PH5270 Numerical Methods and Programming Lab Assignment

Pranav Satheesh—PH17B008

July 13, 2020

Damped driven pendulum

For this assignment we are looking at the case of a damped pendulum subject to an external driving force. First off, we should write down the equation of motion for this system. This is fairly straight forward but let's begin with identifying and setting some values of parameters in this problem.

- Mass of the body $m = 1$
- Length of the string $l = 1$
- $\beta = \sqrt{\frac{g}{l}}$ and also setting $g = 1$
- Damping force parameter k. The damping force is $-mk\omega$, where $\omega = \dot{\theta}$
- The driving force is given by $mAcos(\Omega t)$, where A is the Amplitude of the driving force and Ω is the frequency.

The equation of motion can be written as,

$$\dot{\theta} = \omega \tag{1}$$
$$\dot{\omega} = -\beta^2 sin\theta - k\omega + Acos(\Omega t) \tag{2}$$

Step-1

(i) Euler predictor-corrector method

By averaging Euler and backward Euler methods we can achieve higher-order accuracy.

The Predictor step

$$\tilde{\theta_{k+1}} = \theta_k + h\omega_k(t_k) \tag{3}$$
$$\tilde{\omega_{k+1}} = \omega_k + hf(t_k, \theta_k, \omega_k) \tag{4}$$

The corrector step

$$\theta_{k+1} = \theta_k + \frac{h}{2}(\omega_k(t_k)\tilde{\omega_{k+1}}) \tag{5}$$
$$\omega_{k+1} = \omega_k + h(f(t_k, \theta_k, \omega_k) + f(t_{k+1}, \tilde{theta}_{k+1}, \tilde{\omega_{k+1}}) \tag{6}$$

Where $f(\theta_k, \omega_k, t_k) = -\beta^2 sin\theta - k\omega + Acos(\Omega t)$

To determine the accuracy of this algorithm, we apply it to a scalar ODE $x' = \lambda x$, obtaining

$$x_k = \left(\frac{1 + h\frac{\lambda}{2}}{1 - h\frac{\lambda}{2}} \right)^k x_0$$

The growth factor when epanded out gives,

$$\frac{1 + h\frac{\lambda}{2}}{1 - h\frac{\lambda}{2}} = 1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{(h\lambda)^3}{4} + \dots$$

This agrees with expression of $e^{h\lambda}$ through terms of order h^2, so this algorithm is **second-order accurate**.

(ii)Taylor series methods

This algorithm uses Taylor series expansion to solve the ODE.

From the Taylor series expansion of x(t+h), we truncate till second order for a *second-order* method.

$$x_{k+1} = x_k + hx_k^{'} + \frac{h^2}{2} x_k^{''}$$

The higher derivatives of x can be obtained by differentiating $x^{'} = f(t, x)$ using chain rule.

$$x^{''} = f_t(t, x) + f_x(t, x) f(t, x)$$

Now applying this algorithm to the equations of motion in this problem we get:

$$\theta_{k+1} = \theta_k + h\omega_k + \frac{h^2}{2}\theta_k^{''} \tag{7}$$

$$\theta_k^{''} = f(\theta_k, \omega_k, t) \tag{8}$$

$$\tag{9}$$

$$\omega_{k+1} = \omega_k + hf(\theta_k, \omega_k, t) + \frac{h^2}{2}\omega_k^{''} \tag{10}$$

$$\omega_k^{''} = \frac{f(\theta, \omega, t)}{dt} \tag{11}$$

$$= -\beta^2 \omega\ cos\theta - A\Omega\ sin(\Omega t) - kf(\theta_k, \omega_k, t) \tag{12}$$

This method is also **second-order** accurate.

(iii)Second-order Runge-Kutta methods

Runge-Kutta methods are a class of methods which judiciously uses the information on the 'slope' at more than one point to extrapolate the solution to the future time step.

