Timebomb of approximation method in physics

vijay panchal

September 18, 2022

Contents

1 Introduction

Approximation method is yet most essential topic in physics. Physicists love to do approximations, like in functional expansion for getting polynomials for their ease or may be specialized idealization in particular topic. Approximation help them to **doing physics** instead going in to maze of exactness in mathematics. Getting interpretation or more i say knowing system is sometime more important then going for regorious mathematics. For example, famous equation of fluid dynamics **Navier-Stoke equation** can be imposible to solve but as physicist they know what it is.

Be aware, that approximation is just approximation. We should remember everytime we do that. Sometime we forgot actual system which is far from ideal. We should know that we are on mission to know nature not just building new theories.

Let's dive into one example, what do i imply by consequences of approximation methods. In classical mechanics, we have merely some major theories. In Oscillation theory we studied **Simple Harmonic Oscillation**, but as we are going to see that simple harmonic oscillation is not exactly that simple.

Write more

2 Example of Approximate method

Approximation is very used in almost every in physics. Not, just physics but every field in sciences. We are going to give profound example of understanding advantages and disadvantages of approximation.

2.1 Defining problem

We learned simple pendulum from early time of physics course. But what if say that simple pendulum is not actually simple in sense that approximation hide most of things away from our eyes to see.

We took physical pendulum. By taking length $l = 1m$ string (assuming non expanding) attaching to bob of mass $m = 0.1kg$. String attached to rigid wall as shown in figure.

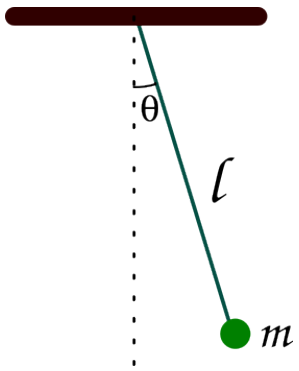

Figure 1: pendulum with string lenth l and mass m

For understanding consequences of approximation, we took simulations by solving both equation of motion (approximated and exact). For getting equations of motion we used **Newtonian formulation** which is quite easy to work with in this type of problems, since we are working with **non conservative** system. This is our derivation of equation of motion.

First of all, we took horizontal and vertical forces.

$$F_{damping}cos(\theta) - Tsin(\theta) = ma_x \tag{1}$$

$$F_{damping}sin(\theta) + Tsin(\theta) - mg = ma_y \tag{2}$$

Adding equation 1 and equation 2 with multiplication by $cos(\theta)$ and $sin(\theta)$ respectively.

$$F_{damping}sin^2(\theta) + F_{damping}cos^2(\theta) - mgsin(\theta) = ma_xcos(\theta) + ma_ysin(\theta)$$

$$F_{damping} - mgsin(\theta) = m(asin^2(\theta) + acos^2(\theta))$$

$$F_{damping} - mgsin(\theta) = ma \tag{3}$$

From,

$$a = (\ddot{r} - r\dot{\theta}^2)\hat{r} + (r\ddot{\theta} + 2\dot{r}\dot{\theta})\hat{\theta}$$

Where, $r = l$ and since $\dot{l} = 0$, $a = l\ddot{\theta}$. So, equation 3 becomes,

$$F_{damping} - mgsin(\theta) = ml\ddot{\theta}$$

We are taking damping force to velocity proportional. So, $F_{damping} = -bl\dot{\theta}$ (since, $\dot{l} = 0$). Here, we are assuming that damping is linear.

$$-bl\dot{\theta} - mgsin(\theta) = ml\ddot{\theta}$$

Rearranging this equation will give our equation of motion,

$$\ddot{\theta} + \frac{b}{m}\dot{\theta} + \frac{g}{l}sin(\theta) = 0 \tag{4}$$

This is **exact equation of motion**. Which is **second order non linear equation**. Finding it's exact solution is massive task. We will use numerical methods to find it's solution. This equation has much more to say than looks. Will learn about this in later.

