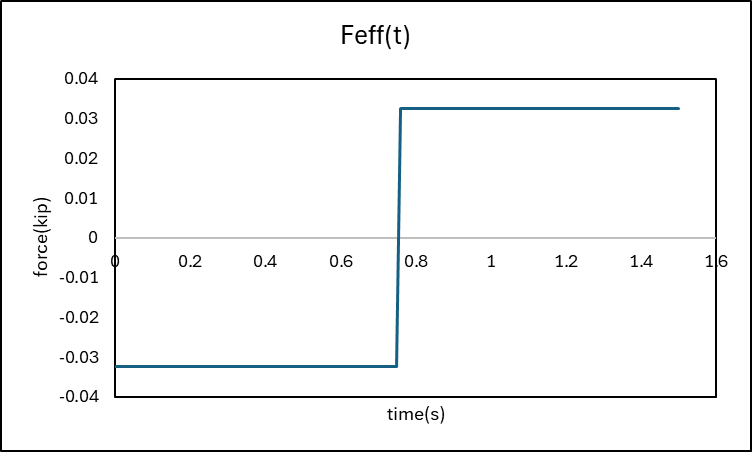
Earthquake Engineering Homework3

土木碩一11372009邱昱倫

1. Plot Feff(t)



1. Plot x(t)

一張含有 文字, 繪圖, 行, 圖表 的圖片

AI 產生的內容可能不正確。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| displacement | | | | |
| Time (s) | Avg Acc (in) | Linear Acc (in) | Central Diff (in) | Wilson's theta(in) |
| 0 | 0 | 0 | 0 | 0 |
| 0.01 | -0.004584852 | -0.004664328 | 0 | -0.001667459 |
| 0.02 | -0.017712594 | -0.017913159 | -0.009341013 | -0.006460946 |
| 0.03 | -0.037878957 | -0.038221382 | -0.026722132 | -0.013725884 |
| 0.04 | -0.063158643 | -0.06364006 | -0.050325588 | -0.022735588 |
| 0.05 | -0.09137323 | -0.091967298 | -0.07798595 | -0.032706218 |
| 0.06 | -0.120267017 | -0.120926886 | -0.107374377 | -0.042857957 |

1. Plot x’(t)

一張含有 繪圖, 行, 圖表, 文字 的圖片

AI 產生的內容可能不正確。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| velocity | | | | | |
| Time (s) | Avg Acc (in/s) | Linear Acc (in/s) | Central Diff (in/s) | Wilson's theta (in/s) |
| 0 | 0 | 0 | 0 | 0 |
| 0.01 | -0.916970421 | -0.916673466 | -0.467050643 | -0.343287255 |
| 0.02 | -1.708577962 | -1.707253826 | -1.33610658 | -0.625854593 |
| 0.03 | -2.324694728 | -2.321427224 | -2.049228763 | -0.839621849 |
| 0.04 | -2.731242344 | -2.725107522 | -2.563190921 | -0.975719877 |
| 0.05 | -2.911675022 | -2.901917694 | -2.852439464 | -1.031708296 |
| 0.06 | -2.867082515 | -2.853269707 | -2.909862458 | -1.010927194 |

1. Plot x’’(t)

一張含有 文字, 繪圖, 行, 圖表 的圖片

AI 產生的內容可能不正確。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| acceleration | | | | |
| Time (s) | Avg Acc (in/s2) | Linear Acc (in/s2) | Central Diff (in/s2) | Wilson's theta(in/s2) |
| 0 | -96.525 | -96.525 | -96.525 | -37.26733725 |
| 0.01 | -86.86908429 | -86.8096933 | -93.41012865 | -31.39011368 |
| 0.02 | -71.45242381 | -71.30637854 | -80.40105865 | -25.12335392 |
| 0.03 | -51.77092933 | -51.52830122 | -62.22337803 | -17.63009745 |
| 0.04 | -29.53859395 | -29.20775835 | -40.56905354 | -9.58950803 |
| 0.05 | -6.547941735 | -6.154275928 | -17.28065505 | -1.608175865 |
| 0.06 | 15.46644313 | 15.88387318 | 5.796056131 | 5.764396296 |

1. Appendix

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import os

def get\_user\_input():

    """取得使用者輸入的參數"""

    file\_path = input("請輸入地震歷時檔案完整路徑（例如 C:/.../Northridge\_NS.txt 或 .csv）：")

    save\_path = input("請輸入要儲存結果的資料夾路徑：").strip()

    filename = input("請輸入檔案名稱（不含副檔名）：").strip()