For the ODE $\frac{dy}{dt} = f(t, y(t))$

$$\begin{aligned} k1 &= hf(t, y(t)) \\ k2 &= hf(t + \frac{h}{2}, y(t) + \frac{k1}{2}) \\ y(t + h) &= y(t) + k_2 \end{aligned}$$

This is also **second-order** accurate as it is evident from the expansion of $y(t + h)$:

$$y_{n+1} = y_n + hf(y_n, t_n) + \frac{1}{2}h^2(\frac{\partial f}{\partial t} + f\frac{\partial f}{\partial y})(y_n, t_n) + \mathcal{O}(h^3)$$

(iv)Fourth-order Runge-Kutta methods

We can increase the efficiency of the Runge-Kutta method by calculating more slopes. k_1 at the beginning of the interval. k_2 is the slope at the midpoint of the interval, using y and k_1. k_3 is the slope at the midpoint, but now using y and k_2.k_4 is the slope at the end of the interval, using y and k_3.In averaging the four slopes, greater weight is given to the slopes at the midpoint.

For the ODE,

$$\frac{dy}{dt} = f(t, y(t))$$

$$\begin{aligned} k1 &= hf(t_n, y_n) \\ k2 &= hf(t_n + \frac{h}{2}, y_n + \frac{k1}{2}) \\ k3 &= hf(t_n + \frac{h}{2}, y_n + \frac{k2}{2}) \\ k4 &= hf(t_n + h, y_n + hk_3) \end{aligned}$$

$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \\ t_{n+1} &= t_n + h \end{aligned}$$

This Method is **fourth-order accurate**.

Step-2

The files Odefunctions.h, Odefunctions.cpp and main.cpp contains code for solving the ODE for various cases and user's choice of algorithm. User can give

values of the parameter (A, k, Ω) , step size dt and the total number of steps N. The user can input the file name as well. The file name begins with the user's choice of algorithm. The various data files thus obtained are available in this directory. The plots are done using GNUPLOT. I have used six GNUPLOT scripts for my individual plots.

Inference

This section discusses my plots and the various conclusions that can be draen from it.

Comparing the four algorithms used

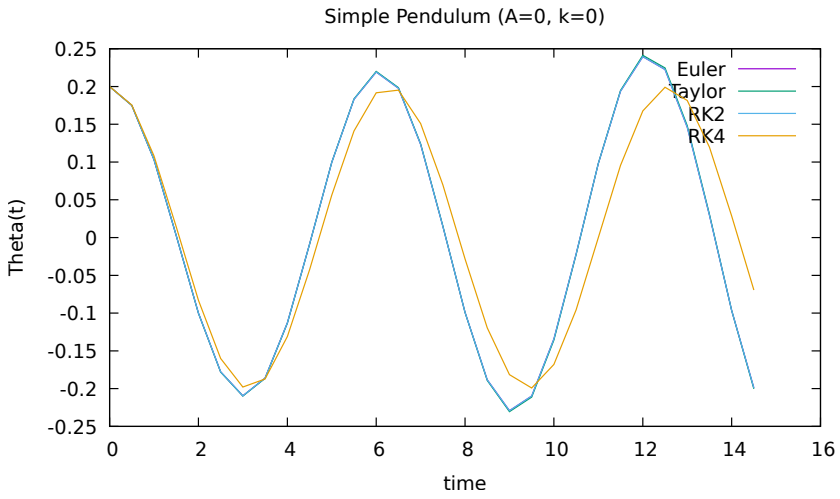

Figure 1: Simple pendulum case for the 4 different methods

In this plot, I'm considering the simple pendulum case $(A = 0, k = 0)$ for all the four methods considering a fixed time step (0.5) and number of points (30). The plot [1] clearly shows a match for Euler,Taylor and Runge-Kutta order 2 (RK2) methods. This is because they all are **2nd order** methods. Whereas, the **4th order** accurate *Runge-Kutta* (RK4) is different from the rest. Clearly, RK4 is the best and the most accurate method among the four algorithms discussed here.

