**Numerical Methods and its Applications**


Alejandro Arango

Sebastián Duque

Pablo Martínez

Daniel Hurtado

Gabriel Castaño

Department of Engineering, Universidad EAFIT


ST-041 (001): Procesos Numéricos

Julián Esteban Rendón Roldan

2/06/2022

## Introduction

The following practice is focused on the implementation, analysis, and evaluation of different numerical methods in solving problems directly related to the student's professional career. In this work, situations like those that the student may find in the workplace are presented, including cases of heat transfer (nonlinear equations), stress analysis using gauges (system of equations), and stress-strain curves (interpolation). All of these are to educate and train the student in the use of these strategies in real and recurrent cases.

This exercise requires a process of identification, application, and justification of numerical methods that relate to the problem faced. Using Matlab and Python, these problems can be solved efficiently and quickly. By programming various techniques developed throughout history, we can find satisfactory results that allow us to make a more detailed analysis of the situation and find the answers we are looking for.

### Objectives

I. Using the knowledge acquired during the semester, the different numerical methods learned will be implemented to solve problems of nonlinear equations, systems of equations, interpolation, and differential equations, focused on the professional interests of the student.

II. Compare the results obtained from the application of each method to identify which of them has greater accuracy and efficiency.

## Chapter 1: Numerical solutions for nonlinear equations

### Analysis of the cooling process of industrial furnaces

Industrial furnaces are frequent tools in the metal industry, being one of the most used the "Clam Shell Furnace." These are recurring in companies specialized in forge, ferrous metals, aluminum heat treatment, transformers and engines, and pipe, rod and wire, performing austenized and temperate processes of forged parts and steel molds, iron casting malleability, annealing, among others (Nutec Bickley, 2022).

For this case, the cooling process of a clamshell type furnace after carrying out a hot forging procedure of steel at a temperature of 1200°C will be analyzed. For ease, let's speculate that after an hour of finishing the process, the furnace temperature dropped to 900°C, while the ambient temperature was around 23.3°C. Based on these values, Newton's law of cooling can be used to set up an equation that describes the change in temperature of the body as a function of time.

Initially, table 1.1 allows us to view more clearly the change in the furnace temperature:

Table 1.1. Temperature of the furnace at a given time.

| t $[minutes]$ | 0 | 60 |
|---|---|---|
| T [°C] | 1200 | 900 |

On the other hand, Newton's formulation on the cooling of bodies indicates that: "the rate of loss of heat from a body is directly proportional to the difference in the temperature of the body and its surroundings." This relationship can be expressed mathematically as follows:

$$\frac{dT}{dt} = k(T_t - T_s) \text{ (Equation 1.1)}$$

Where $T_t$ is the temperature of the body at time $t$, $T_s$ is the temperature of the surrounding, and $k$ is a constant that depends on the area and nature of the body under consideration. Solving formula 1.1 using the method of separable variables leads to the following formula:

$$T(t) = T_s + Ce^{kt} \text{ (Equation 1.2)}$$

Taking into consideration the initial data, we can solve the equation to find the value of C and k. From table 1 it is known that the initial temperature, when $t = 0$, is $T(0) = 1200°C$. By substituting these values in the expression, we obtain that C is given by the formula:

$$C = T_0 - T_s \text{ (Equation 1.3)}$$

From this we can determine that C is equal to $1176.7$ °C and, using the same strategy with the second data pair, we can find the constant k, which would have a value of $-4.90501 \times 10^{-3}$.

Finally, due to the possible variations in the room temperature that may occur during the cooling process, it was decided to add one more element to the equation:

$$T(t) = 23.3 + \sin((-4.90501 \times 10^{-3})t) + 1176.7e^{(-4.90501\times10^{-3})t} \text{ (Equation 1.4)}$$

This $\sin(kt)$ is expected to allow the function to consider small changes in temperature around the furnace.

**Numerical Analysis**

In this case, we are interested in knowing the instant in which the temperature reaches 100°C, therefore, a certain equation is needed so it can be used in each method.

