Project 4: Numerical integration and curve fitting.

Cruz Perez Gustavo Hazael
Superior School of Physics and Mathematics, Mexico City, Mexico.
Email: haxaleg@gmail.com

1) Equations solved.

Gamma function

The Gamma function is defined as.

$$\Gamma(x) = \int_0^\infty t^{(x-1)} e^{-t}\, dt$$

Evaluating in $x = \frac{3}{2}$

$$\Gamma\left(\frac{3}{2}\right) = \int_0^\infty \sqrt{t}\, e^{-t}\, dt$$

Elliptic integrals

Complete elliptic functions of the first and second type are defined as.

$$K(x) = \int_0^{\frac{\pi}{2}} \frac{dt}{\sqrt{1 - x^2 \sin^2 t}}$$

$$E(x) = \int_0^{\frac{\pi}{2}} \sqrt{1 - x^2 \sin^2 t}\, dt$$

Evaluating in $x = \frac{1}{2}$

$$K\left(\frac{1}{2}\right) = \int_0^{\frac{\pi}{2}} \frac{dt}{\sqrt{1 - \frac{1}{4}\sin^2 t}}$$

$$E\left(\frac{1}{2}\right) = \int_0^{\frac{\pi}{2}} \sqrt{1 - \frac{1}{4}\sin^2 t\, dt}$$

Breit Wigner resonance formula adjustment

Problem statement.

Determine the values of the effective sections whose energies are between the values of the following graph.

Table 1: Experimental data of the dispersion of effective sections

i	$E_i(MeV)$	$f(E_i)(MeV)$	$\sigma_i(MeV)$
1	0	10.6	9.34
2	25	16.0	17.9
3	50	45.0	41.5
4	75	83.5	85.5
5	100	52.8	51.5
6	125	19.9	21.5
7	150	10.8	10.8

8	175	8.25	6.29
9	200	4.7	4.14

The following equation is Breit Wigner's formula gives the effective sections measured for the resonant scattering of a neutron of a nucleus, where E_r, f_r y Γ are unknown parameters in the fit, but theoretically it can be obtained that , $E_r = 78\ MeV\ y\ \Gamma = 55\ MeV$we will compare these values obtaining them in different ways.

$$f(E) = \frac{f_r}{(E - E_r)^2 + \frac{\Gamma^2}{4}}$$

In order to find the values between the given energies, you can do some kind of interpolation, as shown below, or try to find theE_r, f_r y Γunknowns, proposed an appropriate system of equations, for this it is proposed and the following variable change.

$$a_1 = f_r; \quad a_2 = E_r; \quad a_3 = \frac{\Gamma^2}{4}; \quad x = E; \; f = g$$

We can write the function as.

$$g(x) = \frac{a_1}{(x - a_2)^2 + a_3} \qquad \ldots(1)$$

To find the values of the unknowns we minimize χ^2, this generates a system of equations. $N_D = 9$

$$\chi^2 = \sum_{i=1}^{N_D} \left(\frac{y_i - g\left(x_i : \left\{a_1, \ldots, a_{M_p}\right\}\right)}{\sigma_i} \right)^2$$

$$\chi^2 = \sum_{i=1}^{9} \left(\frac{y_i - g(x_i, a_1, a_2, a_3)}{\sigma_i} \right)^2$$

When takingpartial derivatives we have the following system of equations

$$\frac{\partial \chi^2}{\partial a_1} = \sum_{i=1}^{9} \left[\left(\frac{y_i - g(x_i)}{\sigma_i^2} \right) \frac{\partial g(x_i)}{\partial a_1} \right] = 0$$

$$\frac{\partial \chi^2}{\partial a_2} = \sum_{i=1}^{9} \left[\left(\frac{y_i - g(x_i)}{\sigma_i^2} \right) \frac{\partial g(x_i)}{\partial a_2} \right] = 0$$

$$\frac{\partial \chi^2}{\partial a_3} = \sum_{i=1}^{9} \left[\left(\frac{y_i - g(x_i)}{\sigma_i^2} \right) \frac{\partial g(x_i)}{\partial a_3} \right] = 0$$

Writing the corresponding partial derivatives we have to.

$$\frac{\partial g(x)}{\partial a_1} = \frac{1}{(x - a_2)^2 + a_3}$$

$$\frac{\partial g(x)}{\partial a_2} = -\frac{2a_1(x - a_2)}{((x - a_2)^2 + a_3)^2}$$

$$\frac{\partial g(x)}{\partial a_3} = -\frac{a_1}{((x-a_2)^2+a_3)^2}$$

Therefore our system of equations to be solved is as follows.