Comparing the effect of time-step

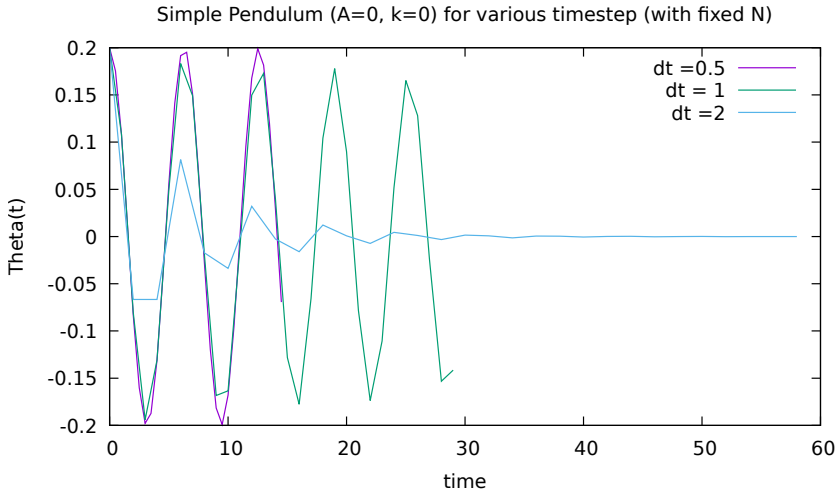

Figure 2: Simple pendulum case for different time steps

Here I'm plotting [2] the simple pendulum case for three different time steps $dt = 0.5, 1, 2$ for fixed number of points (30). We find that a smaller time step (0.5) reduces the error compared to a time step like 2.

Various physical situations

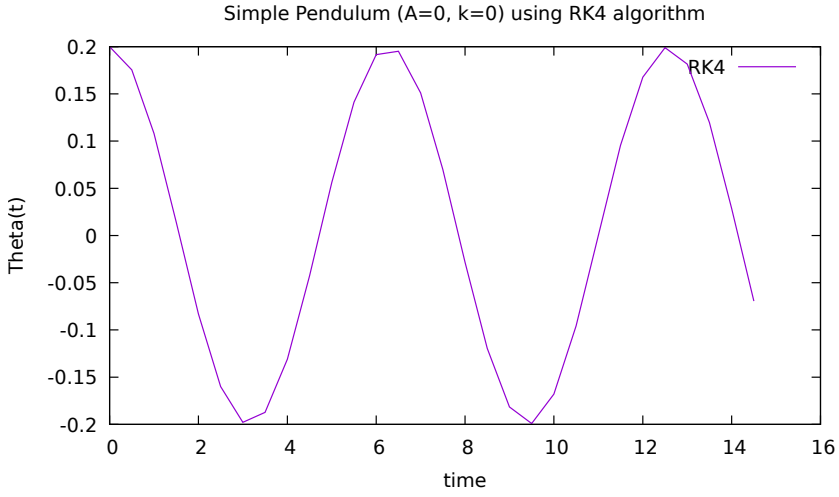

Figure 3: Simple pendulum case using RK4

The most straightforwad case is the simple pendulum which is basically a sinusoid [3].