For 100°C, the following formula shall be used:

$$0 = \sin((-4.90501 \times 10^{-3})t) + 1176.7e^{(-4.90501\times10^{-3})t} - 76.7 \text{ (Equation 1.5)}$$

Equation 1.5 is the one that will be employed in every method. Each time, a tolerance of five correct decimal places (Tol=0.5e-5) will be set, considering a maximum number of 100 iterations per method. The function that will be implemented in Matlab is the following:

$f = \sin\left((-4.90501 \times 10^{-3}) * x\right) + 1176.7 * e\left((-4.90501 \times 10^{-3}) * x\right) - 76.7$ ; where x=t

The purpose of the numerical analysis is to compare how each method behaves and to determine the most efficient one. Then, ascertain the time in which the furnace reaches this temperature. The following plot serve as a guide for establishing a initial value for the methods that require this parameter.

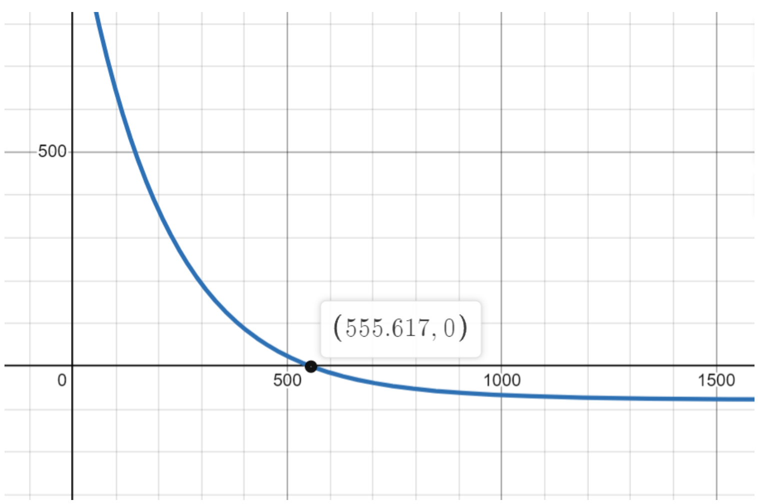



Figure 1.1. Equation 1.5 plot.

Figure 1.1 illustrate the positive root of the function. Equation 1.5, which shows the time when the furnace reaches a temperature of 100°C. The information given by the plot is crucial for the analysis, as every numerical method of this chapter will be run to find the approximate value of each root.

**Bisection method**

For the root of the function, the bisection method works with the interval [550, 560], where $x_i = 550$ and $x_s = 560$. The value obtained for this root, within 20 iterations, is 555.6174802780151 with an error of 4.768371582031250e-6. (Table 1.2)

```
n                   Xa                  F                   E
1.0e+02 *

                  0                  0   0.002311162773095   0.010000050000000
0.010000000000000   5.550000000000000  -0.007002244238391   0.025000000000000
0.020000000000000   5.575000000000000  -0.002360064037754   0.012500000000000
0.030000000000000   5.562500000000000  -0.000028092671905   0.006250000000000
0.040000000000000   5.556250000000000   0.001140623135339   0.003125000000000
0.050000000000000   5.553125000000000   0.000556037428739   0.001562500000000
0.060000000000000   5.554687500000000   0.000263915449637   0.000781250000000
0.070000000000000   5.555468750000000   0.000117897159416   0.000390625000000
0.080000000000000   5.555859375000000   0.000044898686736   0.000195312500000
0.090000000000000   5.556054687500000   0.000008402118203   0.000097656250000
0.100000000000000   5.556152343750000  -0.000009845499148   0.000048828125000
0.110000000000000   5.556201171875000  -0.000000721746048   0.000024414062500
0.120000000000000   5.556176757812500   0.000003840172184   0.000012207031250
0.130000000000000   5.556164550781250   0.000001559209595   0.000006103515625
0.140000000000000   5.556170654296875   0.000000418730905   0.000003051757813
0.150000000000000   5.556173706054688  -0.000000151507788   0.000001525878906
0.160000000000000   5.556175231933594   0.000000133611504   0.000000762939453
0.170000000000000   5.556174468994141  -0.000000008948156   0.000000381469727
0.180000000000000   5.556174850463867   0.000000062331671   0.000000190734863
0.190000000000000   5.556174659729004   0.000000026691757   0.000000095367432
0.200000000000000   5.556174755096436   0.000000008871800   0.000000047683716
```