$$\sum_{i=1}^{9}\left[\left(\frac{1}{\sigma_i^2}\right)\left(\frac{y_i((x_i-a_2)^2+a_3)-a_1}{((x_i-a_2)^2+a_3)^2}\right)\right]=0$$

$$\sum_{i=1}^{9}\left[\left(\frac{1}{\sigma_i^2}\right)\left(\frac{(y_i((x_i-a_2)^2+a_3)-a_1)(x_i-a_2)}{((x_i-a_2)^2+a_3)^3}\right)\right]=0 \qquad \ldots(2)$$

$$\sum_{i=1}^{9}\left[\left(\frac{1}{\sigma_i^2}\right)\left(\frac{y_i((x_i-a_2)^2+a_3)-a_1}{((x_i-a_2)^2+a_3)^3}\right)\right]=0$$

2) Numerical method or algorithm used.

Trapeze rule

It tells us that for a closed interval , $[a,b]$ we take the partition $h=\frac{b-a}{N-1}$, for N subintervals, then the approximation of the integral defined for a function $f[a,b]\to\mathbb{R}$, is.

$$\int_a^b f(x)dx \approx h\left(\frac{1}{2}f(x_1)+f(x_2)+\cdots+f(x_{N-1})+\frac{1}{2}f(x_N)\right)$$

where $x_i=a+(i-1)h$.

Simpson's rule

The same partition of the interval is taken as in the trapezoid rule, $h=\frac{b-a}{N-1}$ but it must be an odd number$N$ and every three extreme points of subintervals the function approximates a parabola, so the approximate value of the integral is given by.

$$\int_a^b f(x)dx \approx \frac{h}{3}(f(x_1)+4f(x_2)+2f(x_3)+\cdots+4f(x_{N-1})+f(x_N))$$

where $x_i=a+(i-1)h$.

Lagrange interpolation.

For an dataset.

Table 2: data to interpolate

x	x_1	x_2	...	x_{n-1}	x_n
$g(x)$	$g(x_1)$	$g(x_2)$	...	$g(x_{n-1})$	$g(x_n)$

The interpolant polynomialof Lagrange is constructed, which passes through all points $g(x_i)$ and approximates the function $g(x)$, as follows.

$$g(x)\approx g_1L_1(x)+g_2L_2(x)+\cdots+g_nL_n(x)$$

Where.

$$L_i(x)=\prod_{j(\neq i)=1}^{n}\frac{x-x_j}{x_i-x_j}$$

Cubic splines**.**

It consists of fitting to a cubic polynomial in each subinterval $[x_i,x_{i+1}]$, that is.

$$g(x) \approx g_i(x)\ ;\ \forall\ x\ , x_i \leq x \leq x_{i+1}$$

Where the cubic polynomial $g_i(x)$ is of the form.

$$g_i(x) = g_i + g_i{}'(x - x_i) + g_i''(x - x_i)^2\ + g_i'''(x - x_i)^3$$

To determine the coefficients continuity conditions are applied on the polynomial, its first and second derivatives, but this is not sufficient to determine the coefficients, so additional conditions are required that are taken from the form of conditions in the frontera, some ways of these are.

1. Natural Spline
2. Give values of g' at the ends
3. Give values of g''' at the ends

Newton-Raphare in several variables.

For a system $\vec{\boldsymbol{f}}(\vec{\boldsymbol{y}})$ of n homogeneous nonlinear equations, with n unknowns.

$$\vec{\boldsymbol{f}}(\vec{\boldsymbol{y}}) = \vec{0}$$

Or

$$\begin{pmatrix} f_1(x_1, x_2, \dots, x_{n-1}, x_n) \\ f_2(x_1, x_2, \dots, x_{n-1}, x_n) \\ \vdots \\ f_{n-1}\,(x_1, x_2, \dots, x_{n-1}, x_n) \\ f_n(x_1, x_2, \dots, x_{n-1}, x_n) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

Its solutions can be found by extending the one-dimensional Newton-Raphson method, for this we need to approximate in each iteration by steps Δx_j, for which.

$$f_i(x_1 + \Delta_1, x_2 + \Delta_2, \dots, x_{n-1} + \Delta_{n-1}, x_n + \Delta_n) = 0$$

We approximate using Taylor.

$$f_i(x_1 + \Delta_1, x_2 + \Delta_2, \dots, x_{n-1} + \Delta_{n-1}, x_n + \Delta_n) = f_i(x_1, x_2, \dots, x_{n-1}, x_n) + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j} \Delta x_j$$

Rewriting the above for equations n matrix.