2.2 Approximation of equation of motion : Linear differential equation

Solving 4 is very hard tasks. So, we as physics majors try to make this as friendly as possible. In class, we approximated this as $\theta \to 0$ as $sin(\theta) \to \theta$. Consequently, this equation becomes very easy to solve. This becomes,

$$\ddot{\theta} + \frac{b}{m}\dot{\theta} + \frac{g}{l}\theta = 0 \tag{5}$$

$$\ddot{\theta} + \Gamma\dot{\theta} + w_0^2\theta = 0 \tag{6}$$

Where we took $\Gamma = \frac{b}{m}$ and w_0^2.

We can solve this linear equation 6 by usual methods of linear differential equation. Simply taking $\theta = e^{\lambda t}$, which gives polynomials of second order.

$$\lambda^2 + \Gamma\lambda + w_0^2 = 0 \tag{7}$$

We can find roots of this quadratic equation.

$$\lambda = \frac{-\Gamma}{2} \pm \frac{\sqrt{\Gamma^2 - 4w_0^2}}{2} \tag{8}$$

$$\lambda = \frac{-\Gamma}{2} \pm \sqrt{\frac{\Gamma}{2}^2 - w_0^2} \tag{9}$$

Here we getting three type of roots,

1. Roots where $\frac{\Gamma}{2} = w$. this is **critical damping condition**, where we getting $\lambda = \frac{-\Gamma}{2}$. Putting λ into our solutions, $\theta = e^{\frac{-\Gamma}{2}t}$. Which suggest this will only decay with time and never overshoots from equilibrium position. Which is desired in certain condition but not for us.

2. Roots where $\frac{\Gamma}{2} > w$. this is **overdamping condition**, where we getting $\lambda = \frac{-\Gamma}{2} \pm \sqrt{\frac{\Gamma}{2}^2 - w_0^2}$. So from here we get $\theta = e^{\frac{-\Gamma}{2}t}e^{\pm\sqrt{\frac{\Gamma}{2}^2 - w_0^2}t}$. This also have exponential term in it which will only decay with time and never overshoots from equilibrium position.

3. Roots where $\frac{\Gamma}{2} < w$. this is **underdamping condition**, here $\lambda = \frac{-\Gamma}{2} \pm i\sqrt{w_0^2 - \frac{\Gamma}{2}^2}$. $\theta = e^{\frac{-\Gamma}{2}t}e^{\pm i\sqrt{w_0^2 - \frac{\Gamma}{2}^2}t}$. This has complex term, which implicitly suggest that it'll overshoot and oscillate. This our topic of interest for this project.

Solve linear equation

2.3 Simulations

2.3.1 Understand Runge-Kutta method

2.3.2 Animations

I done animation with python's pygame library which is acurate in time. This library make sures that our system follows as real life as possible.

```
def bob1(window,x,y):
    image = pygame.image.load("bitmap2.png")
    window.blit(image, (x,y))

def bob2(window,x,y):
    image = pygame.image.load("bitmap.png")
    window.blit(image, (x,y))

def surface(window,x,y):
    image = pygame.image.load("drawings.png")
    window.blit(image, (x, y))

def position(l,theta):
    return origin_x-10+l*cos((1.5*pi)-theta),origin_y-10-l*sin((1.5*pi)-
        theta)

def mainloop(window,fps):
    global t
    run = True
    clock = pygame.time.Clock()
    c = 0
    while run:
        for event in pygame.event.get():
            if event.type== pygame.QUIT:
                run= False
                break
        clock.tick(fps)

        x,y = position(l,exact[c])
        x2,y2 = position(l,appro[c])
        if c==len(exact):
            break
        window.fill("#ffffff")
        surface(window,175,origin_y-25)
        pygame.draw.aaline(window,color="#5BDEC1",start_pos=(origin_x,
            origin_y),end_pos=(x+radius,y+radius))
        pygame.draw.aaline(window,color="#5BDEC1",start_pos=(origin_x,
            origin_y),end_pos=(x2+10,y2+10))
        bob1(window,x,y)
        bob2(window,x2,y2)
        pygame.display.update()
```

```
            c+=1
        pygame.quit()

    if __name__ == "__main__":
        pygame.init()
        window = pygame.display.set_mode((width,height))
        exact = nonlinear(5,30)
        appro = linear(5,30)
        t = time.perf_counter()
        mainloop(window, 30)
```

3 What is meaning of all this ?