Climate Modeling

October 24, 2024

PRACTICAL REPORT

MODEL HIERARCHY & SIMPLIFIED CLIMATE MODELS

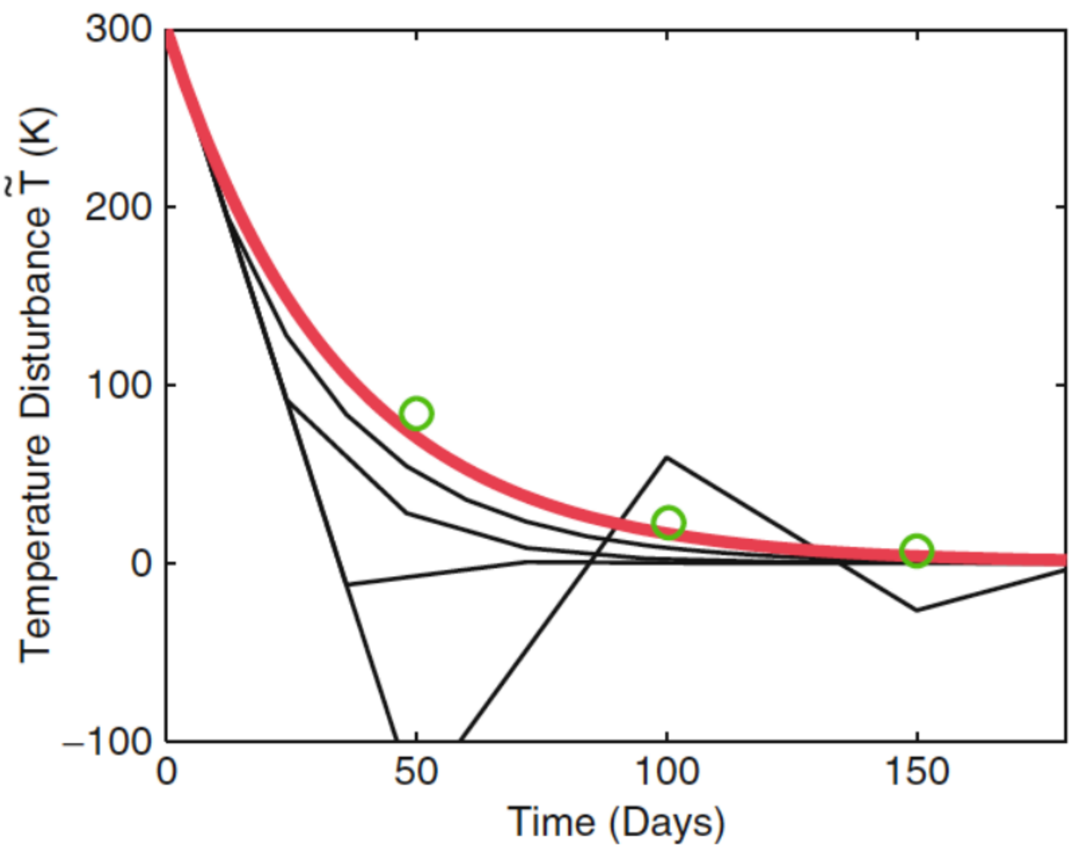
By:

Duong Thu Phuong

22BI13362

Contents

Reproduce the following figure.



Numerical solutions of (7) with initial perturbation = 300K computed with the Euler scheme and time steps of 12, 24, 36, 50 days. The analytical solution is in red; the results from the classical Runge-Kutta scheme (t = 50days) are labelled with green circles.