    W\_kip = float(input("請輸入結構重量 W (kips)："))

    k\_kip\_in = float(input("請輸入結構勁度 k (kip/in)："))

    zeta = float(input("請輸入阻尼比 ζ (0~1)："))

    u0 = float(input("請輸入初始位移 u0 (in)："))

    v0 = float(input("請輸入初始速度 v0 (in/s)："))

    dt = float(input("請輸入時間間隔 Δt (s)："))

    return file\_path, save\_path, filename, W\_kip, k\_kip\_in, zeta, u0, v0, dt

def compute\_structure\_parameters(W\_kip, k\_kip\_in, zeta):

    """計算結構參數"""

    W\_lbf = W\_kip \* 1000

    m = W\_lbf / 386.1  # slug

    k = k\_kip\_in \* 1000  # lbf/in

    wn = np.sqrt(k / m)

    c = 2 \* zeta \* m \* wn

    return m, k, c

def read\_earthquake\_data(file\_path):

    """讀取地震歷時資料"""

    if file\_path.endswith('.txt'):

        data = np.loadtxt(file\_path)

    elif file\_path.endswith('.csv'):

        data = pd.read\_csv(file\_path).values

    else:

        raise ValueError("不支援的檔案格式，請提供 .txt 或 .csv 檔案")

    time = data[:, 0]

    ag = data[:, 1] \* 386.1  # g → in/s²

    return time, ag

def average\_acceleration\_method(time, ag, m, k, c, u0, v0, dt):

    """使用平均加速度法計算時間歷程分析"""

    gamma = 0.5

    beta = 0.25

    npts = len(ag)

    u = np.zeros(npts)

    v = np.zeros(npts)

    a\_resp = np.zeros(npts)

    u[0] = u0

    v[0] = v0

    p = -m \* ag  # 慣性力

    a\_resp[0] = (p[0] - c \* v[0] - k \* u[0]) / m

    k\_eff = k + gamma / (beta \* dt) \* c + m / (beta \* dt \*\* 2)

    for i in range(1, npts):

        rhs = p[i] + \

              m \* ((1 / (beta \* dt \*\* 2)) \* u[i - 1] + (1 / (beta \* dt)) \* v[i - 1] + (1 / (2 \* beta) - 1) \* a\_resp[i - 1]) + \

              c \* ((gamma / (beta \* dt)) \* u[i - 1] + ((gamma / beta) - 1) \* v[i - 1] + dt \* ((gamma / (2 \* beta)) - 1) \* a\_resp[i - 1])

        u[i] = rhs / k\_eff

        a\_resp[i] = (1 / (beta \* dt \*\* 2)) \* (u[i] - u[i - 1] - dt \* v[i - 1]) - (1 - 2 \* beta) / (2 \* beta) \* a\_resp[i - 1]

        v[i] = v[i - 1] + dt \* ((1 - gamma) \* a\_resp[i - 1] + gamma \* a\_resp[i])

    return u, v, a\_resp

def linear\_acceleration\_method(time, ag, m, k, c, u0, v0, dt):

    """使用線性加速度法計算時間歷程分析"""

    gamma = 0.5

    beta = 1 / 6

    npts = len(ag)

    u = np.zeros(npts)

    v = np.zeros(npts)

    a\_resp = np.zeros(npts)

    u[0] = u0

    v[0] = v0

    p = -m \* ag  # 慣性力

    a\_resp[0] = (p[0] - c \* v[0] - k \* u[0]) / m

    k\_eff = k + gamma / (beta \* dt) \* c + m / (beta \* dt \*\* 2)

    for i in range(1, npts):

        rhs = p[i] + \

              m \* ((1 / (beta \* dt \*\* 2)) \* u[i - 1] + (1 / (beta \* dt)) \* v[i - 1] + (1 / (2 \* beta) - 1) \* a\_resp[i - 1]) + \

              c \* ((gamma / (beta \* dt)) \* u[i - 1] + ((gamma / beta) - 1) \* v[i - 1] + dt \* ((gamma / (2 \* beta)) - 1) \* a\_resp[i - 1])

        u[i] = rhs / k\_eff

        a\_resp[i] = (1 / (beta \* dt \*\* 2)) \* (u[i] - u[i - 1] - dt \* v[i - 1]) - (1 - 2 \* beta) / (2 \* beta) \* a\_resp[i - 1]

        v[i] = v[i - 1] + dt \* ((1 - gamma) \* a\_resp[i - 1] + gamma \* a\_resp[i])

    return u, v, a\_resp

def central\_difference\_method(time, ag, m, k, c, u0, v0, dt):

    """使用中央差分法計算時間歷程分析"""

    npts = len(ag)

    u = np.zeros(npts)

    v = np.zeros(npts)

    a\_resp = np.zeros(npts)

    # 初始條件

    u[0] = u0

    v[0] = v0

    a\_resp[0] = (1 / m) \* (-c \* v[0] - k \* u[0] - m \* ag[0])

