CS101 Advanced Engineering Mathematics (I)

工程數學(一)

【Guidelines】

1. All the homework in this course will involve solving advanced engineering mathematics problems (differential equations in particular) by hand and computer.
2. While discussion in class is allowed, you MUST work independently to generate your own solutions to the problems.
3. Python programming will be used for solving problems and plotting solutions. You may reference the Appendix (講義附錄) for details.

本次作業改為**個人作業** (不進行分組)，請同學依 Template 完成書面作業 (紙面)，並於期限內繳交至計算機視覺研究室 (電學大樓 603 室)。

【General Instructions】

To get a good grading on the homework assignments, you are advised to do the following:

- Start early, don't wait until last minutes! (請盡早開始，不然你肯定做不完)
- Do not copy other classmate's works! (請遵守學術倫理，嚴禁抄襲)
- Prepare your written reports in good quality (使用 Template 檔，編寫專業報告)
- Meet the deadline! Late homework will **NOT** be collected. (按時繳交，逾時不候)

【課程助教】

計算機視覺研究室 (電學大樓 603 室)
電話：(03) 265-4726

指導教授：張元翔

Homework Assignment 2
Differential Equations and Applications

Deadline: 1 / 2 / 2026 (星期五)

【Problem 1】

The ***Gradient Descent method*** & ***Newton's method*** are useful for solving the optimization problems. In this homework assignment, your goal is to design the Python program that applies the two methods.

The iteration equation using ***Gradient Descent method*** can be defined as:

$$x_{n+1} = x_n - \alpha f'(x_n)$$

where x_n is the *n*th iteration and α is the learning rate.

The iteration equation using ***Newton's method*** can be defined as:

$$x_{n+1} = x_n - \alpha \frac{f(x_n)}{f'(x_n)}$$

Please reference the Python Tutorial (課程講義) for details.

Suppose the function can be defined as: $y = f(x) = x^2 - 2x + 5$, please answer the following:

(a) Design the Python program using ***Gradient Descent*** method and print the values of $(x_n, y_n), n = 0...10$ (i.e., the first 10 iterations) given the initial guess $x_0 = 0$ and $\alpha = 0.1$.

(b) Design the Python program using ***Newton's method*** and print the values of $(x_n, y_n), n = 0...10$ (i.e., the first 10 iterations) given the initial guess $x_0 = 0$ and $\alpha = 1.0$.

(c) Plot the function and show $(x_n, y_n), n = 0...10$ in the ***same figure*** (For plotting, *x* is in the range of –0.5 ~ 2.5). As a result, you will generate two plots.

Note: All the figures must be carefully ***labeled***, ***titled***, and ***copyright*** for full credit.
注意：Copyright 訊息必須有你的姓名，中英文均可，否則不予計分。

【Problem 2】

Direction fields are particularly useful for solving first-order differential equations when analytic solutions can't be found. To plot a direction field for a first-order differential equation with Python programming, the equation must be in normal form. A first-order differential equation is in normal form if it is expressed as:

$$\frac{dy}{dx} = f(x, y)$$

Following the aforementioned instructions, use the Python programming to obtain the direction field for each of the following differential equations (the interval I is given for (x, y) coordinates accordingly). Attach the figures in your written report.

(a) $\frac{dy}{dx} = x + y$; I: [-5 : 0.5 : 5, -5 : 0.5 : 5]

(b) $\frac{dy}{dx} = x - y$; I: [-5 : 0.5 : 5, -5 : 0.5 : 5]

(c) $\frac{dy}{dx} = xy$; I: [-5 : 0.5 : 5, -5 : 0.5 : 5]

(d) $\frac{dy}{dx} = \sin x \cos y$; I: [-5 : 0.5 : 5, -5 : 0.5 : 5]

Finally, discuss your findings in details (中文解釋即可).

Note: All the figures must be carefully ***labeled***, ***titled***, and ***copyright*** for full credit.

【Problems 3】

Please answer the following:

(1) Solve the following ***Differential Equation*** (DE) or ***Initial-Value Problem*** (IVP) (手寫推導);

(2) Use the Python programming to plot the solution curve. The interval I is the given range for the x-data in the plots.