Initial perturbation 300K, tau=35 days

1 Plot the analytical red curve

$$T = ae^{-\frac{t}{\tau}}$$

where:

$$\tau = 35$$

$$a = T(0) = 300k$$

For analytical result, I choose dt = 50

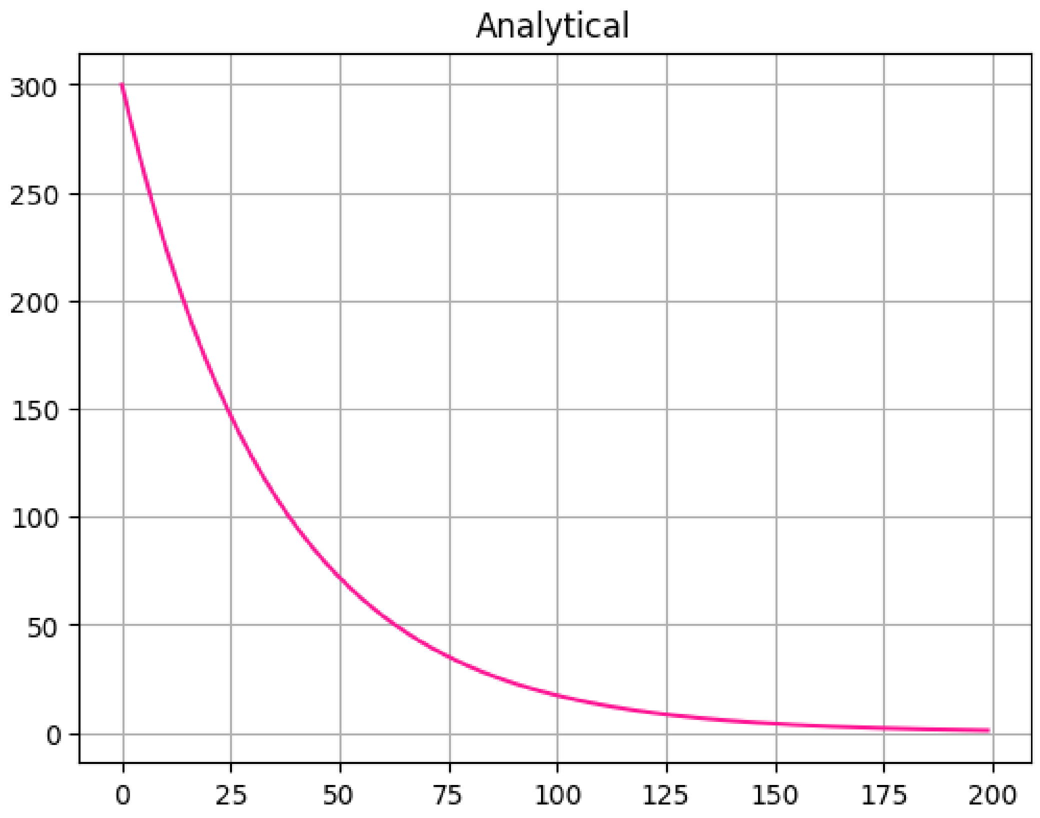
```
[1]: import numpy as np
     import matplotlib.pyplot as plt
```

```
[2]: T0 = 300
     tau = 35
```

```
dt_rk = 50

t_ana = np.arange(0, 200)
Ts_ana = T0 * np.exp(-t_ana / tau)

plt.plot(t_ana, Ts_ana, color='deeppink')
plt.title('Analytical')
plt.grid()
plt.show()
```



