Scientific computing: Project 1

Tomás Fernández Bouvier

September 14, 2020

a)

For our 3 frequencies $\omega = [0.800, 1.146, 1.400]$ we obtain respectively $cond_\infty(\mathbf{E} - \omega\mathbf{S}) = [327.82, 1.53 \cdot 10^5, 227.19]$. If we have an algebraic equation of the form $A\vec{x} = \vec{b}$ the definition of condition number yields :

$$\frac{\|\delta x\|}{\|x\|} \leq cond(\mathbf{M})\frac{\|\delta b\|}{\|b\|}$$

taking the logarithms in both sides we can see how does the error associated to an input b (in our case z) is reflected at the output, i.e how many digits of accuracy we will loose or gain:

$$\log_{10}\left(\frac{\|\delta x\|}{\|x\|}\right) \leq \log_{10}(cond(\mathbf{M})) + \log_{10}\left(\frac{\|\delta b\|}{\|b\|}\right) = \begin{cases} -5.48 & \text{for } \omega = 0.800 \\ -2.18 & \text{for } \omega = 1.146 \\ -5.64 & \text{for } \omega = 1.400 \end{cases}$$

so we can trust 5, 2 and 5 decimal digits respectively.

b)

We computed the following bounds for perturbation in ω ($\delta\omega = \frac{1}{2}\omega$), therefore in the matrix:

$$\frac{\|\delta x\|}{\|x\|} \leq \begin{cases} 0.0052 & \text{for } \omega = 0.800 \\ 2.405 & \text{for } \omega = 1.146 \\ 0.0036 & \text{for } \omega = 1.400 \end{cases}$$

Again, we recall the previous formula to compute the number of digits that we can trust:

$$\log_{10}\left(\frac{\|\delta x\|}{\|x\|}\right) \leq \log_{10}(cond(\mathbf{A})) + \log_{10}\left(\frac{\|\mathbf{E}\|}{\|\mathbf{A}\|}\right) = \begin{cases} -0.48 & \text{for } \omega = 0.800 \\ 2.18 & \text{for } \omega = 1.146 \\ -0.64 & \text{for } \omega = 1.400 \end{cases}$$

where $\mathbf{E}$ is the perturbation in the matrix $\mathbf{A}$.

So we can trust the results to the units, hundreds and units respectively

c)

See code

d)

	$\omega = 0.800$	$\omega = 1.146$	$\omega = 1.400$
α	1.636	$2.6 \cdot 10^3$	2.706
$\Delta\alpha$	0.017	$5.2 \cdot 10^3$	0.014

Table 1: Table of polarizabilities for different frequencies

The perturbation is induced in the matrix so we have to compare our results with the ones from b). More or less they live up to our expectations and are $\leq$ than the bounds calculated before.

e)

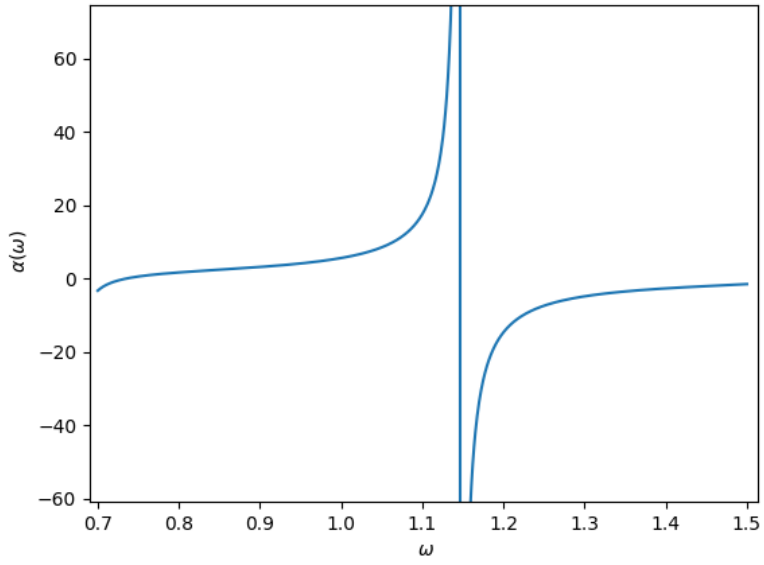


Figure 1: Representation of the solutions yielded by the algebraic equation as a function of ω