(a) $y'' + 2y' + 10y = 0,\ y(0) = 2,\ y'(0) = -1$
(For plotting, let I : [0, 1])

(b) $y'' - y = e^x$
(For plotting, let I : [0, 1] and $c_1 = c_2 = 1$)

(c) $y'' + 4y = \cos(2x),\ y(0) = 1,\ y'(0) = 0$
(For plotting, let I : [0, 4π])

(d) $y''' - y'' + 2y = 0$
(For plotting: let I : [0, 1] and $c_1 = c_2 = c_3 = 1$)

Note: All the figures must be carefully ***labeled***, ***titled***, and ***copyright*** for full credit.

【Problem 4】

The objective is to explore the chaotic dynamics of a ***double pendulum*** system by numerically solving its equations of motion and visualizing the trajectory of the second mass.

Please reference and modify the sample codes <double_pendulum.py> and generate two plots with following conditions:

(a) Moving trajectory of the second mass (i.e., chaotic motion) with:
$m_1 = m_2 = 1.0$ kg, $l_1 = l_2 = 1.0$ m, $t = 0 \sim 60$ seconds,

$\theta_1 = 60°, \theta_2 = 0°$.

(b) Moving trajectory of the second mass with the same condition as (a),
but with slightly different initial condition:
$m_1 = m_2 = 1.0$ kg, $l_1 = l_2 = 1.0$ m, $t = 0 \sim 60$ seconds,

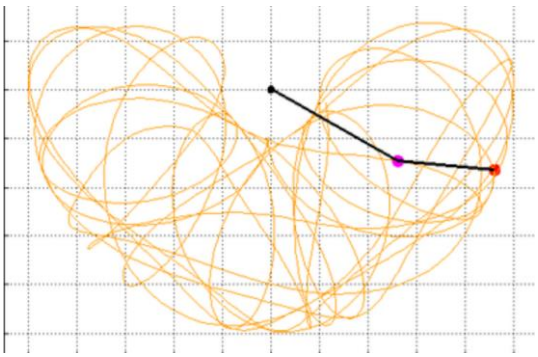
$\theta_1 = 60.01°, \theta_2 = 0°$.

(c) Moving trajectory of the second mass (i.e., chaotic motion) with:
$m_1 = 1.0$ kg, $m_2 = 0.5$ kg, $l_1 = 1.0$ m, $l_2 = 0.5$ m, $t = 0 \sim 60$ seconds,

$\theta_1 = 60°, \theta_2 = 0°$.

Finally, discuss your findings in details (中文解釋即可).

Note: All the figures must be carefully ***labeled***, ***titled***, and ***copyright*** for full credit.
(請參考下圖，畫出第二顆球的運動軌跡即可)



【混沌理論】

混沌理論中的初始條件是指系統的最初狀態，其極小的差異會在系統演化過程中被不斷放大，最終導致結果產生巨大差異，這就是「**蝴蝶效應**」。這種對初始條件的極度敏感性是混沌系統最關鍵的特徵之一，使得其長期行為難以準確預測。

【Problem 5】

The objective is to implement an image special effect, known as **slim-face effect** (瘦臉特效), in the field of digital image processing.

An example of the **slim-face effect** is given as follows. Here, you may reference the related software packages, i.e., OpenCV, MediaPipe, and SciPy.

Original Image　　Slim-Face Effect

Fig. An example of the slim-face effect.

In this homework assignment, you must demonstrate experimental results of at least 4 sets of digital images (Original Image & Slim-Face Effect) in your written report. Two of them must be the given images, i.e., <chinese_girl.jpg> & <mona_lisa.jpg>, while other 2 digital images can be your choices of interest.

