D. C. Kammer (1996) "Chaos in a Spacecraft Attitude Maneuver Due to Time-Periodic Perturbations," *ASME Journal of Applied Mechanics*, **63**(2), pp. 501–508, and Daniel C. Kammer and Gary L. Gray (1993) "A Nonlinear Control Design for Energy Sink Simulation in the Euler-Poinsot Problem,"

- δ is the amount of linear viscous damping,
- α is the linear stiffness or restoring force,
- β is the amount of nonlinearity in stiffness or restoring force,
- γ is the amplitude of the periodic forcing,
- ω is the angular frequency of the periodic forcing.*

Our goal is to study some of the behaviors of this differential equation. For each of the following parts use $\delta = 0.3$, $\alpha = -1$, $\beta = 1$, and $\omega = 1.2$. For Parts (a)–(e) solve the equation using the specified γ, for $0 \leq t \leq 200$, and then do the following:

- Plot the response, i.e., x as a function of time.
- In the response plot, you will notice that it takes 30–40 s for the solution to settle down to a regular motion. This is the homogeneous solution dying out. The periodic solution after about 40 s is the particular solution. With this in mind, plot the *phase space*, that is, $\dot{x}$ versus x, for $40 \leq t \leq 200$.† Use `xlim` and `ylim` to set the limits on the axes.
- Plot the phase space only at times t that are multiples of the forcing period for $40 \leq t \leq 200$.‡ Use the same axis limits as you used for the phase space. This is called the *first return map* or *Poincaré map*. You should see that the number of clusters of points on the Poincaré maps in Parts (a)–(e) will equal the period of the oscillation.

I recommend using `subplot` for these to avoid a proliferation of windows.

(a) **Period-1 Oscillation.** Solve the equation using $\gamma = 0.2$.

(b) **Weak Period-2 Oscillation.** Solve the equation using $\gamma = 0.28$.

(c) **Strong Period-2 Oscillation.** Solve the equation using $\gamma = 0.65$.

(d) **Period-4 Oscillation.** Solve the equation using $\gamma = 0.29$.

(e) **Period-5 Oscillation.** Solve the equation using $\gamma = 0.37$.

(f) **Chaos!** Solve the equation using $\gamma = 0.5$. Now integrate from $0 \leq t \leq 20{,}000$ and do the following:

The Journal of the Astronautical Sciences, **41**(1), pp. 53–72. In grad school, I was taking a class on nonlinear dynamics and chaos and we had just looked at chaos in the Duffing equation, when I stumbled upon the paper J. L. Junkins, I. D. Jacobson, and J. N. Blanton (1973) "A Nonlinear Oscillator Analog of Rigid Body Dynamics," *Celestial Mechanics*, **7**, pp. 398–407, while reading the spacecraft dynamics literature to get research ideas. It was that class combined with that paper that give me the idea for my Ph.D. thesis.

*The *period* of the forcing is $\tau = 2\pi/\omega$.

†To start plotting at 40 s, you can simply divide 40 by your step-size h to determine where you should start in your solution, but that will likely not be an integer and when you use that number as an index, MATLAB will say `Warning: Integer operands are required for colon operator when used as index.` It will still work and you can ignore this warning, but it is not very elegant. This can be fixed by making that number an integer.

‡You will want to reference integer indices for times in the solution that are one period τ apart. To make this easier, I chose a step-size h for `rk4` such that τ/h is equal to an integer. You will still want h to be about the same size as that found in Problem 2.

- Plot the response for $0 \leq t \leq 250$.
- Plot the phase space for $40 \leq t \leq 250$. Use `xlim` and `ylim` to set the limits on the axes.
- Plot the Poincaré map for $40 \leq t \leq 20{,}000$. Use the same axis limits as you used for the phase space. The shape you see is called a *strange attractor*. It is the part of phase space to which all solutions are attracted in the presence of damping.

(g) **Flowing Poincaré Maps!** Using your solution from Part (f), use `subplot` to plot a 4×5 grid of 20 Poincaré maps, with the starting times $t = 40 + \tau/n$, $n = 1, 2, \ldots, 20$, where τ is the period of the forcing. You will see that the maps "flow" from one starting time to the next.

(h) 🦄 **Extra Credit (10%): Animated Poincaré Maps!** Use MATLAB's `getframe` and `movie` commands to animate the maps computed in Part (g) on a single plot. Go nuts and animate more than 20 maps!