2 Overlay the numerical solutions using the Euler scheme with different time steps of 12, 24, 36, 50 days

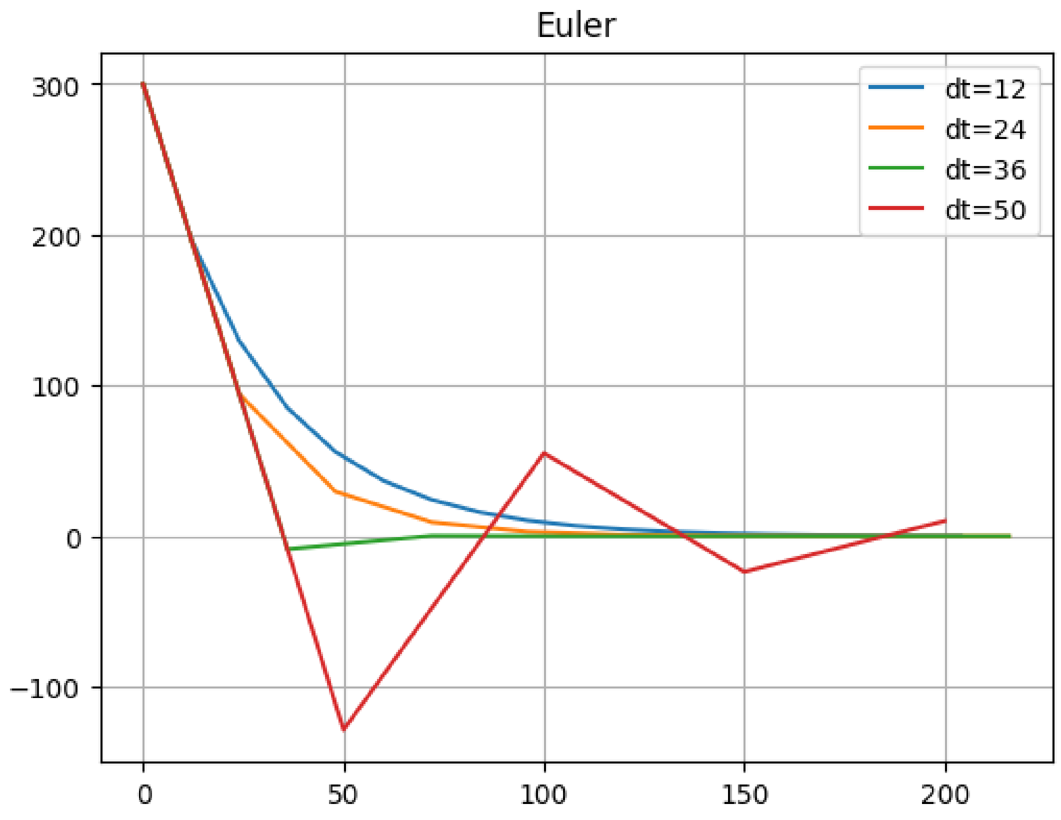
$$T_{x+1} = T_x - \frac{T_x \cdot dt}{\tau}$$

I repeat this calculation for different time step 12, 24, 36, and 50.

[3]:
```
euler=[]
for dt in [12, 24, 36, 50]:
    t = np.arange(0, 200 + dt, dt)
    Ts = np.zeros(len(t))
    Ts[0] = T0

    for i in range(1, len(t)):
        Ts[i] = Ts[i - 1] * (1 - dt / tau)
    euler.append((t, Ts, dt))
    plt.plot(t, Ts, label=f'dt={dt}')

plt.title('Euler')
plt.grid()
plt.legend()
plt.show()
```



Lower dt gives much better approximation than high dt.

3 Overlay the results from the Runge-Kutta scheme with Δt= 50 days

Differential equation:

$$f(x, y) = -\frac{1}{\tau}y$$

[4]:
```
def f(t, T):
    return (-1 / tau) * T
```

Runge Kutta:

$$F(x_n, y_n) = \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

where:

$$K_1 = f(x_n, y_n)$$
$$K_2 = f\left(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}\Delta x K_1\right)$$
$$K_3 = f\left(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}\Delta x K_2\right)$$
$$K_4 = f(x_n + \Delta x, y_n + \Delta x K_3)$$

In the code it creates a function that takes three inputs: time, temperature, and time step. It use a loop to calculate the temperature at each point using the Runge-Kutta formula:

