Numerical Methods

Zi-Seok Lee

2023-11-06

Three Different Methods of Approximation

- Euler’s Method
- Improved Euler’s Method
- Fourth-Order Runge-Kutta Method

Main Idea: Approximation by Iteration

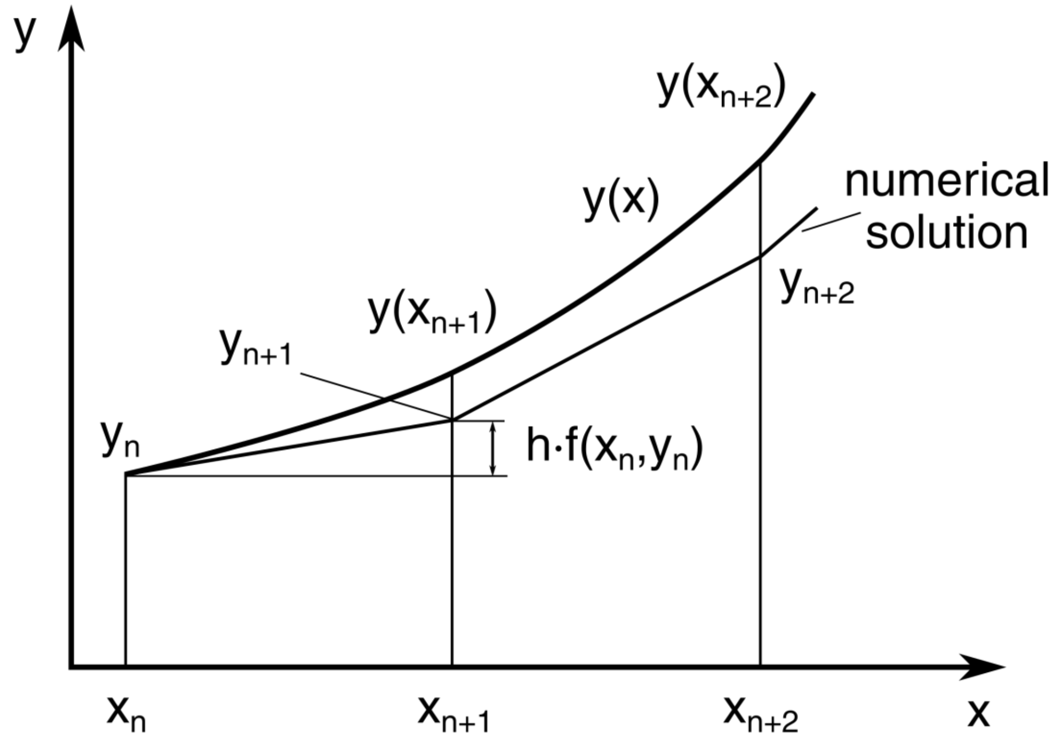
- Each of these methods involves an iterative process
- We find a sequence of points (t_k, x_k) that approximates the graph of a solution to

$$\dot{x} = f(t, x)$$

- Begin with an initial value $\left(\mathrm{t}_0, x(0)\right) = (0, x_0)$
- Choose a sufficiently small step size Δt and recursively generate $t_{k+1} = t_k + \Delta t$

Euler's Method

- Approximate the next step using a line:



$\rightarrow \quad x_{k+1} = x_k + f(t_k, x_k) \cdot \Delta t$

Improved Euler's Method

- We use the average of two slopes from (t_k, x_k) to (t_{k+1}, x_{k+1})

$$m_k = f(t_k, x_k)$$

$$n_k = f(t_{k+1}, y_k)$$

- Here, $y_k = x_k + m_k \Delta t$ is the point determined by the original Euler's method.
- Then we have

$$x_{k+1} = x_k + \left(\frac{m_k + n_k}{2}\right) \Delta t$$

4^{th} Order Runge-Kutta Method

- This method has served as a general-purpose solver for decades

$$x_{k+1} = x_k + \left(\frac{m_k + 2a_k + 2b_k + c_k}{6}\right)\Delta t$$

- Let's draw on the board to understand this method.
  - $m_k = f(t_k, x_k)$ as in Euler's method
  - $a_k = f(t_k + \frac{\Delta t}{2}, y_k)$ where $y_k = x_k + m_k \frac{\Delta t}{2}$
  - $b_k = f(t_k + \frac{\Delta t}{2}, z_k)$ where $z_k = x_k + a_k \frac{\Delta t}{2}$
  - $c_k = f(t_{k+1}, w_k)$ where $w_k = x_k + b_k \,\Delta t$

Why is the Runge-Kutta method "4th Order"?

1. Use the three methods to approximate the value of $x(1) = e$ in the following system:

$$\dot{x} = x, \qquad x(0) = 1$$

Discuss how the errors change as you shorten the step size Δt

2. (***Chaos; sensitive dependence on initial conditions***) Sketch the graph of the system:

$$\dot{x} = e^t \sin x\,, \qquad x(0) = 0.3$$

(1) Use Euler's method with $\Delta t = 0.3,\ 0.001,\ 0.002,\ 0.003.$

(2) Repeat for $x(0) = 0.301,\ 0.302.$

(3) Is there any change when using RK4 method?