Table 1.2

**Newton's method**

Newton's method uses an initial value of $x_0 = 550$ for the root of the function. The result is 555.6174826519851 after 4 iterations, with an error of 0.000000000000006. Its worth noting that the answer given by this method is an exact root of the function. (Table 1.3)

```
n                   Xn                  Fx                  Error
1.0e+02 *

                  0   5.500000000000000   0.021290148851047   0.010000050000000
0.010000000000000   5.555394972994905   0.000291496993060   0.055394972994904
0.020000000000000   5.556174674803485   0.000000056698163   0.000779701808581
0.030000000000000   5.556174826519846   0.000000000000002   0.000000151716361
0.040000000000000   5.556174826519851                   0   0.000000000000006
```

Table 1.3

**Regula falsi method**

For the root, the regula falsi method works within the interval [550, 560], $x_i = 550$ and $x_s = 560$. The value obtained is 555.6174826543078 with an error of 0.000000001641951 within 4 iterations. (Table 1.4)

```
n                   Xa                  F                   E
1.0e+02 *

                  0                   0  -0.000229256120914   0.010000050000000
0.010000000000000   5.556788377823485  -0.000003198437167   0.000604992713776
0.020000000000000   5.556183385109709  -0.000000044613671   0.000008439209895
0.030000000000000   5.556174945899815  -0.000000000622296   0.000000117714786
0.040000000000000   5.556174828185029  -0.000000000008680   0.000000001641951
```

Table 1.4

**Secant method**

Although the secant method does not rely on intervals, the initial values remain the same as the values used in the previous method. Consequently, for the root, the initial parameters are $x_0 = 550$ and $x_1 = 560$. The solution attained was 555.6174826519853 at the 4th iteration with an error of 0.000000001030887. (Table 1.5)

```
n                   X1                  F                   E
1.0e+02 *

                  0                   0  -0.016200178689604   0.010000050000000
0.010000000000000   5.556788377823485  -0.000229256120914   0.043211622176515
0.020000000000000   5.556168092499579   0.000002516585473   0.000620285323906
0.030000000000000   5.556174827550739  -0.000000000385255   0.000006735051161
0.040000000000000   5.556174826519853  -0.000000000000001   0.000000001030887
```

Table 1.5

### Fixed point method

The fixed-point method requires a function $g(x)$ to run properly. Taking this into consideration, the following function is the one used in the code:

$$x = \sin((-4.90501 \times 10^{-3})t) + 1176.7e^{(-4.90501\times10^{-3})t} + x - 76.7$$

The value obtained is 555.6174756328498 with an error of 0.000000041883770 within 29 iterations. (Table 1.6)

