Assignment #8

Math 2604: Mathematical Programming in Python

https://people.sc.fsu.edu/~jburkardt/classes/python_2025/assignment08/assignment08.pdf

---

Instructions: Choose 3 of the following problems to work on. Submit your responses as Python text files, with the extension `.py`. Each file should include your name and the problem number.

Several of the problems below ask you to work with the `sinc()` function, defined by

$$\operatorname{sinc}(x) = \frac{\sin(x)}{x}$$

Of course this function has a (removable) singularity at $x = 0$, and so various Python functions may complain or fail if you don't handle this properly. However, the `numpy()` library already has defined an `np.sinc()` function, so I would suggest you use that instead!

- *Problem 8.0:* The function `scipy.linalg.lstsq()` solves an overdetermined linear system $Ax = b$ in the least squares sense, returning an approximate solution x which minimizes the Euclidean norm of the residual $||Ax - b||$. The function returns four arguments, but we are only interested in x, the first one, so you might call it this way:

  ```
  x, _, _, _ = lstsq ( A, b )
  ```

  Use this function to solve the linear system in which

  $$A = \begin{bmatrix} 0.00 & 0.00 & 1.00 \\ 0.04 & 0.20 & 1.00 \\ 0.16 & 0.40 & 1.00 \\ 0.36 & 0.60 & 1.00 \\ 0.64 & 0.80 & 1.00 \\ 1.00 & 1.00 & 1.00 \end{bmatrix} \quad b = \begin{bmatrix} 150.6970 \\ 179.3230 \\ 203.2120 \\ 226.5050 \\ 249.6330 \\ 281.4220 \end{bmatrix}$$

  Once you have computed the solution, print out the norm of the residual.

- *Problem 8.1:* Suppose an ellipsoid has axes $a > b > c$. We wish to sompute S, the surface area of this object. To do so, we need to import the incomplete elliptic integral functions "K()" = `ellipkinc()` and "E()"=`ellipeinc()` from `scipy.special()`. We define

  $$\cos(\phi) = \frac{c}{a} \quad k = \frac{a\sqrt{b^2 - c^2}}{b\sqrt{a^2 - c^2}}$$

  Then

  $$S = 2\pi c^2 + \frac{2\pi ab}{\sin(\phi)} (K(\phi, k^2)\cos^2(\phi) + E(\phi, k^2)\sin^2(\phi))$$

  Compute the value of S, assuming $a = 5, b = 4, c = 2$. If you are doing things correctly, your answer should be between 160 and 170.

- *Problem 8.2:* Use the `scipy.integrate()` function `quad()` to estimate the integral of `sinc(x)` over the interval $0 \leq x \leq 8$.

- *Problem 8.3:* Use the `scipy.integrate()` function `dblquad()` to estimate the double integral

  $$\int_1^4 \int_0^2 x^2\, y\, dy\, dx$$

  Using this function can be very tricky. Read the documentation carefully. The correct value is 42, so check your work if you get something different.

- *Problem 8.4:* A Newton Cotes quadrature rule of order n is a set of $n+1$ weights w and $n+1$ equally spaced points in the interval $[-1,+1]$ which approximate the integral of a function $f(x)$ by

$$I = \int_{-1}^{+1} f(x)\,dx \approx Q = \sum_{i=1}^{n+1} w_i f(x_i)$$

If our integration interval is $[a,b]$, then we redefine the x values using `X = np.linspace(a,b,n+1)` and we rescale the weights by

$$W = \frac{(b-a)*w}{n}$$

and use `X` and `W` to estimate the integral. Call the `scipy.integrate()` function `newton_cotes()` with `n = 16`, and estimate the integral of `sinc(x)` over the interval $0 \le x \le 8$. The function returns two arguments, and you should ignore the second one. Perhaps do this with a call like

```
w, _ = newton_cotes ( n )
```

- *Problem 8.5:* A Gauss Legendre quadrature rule of order n is a set of n weights w and n unequally spaced points in the interval $[-1,+1]$ which approximate the integral of a function $f(x)$ by

$$I = \int_{-1}^{+1} f(x)\,dx \approx Q = \sum_{i=1}^{n+1} w_i f(x_i)$$

If our integration interval is $[a,b]$, then we redefine the x and w values by

$$X = \frac{((x+1.0)\,b + (1.0-x)\,a)}{2}$$
$$W = \frac{(b-a)\,w}{2}$$

and use `X` and `W` to estimate the integral. Call the `scipy.special()` function `p_roots(n)` with `n = 8`, and estimate the integral of `sinc(x)` over the interval $0 \le x \le 8$.

- *Problem 8.6:* Use the `scipy.interpolate()` function `interp1d()` to interpolate the `sinc(x)` function over the interval $-8 \le x \le +8$. Use just 12 equally spaced data points for your interpolaton, and use the interpolation option `kind = 'cubic'`. On one graph, plot
  - your 12 data points as circles or dots
  - the value of your interpolating function at 51 equally spaced points
  - the value of the sinc(x) function at 51 equally spaced points.

- *Problem 8.7:* Estimate the maximum value of the `sinc(x)` function over the interval $1 \le x \le 4$ in two ways:
  - Evaluate the function at 61 points and report the maximum value and the corresponding x value (use `np.argmax()` for that)
  - use the `scipy.optimize()` function `minimize_scalar()`. You want the maximum, so you have to make a function that is the negative of `sinc(x)`. Also, use the `method='bounded'` with bounds [1,4].

- *Problem 8.8:* Use the `scipy.optimize()` function `brentq()` to find the zero of "jumper(x)" within the interval $-1 \le x \le 2$. The function is defined by

$$jumper(x) = \cos(100x) - 4\,\mathrm{erf}(30x - 10)$$

where `erf(x)` is the error function, which is available in `scipy.special()`.

- *Problem 8.9:* Use the `scipy.optimize()` function `minimize()` to find the minimizer of the L2 norm of the Madsen function. You will have to write a Python function `madsen(xy)` which evaluates the Madsen function at $xy = [x, y]$ defined by

$$f(xy) = \begin{bmatrix} x^2 + y^2 + xy \\ \sin(x) \\ \cos(y) \end{bmatrix}$$

For a starting point, use $xy0 = [3, 1]$ The `minimize()` code returns an object called `result`.

```
result = minimize ( madsen, xy0    )
```

The values of the minimizer vector xy will be returned as `result.x`