$$\begin{pmatrix} f_1(\vec{\boldsymbol{y}}) \\ f_2(\vec{\boldsymbol{y}}) \\ \vdots \\ f_{n-1}\,(\vec{\boldsymbol{y}}) \\ f_n(\vec{\boldsymbol{y}}) \end{pmatrix} + \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_{n-1} \\ \Delta x_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

Where the Jacobian matrix of the function that we will denote as $\vec{\boldsymbol{f}}(\vec{\boldsymbol{y}})$ by which the above system$\boldsymbol{J_f}$, can be written as follows.

$$\vec{\boldsymbol{f}}(\vec{\boldsymbol{y}}) + \boldsymbol{J_f}\,\overrightarrow{\boldsymbol{\Delta x}} = \vec{\boldsymbol{0}}$$

And that has as a solution.

$$\overrightarrow{\boldsymbol{\Delta x}} = -\boldsymbol{J}_{\boldsymbol{f}}^{-1}\vec{\boldsymbol{f}}(\vec{\boldsymbol{y}})$$

3) Visualization.

Due to the large amount of code, I preferred only to show the results, the programs attach them to the page.

Gamma function

Theupper limit value was taken as 10, because the form of the integral demanded a, $h$much less than 1, and since the number N has to be kept at 1000 to avoid increases in errors, and the smaller h you have a better approximation, and since the function $\Gamma\left(\frac{3}{2}\right)$ remains very close to zero after $t = 5$, It is permissible to use that upper limit value

$$\Gamma\left(\frac{3}{2}\right) = \int_0^\infty \sqrt{t}\, e^{-t}\, dt \approx \int_0^{10} \sqrt{t}\, e^{-t}\, dt$$

The program gives us the following values

Table 3: Data obtained for the function Γ

	Trapeze rule	Simpson's rule	Scipy(taking the integral of 0 a inf)
$\Gamma\left(\frac{3}{2}\right)$	0.8858685514694353	0.885995168730758	0.8862269254536111
Error	0.040438173777262214 %	0.026150945790155776 %	0

Elliptic integrals

The program gives us the following approximations.

Table 4: Data obtained for the elliptic integral $K\left(\frac{1}{2}\right)$

	Trapeze rule	Simpson's rule	Scipand
$k\left(\frac{1}{2}\right)$	1.685750354812594	1.6857503548125954	1.6857503548125963
Error	1.3171855743118965e-13%	5.2687422972475864e-14%	0

Table 5: Data obtained for the elliptic integral $E\left(\frac{1}{2}\right)$

	Trapeze rule	Simpson's rule	Scipand
$E\left(\frac{1}{2}\right)$	1.4674622093394256	1.4674622093394263	1.4674622093394272

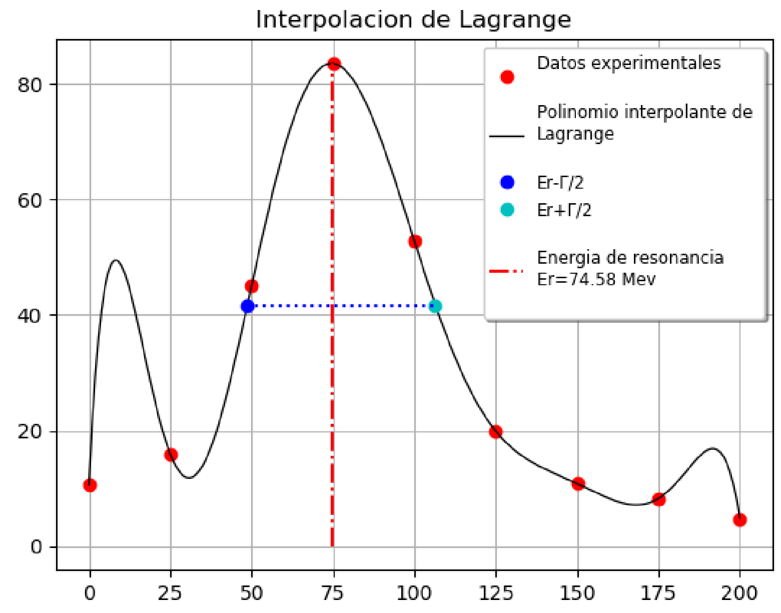
Error	1.0591838240079022e-13%	6.052478994330869e-14%	0

Lagrange interpolation.

Using Lagrange's algorithm to construct a polynomial that passes through the data from lto table 1, we have the following interpolating polynomial.

$$-1.155\times 10^{-13}\,x^8 + 9.976\times 10^{-11}\,x^7 - 3.54310^{-8}\,x^6 + 6.631\times 10^{-6}\,x^5 - 0.0006943\,x^4 + 0.03955\,x^3 - 1.093\,x^2 + 11.41\,x + 10.6$$

Graphing this function together with its most important parameters.