    # 預計算常數

    k\_eff = m / dt\*\*2 + c / (2 \* dt)

    a = m / dt\*\*2 - c / (2 \* dt)

    b = k - 2 \* m / dt\*\*2

    # 時間歷程計算

    for i in range(1, npts - 1):

        p\_eff = -m \* ag[i] - a \* u[i - 1] - b \* u[i]

        u[i + 1] = p\_eff / k\_eff

        v[i] = (u[i + 1] - u[i - 1]) / (2 \* dt)

        a\_resp[i] = (u[i + 1] - 2 \* u[i] + u[i - 1]) / dt\*\*2

    # 最後一點速度和加速度

    v[-1] = (u[-1] - u[-2]) / dt

    a\_resp[-1] = (u[-1] - 2 \* u[-2] + u[-3]) / dt\*\*2

    return u, v, a\_resp

def wilsons\_theta\_method(time, ag, m, k, c, u0, v0, dt, theta=1.4):

    n\_steps = len(ag)

    # 加速度換算為 in/s²，並計算質量

    #g\_in\_s2 = 386.09

    p\_eff = - m  \* ag   # F = ma

    # 初始化

    u = np.zeros(n\_steps)

    v = np.zeros(n\_steps)

    a\_resp = np.zeros(n\_steps)

    # 初始條件

    u[0] = u0

    v[0] = v0

    a\_resp[0] = (p\_eff[0] - c \* v[0] - k \* u[0]) / m

    # 有效剛度

    k\_eff = k + 3 \* c / (theta \* dt) + 6 \* m / (theta\*\*2 \* dt\*\*2)

    for i in range(n\_steps - 1):

        if i + 1 < n\_steps:

            F\_theta = p\_eff[i] + theta \* (p\_eff[i+1] - p\_eff[i])

        else:

            F\_theta = p\_eff[i]

        rhs = F\_theta + \

            m \* (6\*u[i]/(theta\*\*2\*dt\*\*2) + 6\*v[i]/(theta\*dt) + 2\*a\_resp[i]) + \

            c \* (3\*u[i]/(theta\*dt) + 2\*v[i] + theta\*dt\*a\_resp[i]/2)

        u\_theta = rhs / k\_eff

        a\_resp[i+1] = (6/(theta\*\*2\*dt\*\*2)) \* (u\_theta - u[i] - theta\*dt\*v[i]) - 2\*a\_resp[i]

        v[i+1] = v[i] + dt \* (a\_resp[i] + a\_resp[i+1]) / 2

        u[i+1] = u[i] + dt \* v[i] + (dt\*\*2 / 6) \* (a\_resp[i] + a\_resp[i+1] + a\_resp[i+1])

    return u, v, a\_resp

def save\_results\_to\_csv(time, u1, v1, a1, u2, v2, a2, u3, v3, a3, u4, v4, a4, save\_path, filename):

    """儲存所有方法的結果到 CSV"""

    full\_path = os.path.join(save\_path, f"{filename}.csv")

    # 計算每種方法的最大值

    max\_values = {

        "Average Acceleration Method": {

            "Max Displacement (in)": np.max(np.abs(u1)),

            "Max Velocity (in/s)": np.max(np.abs(v1)),

            "Max Acceleration (in/s²)": np.max(np.abs(a1))

        },

        "Linear Acceleration Method": {

            "Max Displacement (in)": np.max(np.abs(u2)),

            "Max Velocity (in/s)": np.max(np.abs(v2)),

            "Max Acceleration (in/s²)": np.max(np.abs(a2))

        },

        "Central Difference Method": {

            "Max Displacement (in)": np.max(np.abs(u3)),

            "Max Velocity (in/s)": np.max(np.abs(v3)),

            "Max Acceleration (in/s²)": np.max(np.abs(a3))

        },

        "Wilson's Method": {

            "Max Displacement (in)": np.max(np.abs(u4)),

            "Max Velocity (in/s)": np.max(np.abs(v4)),

            "Max Acceleration (in/s²)": np.max(np.abs(a4))

        }

    }

    # 建立 DataFrame

    df = pd.DataFrame({

        "Time (s)": time,

        "Avg Acc Displacement (in)": u1,

        "Avg Acc Velocity (in/s)": v1,

        "Avg Acc Acceleration (in/s²)": a1,

        "Lin Acc Displacement (in)": u2,

        "Lin Acc Velocity (in/s)": v2,

        "Lin Acc Acceleration (in/s²)": a2,

        "Central Diff Displacement (in)": u3,

        "Central Diff Velocity (in/s)": v3,

        "Central Diff Acceleration (in/s²)": a3,

        "Wilson's Displacement (in)": u4,

        "Wilson's Velocity (in/s)": v4,

        "Wilson's Acceleration (in/s²)": a4

    })