Application to the Hodgkin-Huxley Model

```python
import math
import numpy as np
import matplotlib.pyplot as plt

def alphaM(V):
    return (2.5-0.1*(V+65)) / (np.exp(2.5-0.1*(V+65)) -1)
def betaM(V):
    return 4*np.exp(-(V+65)/18)
def alphaH(V):
    return 0.07*np.exp(-(V+65)/20)
def betaH(V):
    return 1/(np.exp(3.0-0.1*(V+65))+1)
def alphaN(V):
    return (0.1-0.01*(V+65)) / (np.exp(1-0.1*(V+65)) -1)
def betaN(V):
    return 0.125*np.exp(-(V+65)/80)
```

```python
def HH(I0, T0):
    dt = 0.01
    T = math.ceil(T0/dt)    # [ms]
    gNa0    = 120           # [mS/cm^2]
    ENa     = 115           # [mV]
    gK0     = 36            # [mS/cm^2]
    EK      = -12           # [mV]
    gL0     = 0.3           # [mS/cm^2]
    EL      = 10.6          # [mV]

    t = np.arange(0,T)*dt
    V = np.zeros([T,1])
    m = np.zeros([T,1])
    h = np.zeros([T,1])
    n = np.zeros([T,1])

    V[0] = -70
    m[0] = 0.05
    h[0] = 0.54
    n[0] = 0.34
```

Application to the Hodgkin-Huxley Model

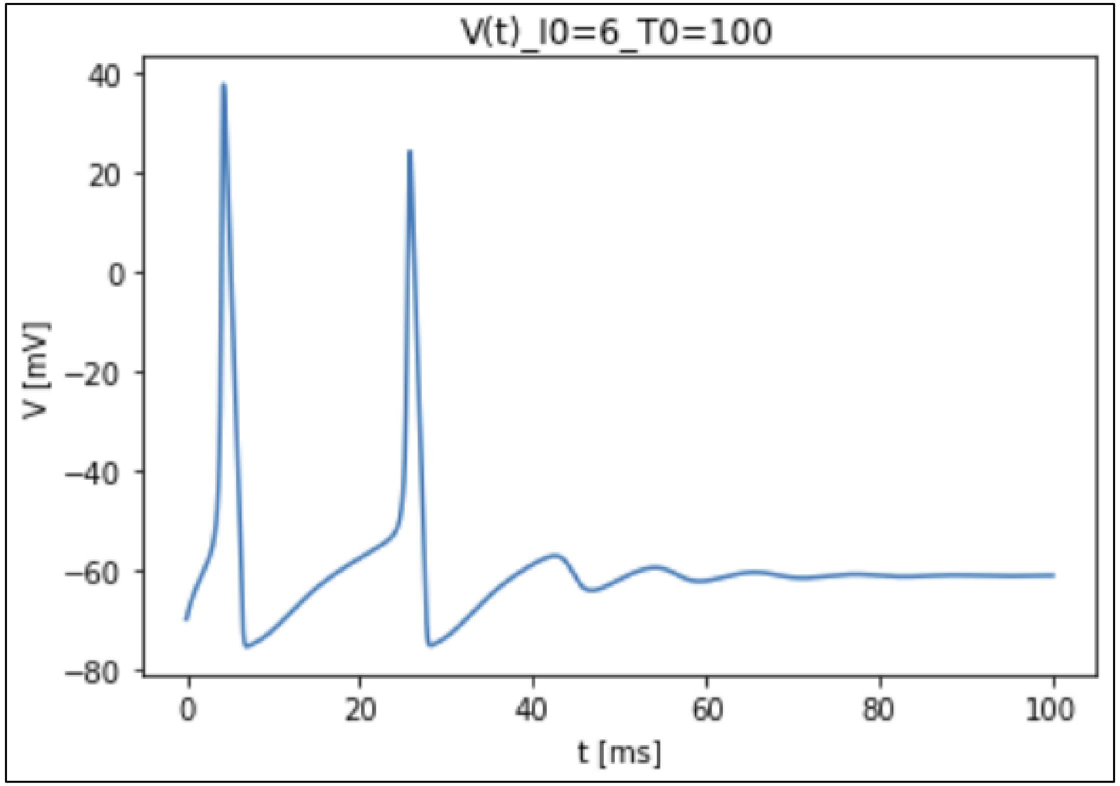
- Euler’s method:

```
# def HH continued
    for i in range(0, T-1):
        V[i+1] = V[i] + dt*(gNa0*m[i]**3*h[i]*(ENa-(V[i]+65)) + gK0*n[i]**4*(EK-(V[i]+65)) + gL0*(EL-(V[i]+65)) + I0)
        m[i+1] = m[i] + dt*(alphaM(V[i])*(1-m[i]) - betaM(V[i])*m[i])
        h[i+1] = h[i] + dt*(alphaH(V[i])*(1-h[i]) - betaH(V[i])*h[i])
        n[i+1] = n[i] + dt*(alphaN(V[i])*(1-n[i]) - betaN(V[i])*n[i])
    return V,m,h,n,t
```

- Q. Should we use other methods, or is the Euler method enough?

Application to the Hodgkin-Huxley Model

- At low input current, examine the HH dynamics
- Repeat for high input currents
- Does your model generate repeated spikes?
- Plot the gating variables (h,n) and describe how the gates open and close during a spike
- Describe the dynamics of the conductances



References

- https://mark-kramer.github.io/Case-Studies-Python/HH.html
- Hirsch, Devaney, and Smale, *Differential Equations, Dynamical Systems, and an Introduction to Chaos*