Graph 1: Lagrange polynomial, we can see that you will have two different values for Γ because the graph is not symmetric.

We can obtain from the graph the following values.

$$E_r = 74.58\ MeV$$
$$\Gamma = 57.68 MeV$$

Comparing you have the following errors.

$$e_\Gamma = 4.87\% ;\quad e_{E_r} = 4.38\%$$

A more convenient form of the Lagrange algorithm is to apply it every three points of the data that you want to interpolate, let's see that in this way the values of $E_r\ y\ \Gamma$

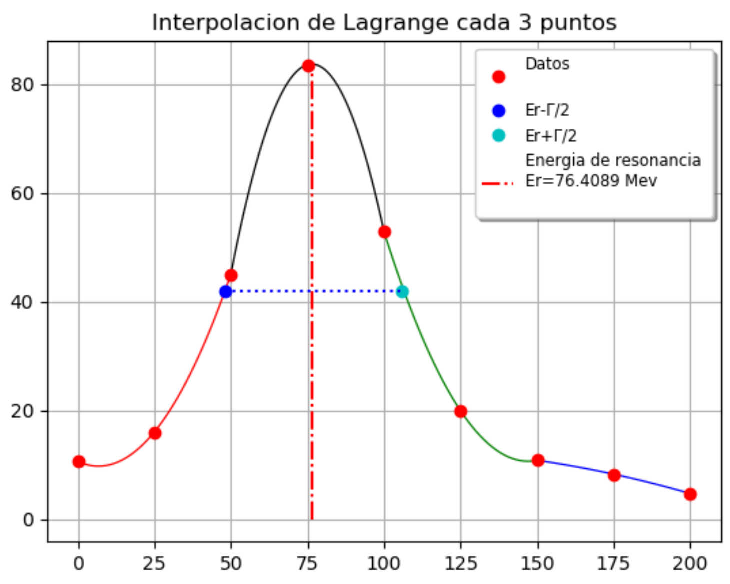


Figure 2: Four different polynomials interpolating the data using Lagrange.

From here we can get.

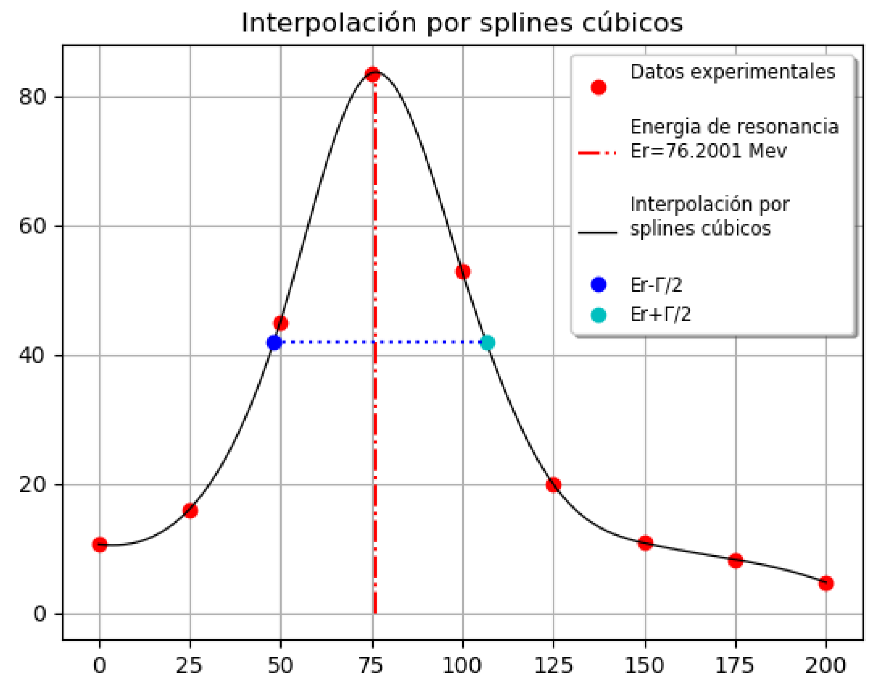
$$E_r = 76.4089\ MeV$$
$$\Gamma = 57.5459\ MeV$$

With the following errors.

$$e_{E_r} = 2.04\% \qquad e_\Gamma = 4.62\%$$

Interpolation via cubic splines.

Using the interpolate subpackage of the Scipy library, the following adjustment is made by means of cubic splines.