    # 空出一列，然後附加最大值

    df = pd.concat([df, pd.DataFrame([{}])], ignore\_index=True)  # 空白列

    for method, values in max\_values.items():

        max\_row = pd.DataFrame({

            "Time (s)": [f"MAX ({method})"],

            "Avg Acc Displacement (in)": [values.get("Max Displacement (in)", "")],

            "Avg Acc Velocity (in/s)": [values.get("Max Velocity (in/s)", "")],

            "Avg Acc Acceleration (in/s²)": [values.get("Max Acceleration (in/s²)", "")]

        })

        df = pd.concat([df, max\_row], ignore\_index=True)

    # 儲存到 CSV

    df.to\_csv(full\_path, index=False)

    print(f"結果已儲存至：{full\_path}")

def plot\_results(time, u1, v1, a1, u2, v2, a2, u3, v3, a3, u4, v4, a4, save\_path, filename):

    """繪製結果圖表，並儲存到指定路徑"""

    # 建立完整的儲存路徑

    displacement\_path = os.path.join(save\_path, f"{filename}\_displacement.png")

    velocity\_path = os.path.join(save\_path, f"{filename}\_velocity.png")

    acceleration\_path = os.path.join(save\_path, f"{filename}\_acceleration.png")

    # 繪製位移圖

    plt.figure(figsize=(10, 6))

    plt.plot(time, u1, color='black', linestyle='-', linewidth=1.5, label='Average Acceleration Method')

    plt.plot(time, u2, color='blue', linestyle='--', linewidth=1.5, label='Linear Acceleration Method')

    plt.plot(time, u3, color='red', linestyle='-.', linewidth=1.5, label='Central Difference Method')

    plt.plot(time, u4, color='green', linestyle=':', linewidth=1.5, label="Wilson's Method")

    plt.title("Displacement vs Time")

    plt.xlabel("Time (s)")

    plt.ylabel("Displacement (in)")

    plt.legend()

    plt.grid(True)

    plt.savefig(displacement\_path)

    plt.show()

    # 繪製速度圖

    plt.figure(figsize=(10, 6))

    plt.plot(time, v1, color='black', linestyle='-', linewidth=1.5, label='Average Acceleration Method')

    plt.plot(time, v2, color='blue', linestyle='--', linewidth=1.5, label='Linear Acceleration Method')

    plt.plot(time, v3, color='red', linestyle='-.', linewidth=1.5, label='Central Difference Method')

    plt.plot(time, v4, color='green', linestyle=':', linewidth=1.5, label="Wilson's Method")

    plt.title("Velocity vs Time")

    plt.xlabel("Time (s)")

    plt.ylabel("Velocity (in/s)")

    plt.legend()

    plt.grid(True)

    plt.savefig(velocity\_path)

    plt.show()

    # 繪製加速度圖

    plt.figure(figsize=(10, 6))

    plt.plot(time, a1, color='black', linestyle='-', linewidth=1.5, label='Average Acceleration Method')

    plt.plot(time, a2, color='blue', linestyle='--', linewidth=1.5, label='Linear Acceleration Method')

    plt.plot(time, a3, color='red', linestyle='-.', linewidth=1.5, label='Central Difference Method')

    plt.plot(time, a4, color='green', linestyle=':', linewidth=1.5, label="Wilson's Method")

    plt.title("Acceleration vs Time")

    plt.xlabel("Time (s)")

    plt.ylabel("Acceleration (in/s²)")

    plt.legend()

    plt.grid(True)

    plt.savefig(acceleration\_path)

    plt.show()

def main():

    """主程式流程"""

    file\_path, save\_path, filename, W\_kip, k\_kip\_in, zeta, u0, v0, dt = get\_user\_input()

    m, k, c = compute\_structure\_parameters(W\_kip, k\_kip\_in, zeta)

    time, ag = read\_earthquake\_data(file\_path)

    # 使用平均加速度法計算

    u1, v1, a1 = average\_acceleration\_method(time, ag, m, k, c, u0, v0, dt)

    # 使用線性加速度法計算

    u2, v2, a2 = linear\_acceleration\_method(time, ag, m, k, c, u0, v0, dt)

    # 使用中央差分法計算

    u3, v3, a3 = central\_difference\_method(time, ag, m, k, c, u0, v0, dt)

    # 使用 Wilson's 方法計算

    u4, v4, a4 = wilsons\_theta\_method(time, ag, m, k, c, u0, v0, dt)

    # 儲存結果到 CSV（僅儲存平均加速度法的結果）

    save\_results\_to\_csv(time, u1, v1, a1, u2, v2, a2, u3, v3, a3, u4, v4, a4, save\_path, filename)

    # 繪製圖表並儲存圖片

    plot\_results(time, u1, v1, a1, u2, v2, a2, u3, v3, a3, u4, v4, a4, save\_path, filename)

if \_\_name\_\_ == "\_\_main\_\_":

    main()