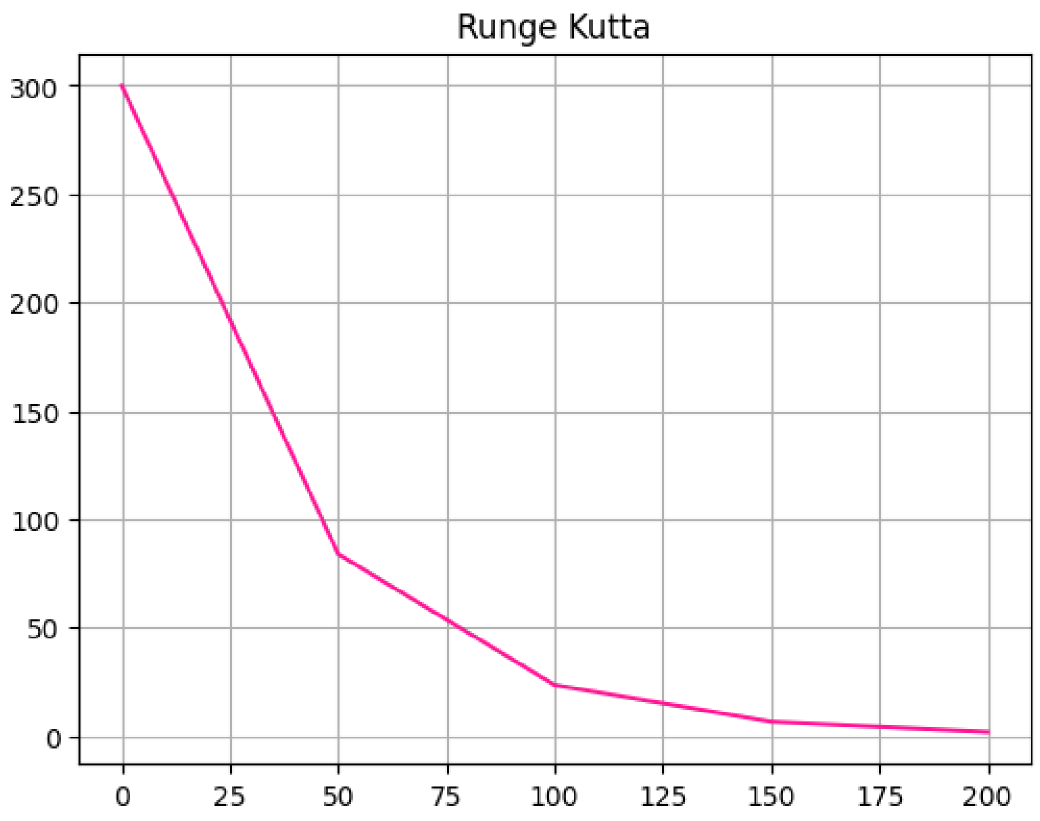
$$T_{x+1} = T_x + dt \cdot F(t_i, T_i, dt)$$

[5]:
```
def runge_kutta_step(t, T, dt):
    K1 = f(t, T)
    K2 = f(t + 0.5 * dt, T + 0.5 * dt * K1)
    K3 = f(t + 0.5 * dt, T + 0.5 * dt * K2)
    K4 = f(t + dt, T + dt * K3)
    return (K1 + 2 * K2 + 2 * K3 + K4) / 6

t_rk = np.arange(0, 200 + dt_rk, dt_rk)
T_rk = np.zeros(len(t_rk))
T_rk[0] = T0

for i in range(len(t_rk) - 1):
    T_rk[i + 1] = T_rk[i] + dt_rk * runge_kutta_step(t_rk[i], T_rk[i], dt_rk)
plt.plot(t_rk, T_rk, color='deeppink')

plt.title('Runge Kutta')
plt.grid()
plt.show()
```

