3801ICT: Numerical Algorithms

2024 Term 1

Final Assignment Specification

Maximum Marks Achievable: 60

This assignment is **worth 60% of the total course marks**.

The assignment files should be submitted online.

Deadline for submission is Week 11, Friday, 24 May 2024, 17:00:00.

Submission: You **must submit a PDF** report, which include your python implement, explanation of the choice of numerical methods, an analysis of the results, and visualizations of achieved results for each task. To avoid confusion for the teaching staff/marker, who may handle many submissions, put your sID in the file names (e.g. **sID_Report.pdf**).

Hints for submission: Jupyter notebook can achieve these by using "Code" mode cells for implementation and visual results and "Markdown" mode cells for text and report, which can be then saved as PDF files.

Objective: This assignment is designed to be data-driven and context-rich, challenging you to apply your theoretical knowledge to solve practical, tangible problems using numerical algorithms.

Descriptions:

Task 1 (15 marks): Weather Trend Analysis.

The data below consists of daily average temperatures recorded over the last 50 years. (1) Implement a numerical method to estimate the rate of temperature change over the years; and (2) Predict the temperature trend for the next 10 years; using methods discussed in the course.

Expected deliverables: Plots of the temperature trend over time and the forecasted potential future temperatures.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import UnivariateSpline

# Constants
years = 50
days_per_year = 365

# Generate time data
time = np.linspace(0, years, years * days_per_year)

# Simulate temperature data
np.random.seed(0)
temperature = 10 + 0.02 * time  # Linear trend: slight increase over time
```

```
temperature += 7 * np.sin(2 * np.pi * time / 365)  # Seasonal variation
temperature += np.random.normal(0, 1, temperature.shape)  # Random daily fluctuations
```

Task 2 (15 marks): Air Pollution Analysis.

Daily average concentrations of major pollutants (e.g., PM2.5, NO2) collected over the past year from an urban monitoring station are given below. (1) Fit the daily pollution data and identify seasonal trends; and (2) Integrate the approximated function over time to estimate the total pollution exposure.

Expected deliverables: A fitted curve to the yearly data, graphical representations of trends, and a calculated exposure index for health impact assessments.

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.polynomial.polynomial import Polynomial

# Constants
days_per_year = 365

# Generate time data (days)
time = np.arange(1, days_per_year + 1)

# Simulate PM2.5 data with seasonal variation and random noise
base_pollution_level = 40  # Average pollution level
seasonal_amplitude = 15  # Amplitude of seasonal variation
noise_level = 5  # Noise level

np.random.seed(0)  # For reproducibility
pollution_data = (base_pollution_level
                  + seasonal_amplitude * np.sin(2 * np.pi * time / days_per_year)
                  + np.random.normal(0, noise_level, days_per_year))
```

Task 3 (15 marks): Satellite Orbit Determination.

Data of time-stamped position observations of a satellite over one orbit is given below. (1) Estimate the satellite's position at times where direct observations aren't available; and (2) Optimize the trajectory by adjusting the estimated points slightly to minimize the hypothetical fuel usage, assuming that smaller changes in velocity require less fuel.

Expected deliverables: Plots of a path of the satellite's orbit and an optimized trajectory that meets fuel efficiency and orbit accuracy goals.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import CubicSpline
from scipy.optimize import minimize

# Constants
num_points = 20  # number of observation points
orbit_period = 24  # period of the orbit in hours
```

```
# Time stamps
time_stamps = np.linspace(0, orbit_period, num_points)

# Generate a circular orbit for simplicity
radius = 10000  # radius of the orbit in km
angles = 2 * np.pi * time_stamps / orbit_period
positions = np.vstack((radius * np.cos(angles), radius * np.sin(angles))).T

# Add some noise to simulate real observations
noise = np.random.normal(0, 100, positions.shape)  # noise in km
observations = positions + noise
```

Task 4 (15 marks): Economic Growth Modelling.

Annual GDP growth rates and investment in research and development (R&D) as a percentage of GDP for the past 30 years are given below. We assume using a simplified model where GDP growth rate ($\frac{dG}{dt}$) is proportional to the current level of GDP (G) multiplied by R&D investment (R).

$$\frac{dG}{dt} = k \cdot G \cdot R$$

where:

- G is the GDP.
- R is the percentage of GDP invested in R&D.
- k is a proportionality constant.
- t is the time.

(1) Solve this equation to calculate how the GDP changes over time ($G(t)$) and (2) integrate the results to estimate the total GDP growth.

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

# Constants
k = 0.02  # Proportionality constant
years = np.linspace(1990, 2020, 31)  # 30 years from 1990 to 2020
R = 0.1 + 0.02 * np.sin(0.2 * np.pi * (years - 1990))  # R&D investment fluctuates

# Initial condition
G0 = 1000  # Initial GDP in billion USD
```