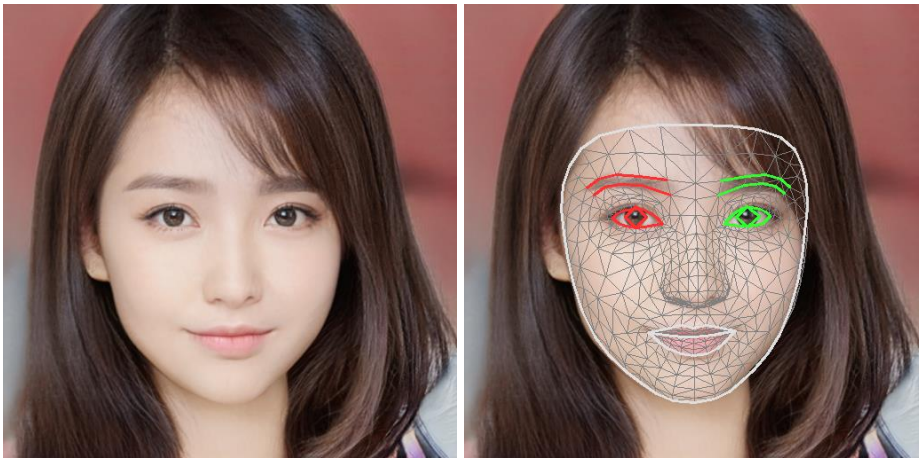
Finally, discuss your findings in details (中文解釋即可).

注意：結果圖、討論與 Python 程式須放在書面報告內。

Slim-Face Effect Pipeline

(1) Face Detection and Facial Landmark Extraction

Use a facial analysis toolkit such as MediaPipe to detect the face region in the input image and extract the corresponding facial landmarks. These landmarks include the eyebrows, eyes, nose tip, lips, and facial contour, which together describe the overall facial geometry. An example of facial landmark extraction using MediaPipe framework is given in the following figure.

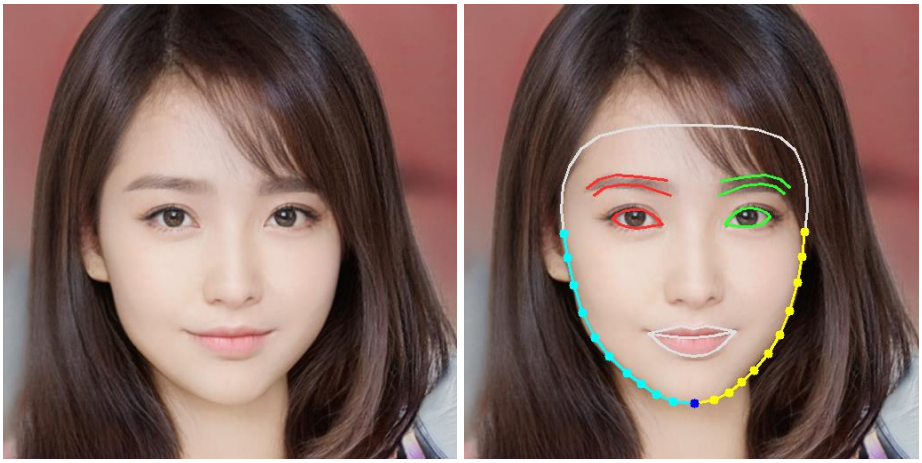


Original Image Facial Landmarks

Fig. An example of the facial landmark extraction.

(2) Selecting Target Landmarks for Deformation

According to the objective of the slim-face effect, select the key regions to be reshaped—primarily the jawline and cheeks. These landmarks serve as control points for deformation. A typical jawline detection result divides the jawline into left and right sides, each containing ten key points (the blue point represents the midpoint of the jawline).



Original Image Jawline

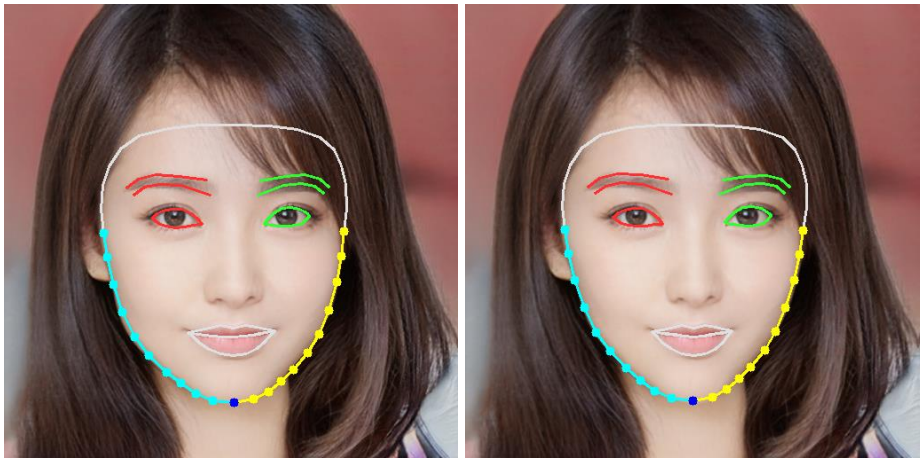
Fig. An example of the facial jawline detection.

