MAS212 Scientific Computing and Simulation

#8: Well-conditioned and ill-conditioned problems

Key resources:

- Lec 8: `http://sam-dolan.staff.shef.ac.uk/mas212/docs/l8.pdf`
- Example code: `http://sam-dolan.staff.shef.ac.uk/mas212/code/poly_example.py`
- Course webpage : `http://sam-dolan.staff.shef.ac.uk/mas212/`

---

1. Wilkinson's polynomial. In Lec. 8 we looked at Wilkinson's polynomial,

$$p_n^{(0)}(x) = (x-1)(x-2)\dots(x-n) = \prod_{k=1}^{n}(x-k)$$

and a particular 'perturbed' version,

$$p_n(x,\epsilon) = p_n^{(0)}(x) + \epsilon x^{n-1}.$$

Clearly, $p_n^{(0)}(x)$ has roots (zeros) at $x = 1, 2, \dots n$. The introduction of a small ($\epsilon \ll 1$) perturbation changes these roots. In the lecture I claimed that a small perturbation could have a large effect when n is large. Let's investigate this claim.

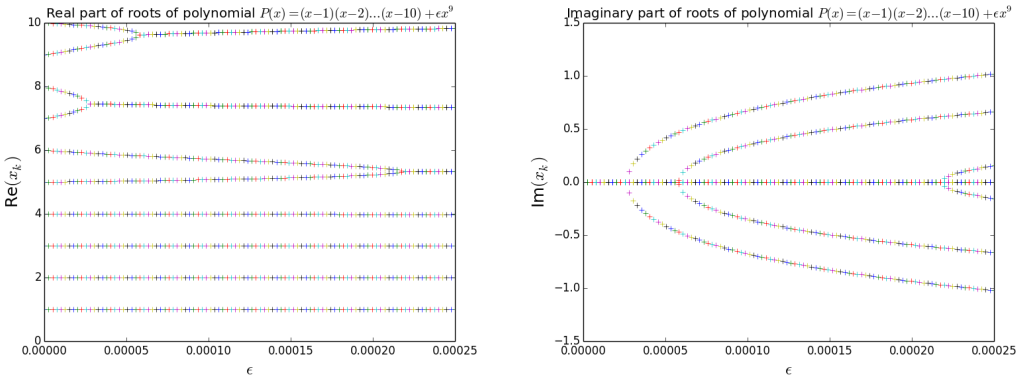
(a) In `poly_example.py` is a function to calculate the coefficients of Wilkinson's polynomial for a given n, using the `sympy` module. The function `wilkinson_polynom_coeffs(n)` returns a list of the coefficients $[a_0, a_1, a_2, \dots]$ of the unperturbed polynomial, where

$$p_n^{(0)}(x) = a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + \dots$$

Using `np.roots()`, find the roots of $p_n^{(0)}(x)$ for $n = 20$. What do you notice?

(b) Now calculate the roots of the perturbed polynomial $p_n(x)$ for $n = 10$ and (i) $\epsilon = 1 \times 10^{-5}$, (ii) $\epsilon = 4 \times 10^{-5}$ and (iii) $\epsilon = 8 \times 10^{-5}$. (*Hint:* To perturb the polynomial, change a_1, the second element in the list of coefficients.) What do you find?

(c) Make plots similar to the ones below, showing the real and imaginary parts of the roots as a function of ϵ for the case $n = 10$.



---

2. Row-sum norm and condition numbers.

The **row sum norm** of an $m \times n$ matrix A is defined as

$$\|A\| = \max_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}|$$

where a_{ij} is the element in the ith row and jth column of A. The **condition number** of a matrix is $C = \|A\| \, \|A^{-1}\|$, where A^{-1} is the matrix inverse.

(a) Write a function `row_sum_norm(A)` to compute the row sum norm of a numpy array or matrix. Write a function `condition_number(A)` to compute the condition number.

(b) Test your functions by computing the condition numbers of the following matrices:

$$\text{(i) } A_1 = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad \text{(ii) } A_2 = \begin{pmatrix} 1 & 1 \\ 1 & 1.001 \end{pmatrix}, \quad \text{(iii) } A_3 = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{pmatrix}.$$

Check your answers using `np.linalg.cond()`. N.B. Please read the docstring for this function (`?np.linalg.cond`), as you will need to specify an optional parameter to calculate the condition number corresponds to the row sum norm.

---

3. Hilbert matrices: `http://en.wikipedia.org/wiki/Hilbert_matrix`

Matrix A_3 in Q2(b)(iii) is an example of a Hilbert matrix.

(a) Let A_n be an $n \times n$ Hilbert matrix, and let C_n be its condition number. Write code to compute the condition number of C_n for a given n. Plot the condition number C_n as a function n, using logarithmic scales on both axes.

(b) Investigate the claim that $C_n \sim \alpha(1 + \sqrt{2})^{4n}/\sqrt{n}$, when n is large. Estimate α.

---

4. Iterative methods.

Consider the equation $A_3 x = b$ where $b = [1, 2, 3]^T$ and A_3 is a Hilbert matrix. The inverse of A_3 is

$$B \equiv A_3^{-1} = 3 \times \begin{pmatrix} 3 & -12 & 10 \\ -12 & 64 & -60 \\ 10 & -60 & 60 \end{pmatrix}$$

and thus the solution is $x = Bb = (27, -192, 210)^T$.

(a) Now consider a matrix equation $(A + \Delta A)x = b$, where ΔA is a 3×3 matrix with randomly-generated coefficients in the range $[-0.1, 0.1]$. Try solving this equation by applying an iterative method:

$$x_{k+1} = R x_k + c.$$

where $R = -B\Delta A$, $c = Bb$ and $x_0, x_1, \ldots$ are a sequence of estimates. Does this sequence always converge?