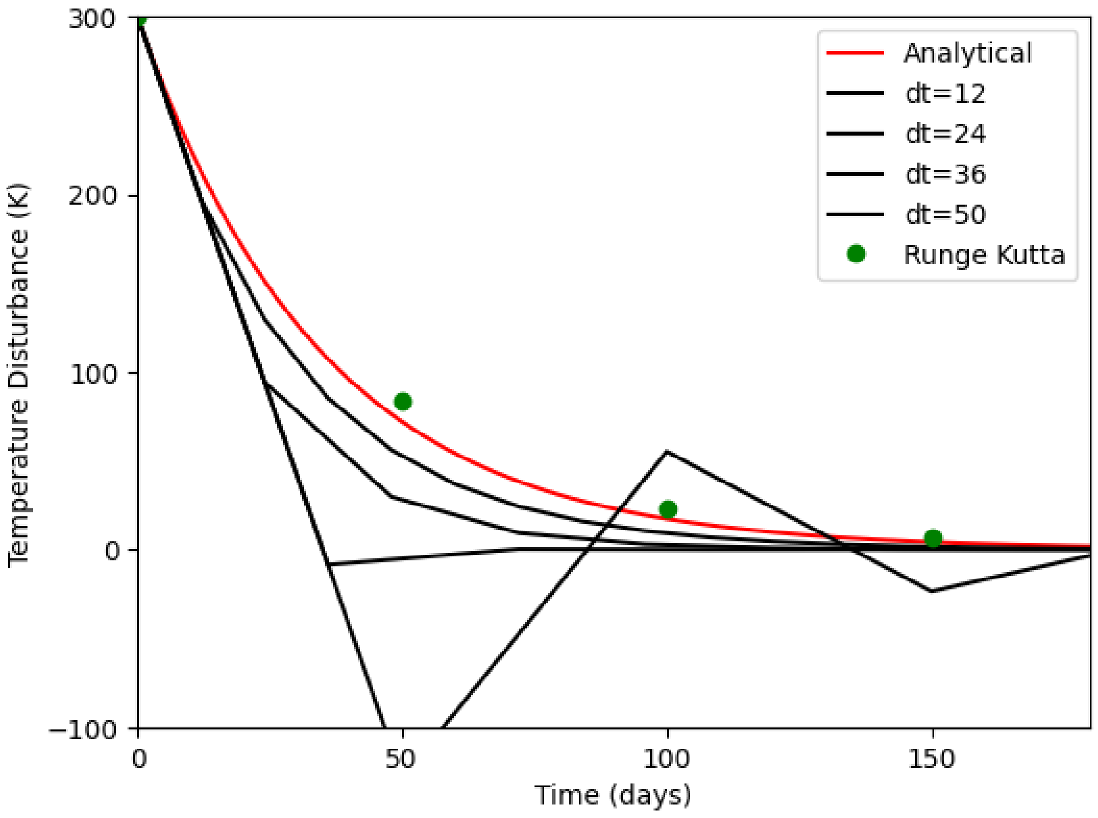


Compare to the Euler method curve with the same dt=50, Runge Kutta method is more accurate.

Then I plot all functions and try to make it looks similar as the given figure as much as possible.

[6]:
```
plt.plot(t_ana, Ts_ana, label="Analytical", color='red')
for t, Ts, dt in euler:
    plt.plot(t, Ts, label=f'dt={dt}', color='black')

plt.plot(t_rk, T_rk, 'o', label='Runge Kutta', color='green')
plt.xlim(0, 180)
plt.xticks([0, 50, 100, 150])
plt.ylim(-100, 300)
plt.yticks([-100, 0, 100, 200, 300])
plt.xlabel("Time (days)")
plt.ylabel("Temperature Disturbance (K)")
plt.legend()
plt.show()
```



4 Conclusion

- For the Euler method, if dt decreases it gives a curve that looks the most like the analytical one, which means a better approximation.
- The Runge Kutta method is much better than the Euler method, with the same dt=50, for Euler it doesn't even have the right shape, but for Runge Kutta the overall shape follows the analytical one really close.