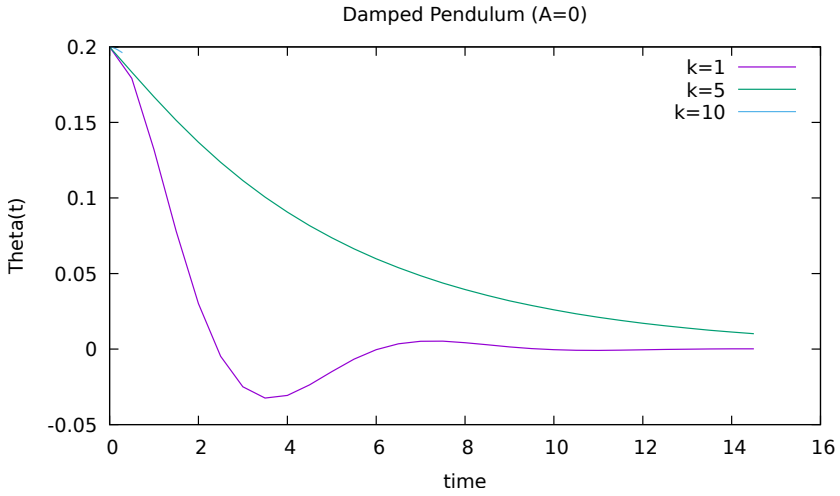

Figure 4: Damped pendulum case using RK4

The damped pendulum plot [4] shows that when you increaseing the damping parameter (k) the pendulum dies of quickly. As evident from the case of k=5. k=10 is even faster but it is not clearly visible in [4]

Here's a version of the plot featuring only k=10. Look how quickly the motion dampends to zero.

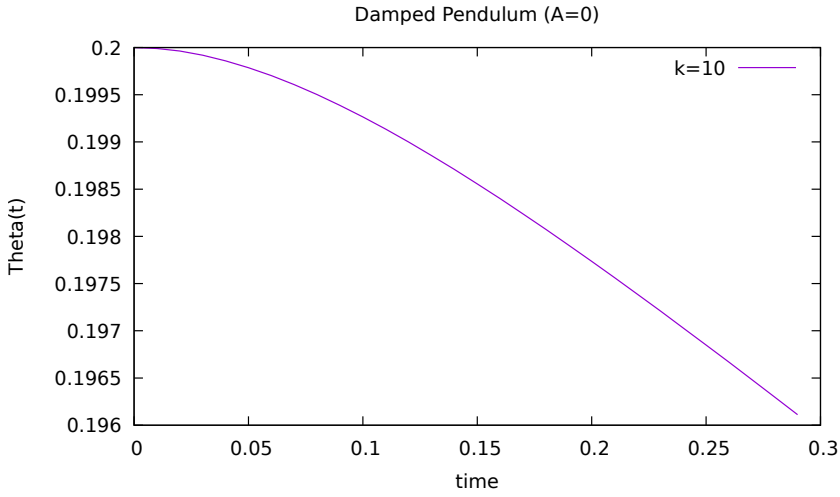

Figure 5: Damped pendulum case using RK4 for k=10

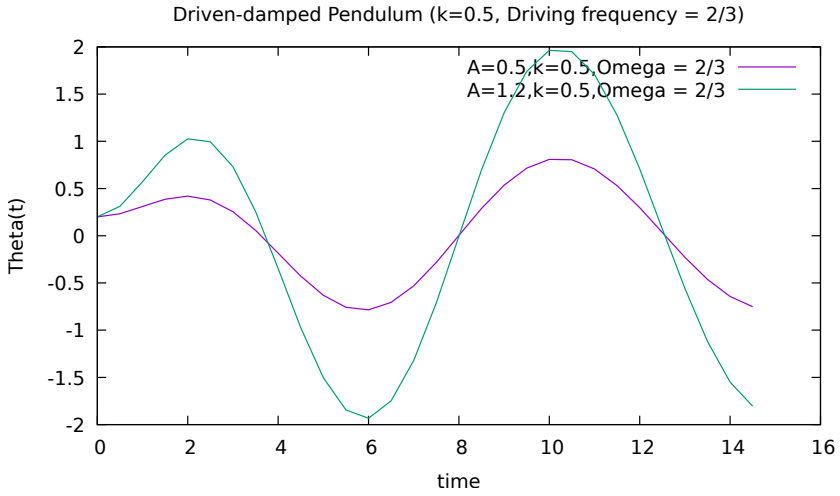

Figure 6: Damped-Driven pendulum case using RK4

The final case is the driven pendulum [6]. Evidently the driving force forces it away from damping and it achieves the amplitude of the driving force given at some point. When the frequency of the pendulum reaches the driving force frequency, the pendulum is in *resonance*.