Graph 3: Graph generated with the interpolate *subpackage of the* Scipy *library*

From where we can obtain the following data.

$$E_r = 76.2001\ MeV$$
$$\Gamma = 58.1082\ MeV$$

With the following errors.

$$e_{E_r} = 2.3075\% \qquad e_\Gamma = 5.661\%$$

Breit Wigner resonance formula adjustment

A program was made that can solve a system of 3 nonlinear equations with three variables, using the multivariable Ne wton-Raphson algorithm, which gives the following solutions for equation (1).

$$E_r = 78.2183$$
$$\Gamma = 56.1648 \qquad \ldots(3)$$
$$f_r = 68937.0956$$

If we compare with the reference values we have, that this method is the one that presents a lower amount of error.

$$e_{E_r} = 0.2798\% \qquad e_\Gamma = 2.1178\%$$

Let's now look at the graph of equation (1) with the values provided by the program

$$g(x) = \frac{68937.0956}{(x - 78.2183)^2 + 788.6223}$$

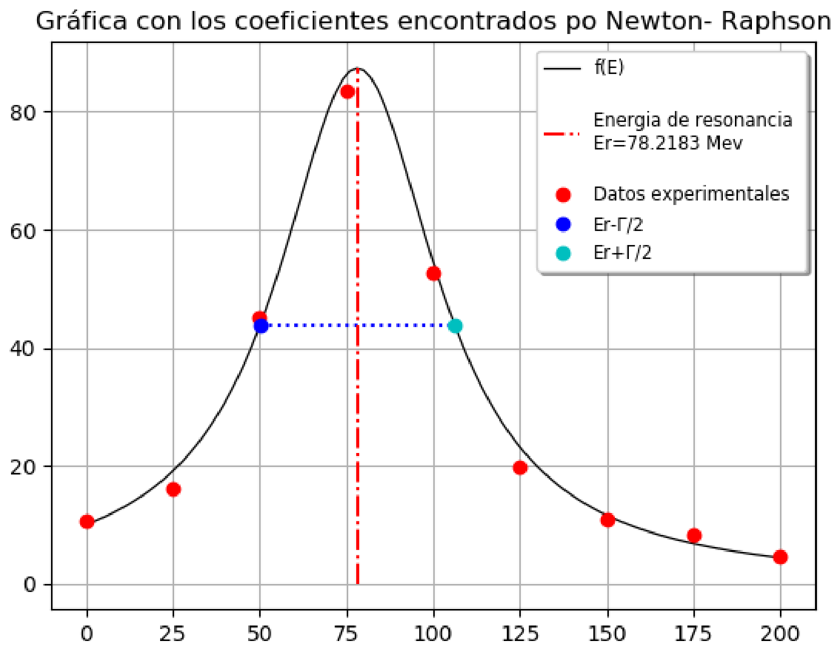


Figure 4: Breit-Wigner resonance formula in black, with parameters obtained by the algorithm

N-R

4) Discussion.

For the function it is important to consider the fact that the Γ form of the function is very sensitive to the type of step we take if we want to approximate its integral using the trapezium and Simpson methods, because it is a very small area that the largest partof itis concentrated between zero and five on the x-axis. So if you chose a larger to one, a good part of the area under the curve was not added in the hs iterations, giving very large errors when comparing, and depends on the extreme value and the number of subintervals, as this can not be made larger than 1000, to prevent rounding errors from dominating, One way in which a good error could be obtained was to leave the value of the upper limit of the integral as 10, although this sounds absurd you get an error of 0.02%. h

In interpolation methods we see that by applying the method of, Lagrange every 3 points you get the best error, while if we want something even more effective, the method that delivered the best errors was to find the coefficients of the Breit Wigner resonance formula by solving a system of n-linear equations, We see here that the best fit does not necessarily have to go through the experimental points, as seen in this case.

The forms that have the functions of the system (2) are extensive, this makes them very sensitive when solving them computationally, because they can throw different values with very similar seeds, to solve this, seeds were taken for the values of , $E_r = 90 \ \ y \, \Gamma = 90$because we only have information of these two, and for f_R, a cycle was made for that goes through values from 0 to 100,000 in steps of 500, so you can see a predominance of solutions (3) for the system (2), this part is on line 90 of the 5.py program, and can be executed optionally.

5) **Criticizes.**

One way to avoid the problem with the function $\Gamma$is to obtain its theoretical value and make approximations around the value of π. The program, which finds the solutions using Newton - Raphson, is very specific (it only works for a system of three equations with three unknowns) because of the complexity of implementation it was not possible to make a program that takes an arbitrary amount of equations and variables, $n$and finds the solutions, the next step would be to try to generalize the program.