(3) Computing the New Jawline for the Slim-Face Effect

After selecting the left and right jawline control points, compute the new positions of these points after applying the slimming transformation. The new landmark positions act as target points for constructing a global deformation field later. The following figure illustrates the original jawline and the new one after applying the slimming logic.

Computation Logic:

- Obtain ten pairs of corresponding landmark coordinates for the left and right jawlines.
- Compute the center point for each pair of landmarks.
- Assign a ratio (scaling factor) based on the landmark's vertical position (i.e., its y value).
- Landmarks near the upper face (smaller y) use ratios close to 1, while those near the lower face (larger y) use smaller ratios.



Original Jawline New Jawline

Fig. An example of the facial jawline deformation.

(4) Constructing the Deformation Field Using RBFInterpolator

Using the original control points (original jawline) and the target control points (slimmed jawline), construct a smooth deformation field that transfers the offset of these control points to the entire image. Here, we employ SciPy's RBFInterpolator — a multidimensional interpolator based on ***Radial Basis Functions*** (RBFs), which is well suited for non-uniform coordinate warping problems such as facial reshaping. In addition, the OpenCV library provides cv2.remap() function, which is well suited for image transformations.

Note: You should add four corners as the control points as well to avoid geometric distortion in other area.

【瘦臉特效的數學原理】

RBF（Radial Basis Function）**內插器**（Interpolator）是一種基於徑向基底函數的空間內插方法，廣泛應用於影像變形、幾何扭曲與非剛性對應等問題。其核心思想是，透過一組已知的對應點（控制點），建立一個平滑且可微分的變形場，使任意空間點的對應位置能夠自然地推算出來。

給定 n 個對應點，定義為：

- **原始控制點：** $\mathbf{x}_i = (x_i, y_i) \in \mathbb{R}^2,\ i = 1, 2, ..., n$
- **目標控制點：** $\hat{\mathbf{x}}_i = (\hat{x}_i, \hat{y}_i) \in \mathbb{R}^2,\ i = 1, 2, ..., n$

我們希望求一個變形函數 $f : \mathbb{R}^2 \to \mathbb{R}^2$，使得：

$$f(\mathbf{x}_i) = \hat{\mathbf{x}}_i$$

為了滿足上述條件，定義函數 f 為徑向基底函數的加權總和：

$$f(\mathbf{x}) = \sum_{i=1}^{n} \omega_i \cdot \phi(\| \mathbf{x} - \mathbf{x}_i \|)$$

其中：

- $\mathbf{x} = (x, y) \in \mathbb{R}^2$ 為任意輸入點。
- $\phi(r)$ 為**徑向基底函數**（RBF）。
- $\omega_i \in \mathbb{R}^2$ 為待求的向量權重。
- $\| \cdot \|$ 為歐幾里得距離。

常見的徑向基底函數，包含：**Linear**、**Cubic**、**Gaussian**、**Multiquadric** 以及 **Thin-Plate Spline（TPS）**。其中，**TPS** 具備良好的全域平滑性與物理解釋，在影像幾何變形中尤為常用。因此，在本節中，我們採用 **TPS** 作為 **RBF 內插的徑向基底函數**，以建立控制點之間的平滑變形場。

Thin-Plate Spline（薄板樣條）是 RBF 內插器中的一種特殊形式，其基底函數為：

$$\phi(r) = r^2 \log r$$

這個函數源自物理中對「金屬薄板彎曲」的模擬，其數學推導是基於最小化以下的彎曲能量函數：

$$E(f) = \iint \left[\left(\frac{\partial^2 f}{\partial x^2} \right)^2 + \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 f}{\partial y^2} \right)^2 \right] dx\ dy$$

TPS 所建構出的變形函數具有極佳的平滑性，並且在控制點之外的區域仍能自然延伸，適合應用於人臉、身體姿勢、字體與地圖等需要細緻彎曲的幾何形變問題。