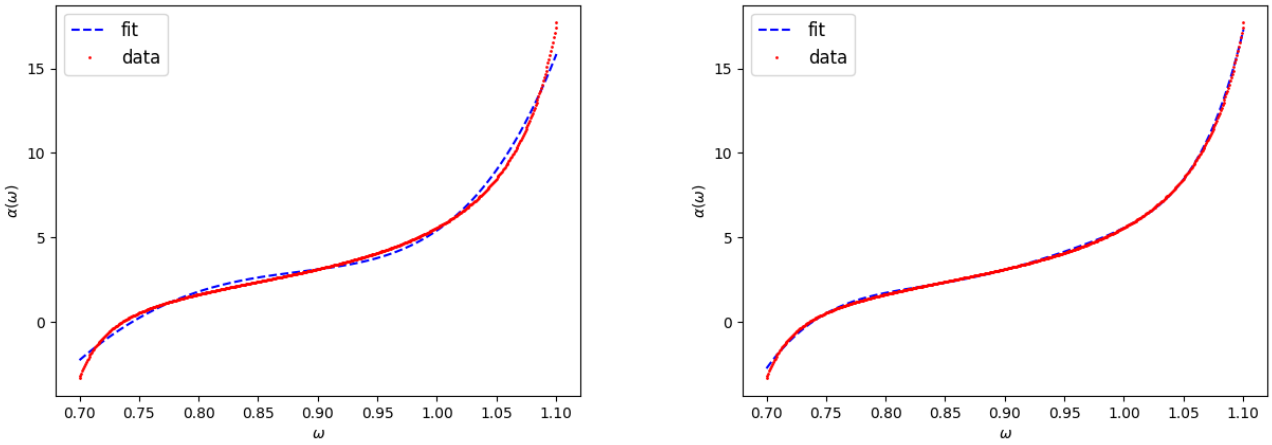
$\omega = 1.14630799$ is a singular point, i.e it makes the matrix $\mathbf{E} - \omega\mathbf{S}$ non invertible. Therefore what we see in the solution is a divergence around those frequencies.

f)

See code

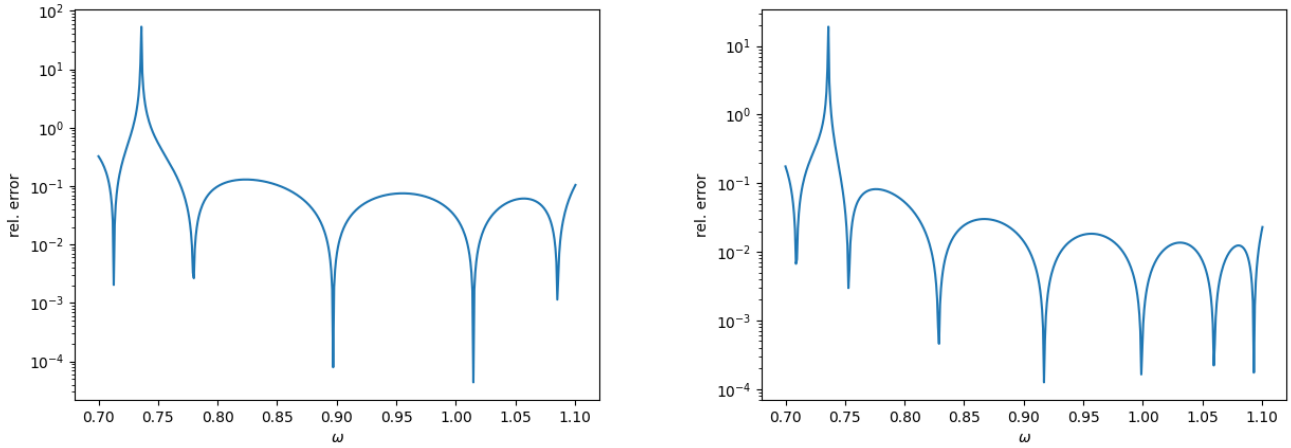
g)

We first perform the fit of our data to the polynomial $P(\omega) = \sum_i^n a_i \omega^{2i}$ Our data presents singularities though, which a polynomial cannot represent since the only solution for $P(x) = \infty$ is actually $x = \infty$ Therefore we have to choose a value far from where our data diverges i.e $\omega < 1.14630799$. We chose the above limit $\omega_p = 1.1$



(a) n=4 (b) n=6

Figure 2: Fit of data



(a) n=4 (b) n=6

Figure 3: Error

For n=4 we obtain

$$a_0 = -70.51 \qquad a_1 = 269.81 \qquad a_2 = -336.02 \qquad a_3 = 142.12$$

For n=6 we obtain

$$a_0 = -459.83 \qquad a_1 = 2838.55 \qquad a_2 = -6958.07$$
$$a_3 = 8481.19 \qquad a_4 = -5133.00 \qquad a_5 = 1236.74$$

From the error plot we can see that both fits fail to approximate the real values near $\omega = 0.75$ and work very well for higher values ($rel.error < 10^{-1}$). For n=6 the approximation is logically better than for n=4 but have to be aware that this could be overfitting.

h)

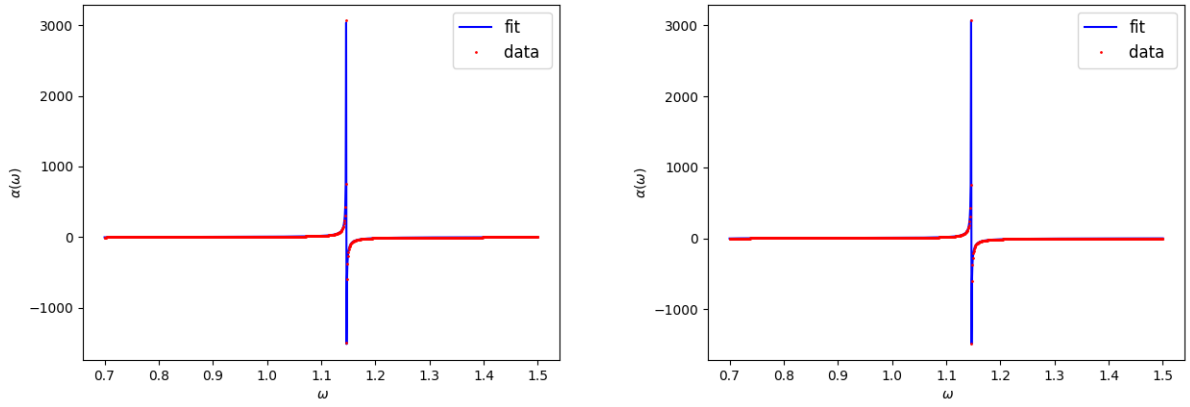
In order to solve the singularity problem, we look for a function that can represent it. We choose a rational one:

$$Q(\omega) = \frac{\sum_{i=0}^{n} a_i \omega^i}{1 + \sum_{j=1}^{n} b_j \omega^j}$$

In this case there will be the possibility to find an ω which makes the denominator equal to 0, so the function diverges as data does. The incovenient now is that this function is not linear anymore so we have to manipulate it to make it linear and use our least-squares algorithm. Assuming that $\alpha \approx Q(\omega)$ we can proceed to do the following manipulations:

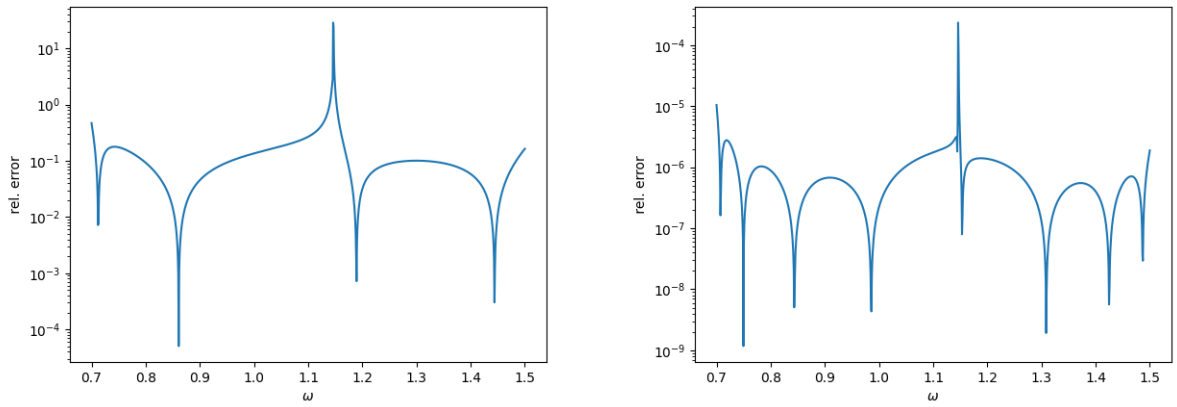
$$Q(\omega)(1 + \sum_{j=1}^{n} b_j \omega^j) = \sum_{i=0}^{n} a_i \omega^i \longrightarrow Q(\omega) = \sum_{i=0}^{n} a_i \omega^i - Q(\omega) \sum_{j=1}^{n} b_j \omega^j \approx \sum_{i=0}^{n} a_i \omega^i - \sum_{j=1}^{n} \alpha(\omega) b_j \omega^j$$

This function is linear and can be implemented in our algorithm to return a vector of coefficients $\vec{x} = (\vec{a}, \vec{b})$.



(a) n=2 (b) n=4

Figure 4: Fit of data



(a) n=2 (b) n=4

Figure 5: Error

For n=2 we obtain

$$a_0 = 1.93 \qquad a_1 = -3.45 \qquad a_2 = 1.14 \qquad b_1 = -2.39 \qquad b_2 = 1.33$$

and for n=4:

$$a_0 = 1.97 \qquad a_1 = -3.89 \qquad a_2 = 1.61 \qquad a_3 = 0.055 \qquad a_4 = -0.0088$$

$$b_1 = -2.00 \qquad b_2 = -0.17 \qquad b3 = 1.92 \qquad b_4 = -0.80$$

The n=4 case fits really well our data and the relative error is very small. We can see that the biggest error overcomes at ω_p and it is still very little for the second approximation (n=6)