| n<br>1.0e+02 * | Xn | F | E |
|---|---|---|---|
| 0 | 5.500000000000000 | 0.021290148851047 | 0.010000050000000 |
| 0.010000000000000 | 5.521290148851047 | 0.013150928689528 | 0.021290148851047 |
| 0.020000000000000 | 5.534441077540575 | 0.008166353529516 | 0.013150928689529 |
| 0.030000000000000 | 5.542607431070093 | 0.005087493958676 | 0.008166353529517 |
| 0.040000000000000 | 5.547694925028768 | 0.003175751352515 | 0.005087493958675 |
| 0.050000000000000 | 5.550870676381283 | 0.001984847736461 | 0.003175751352516 |
| 0.060000000000000 | 5.552855524117743 | 0.001241489544183 | 0.001984847736460 |
| 0.070000000000000 | 5.554097013661927 | 0.000776905373407 | 0.001241489544184 |
| 0.080000000000000 | 5.554873919035333 | 0.000486321998800 | 0.000776905373406 |
| 0.090000000000000 | 5.555360241034134 | 0.000304481892862 | 0.000486321998801 |
| 0.100000000000000 | 5.555664722926995 | 0.000190655874745 | 0.000304481892861 |
| 0.110000000000000 | 5.555855378801740 | 0.000119390821501 | 0.000190655874745 |
| 0.120000000000000 | 5.555974769623240 | 0.000074767307452 | 0.000119390821500 |
| 0.130000000000000 | 5.556049536930691 | 0.000046823631913 | 0.000074767307451 |
| 0.140000000000000 | 5.556096360562603 | 0.000029324209640 | 0.000046823631913 |
| 0.150000000000000 | 5.556125684772244 | 0.000018365064499 | 0.000029324209640 |
| 0.160000000000000 | 5.556144049836742 | 0.000011501690681 | 0.000018365064499 |
| 0.170000000000000 | 5.556155551527424 | 0.000007203322197 | 0.000011501690682 |
| 0.180000000000000 | 5.556162754849622 | 0.000004511336343 | 0.000007203322198 |
| 0.190000000000000 | 5.556167266185964 | 0.000002825389527 | 0.000004511336342 |
| 0.200000000000000 | 5.556170091575492 | 0.000001769505550 | 0.000002825389528 |
| 0.210000000000000 | 5.556171861081042 | 0.000001108219594 | 0.000001769505550 |
| 0.220000000000000 | 5.556172969300636 | 0.000000694064618 | 0.000001108219594 |
| 0.230000000000000 | 5.556173663365255 | 0.000000434684449 | 0.000000694064619 |
| 0.240000000000000 | 5.556174098049703 | 0.000000272237767 | 0.000000434684448 |
| 0.250000000000000 | 5.556174370287470 | 0.000000170499335 | 0.000000272237768 |
| 0.260000000000000 | 5.556174540786807 | 0.000000106781750 | 0.000000170499336 |
| 0.270000000000000 | 5.556174647568557 | 0.000000066876171 | 0.000000106781750 |
| 0.280000000000000 | 5.556174714444728 | 0.000000041883771 | 0.000000066876171 |
| 0.290000000000000 | 5.556174756328499 | 0.000000026231320 | 0.000000041883770 |

Table 1.6

## Chapter 2: Numerical solutions for systems of equations

**Analytical dimensional synthesis of mechanisms**

Dimensional synthesis is a method used to find the dimensions of a mechanism from a set of initial data. It is usually used to conceive mechanism designs of 4 bars or more, basic type, crank-rod-slider, and inverted slider, among others, based on a trajectory described by 3 or more precision points.

In our case, the dimensional synthesis of a basic 5-bar mechanism used in an assembly line will be carried out. This must execute a particular movement determined by a set of angles that describe the inclination of the bars at a given moment. Using this set of data and based on 6 expressions formulated from the cinematic restriction equation of the mechanism, we can solve the system of equations to obtain its dimensions.

In this order of ideas, after a series of mathematical procedures, the following system of equations is obtained:

$K_1 Cos(\phi_1 - \varphi_1) - K_2 Cos(\varphi_1 - \psi_1) - K_3 Cos\phi_1 - K_4 Cos\varphi_1 + K_5 Cos\psi_1 + K_6 = Cos(\phi_1 - \psi_1)$ (Equation 2.1.)

$K_1 Cos(\phi_2 - \varphi_2) - K_2 Cos(\varphi_2 - \psi_2) - K_3 Cos\phi_2 - K_4 Cos\varphi_2 + K_5 Cos\psi_2 + K_6 = Cos(\phi_2 - \psi_2)$ (Equation 2.2.)

$K_1 Cos(\phi_3 - \varphi_3) - K_2 Cos(\varphi_3 - \psi_3) - K_3 Cos\phi_3 - K_4 Cos\varphi_3 + K_5 Cos\psi_3 + K_6 = Cos(\phi_3 - \psi_3)$ (Equation 2.3.)

$K_1 Cos(\phi_4 - \varphi_4) - K_2 Cos(\varphi_4 - \psi_4) - K_3 Cos\phi_4 - K_4 Cos\varphi_4 + K_5 Cos\psi_4 + K_6 = Cos(\phi_4 - \psi_4)$ (Equation 2.4.)

$K_1 Cos(\phi_5 - \varphi_5) - K_2 Cos(\varphi_5 - \psi_5) - K_3 Cos\phi_5 - K_4 Cos\varphi_5 + K_5 Cos\psi_5 + K_6 = Cos(\phi_5 - \psi_5)$ (Equation 2.5.)

$K_1 Cos(\phi_6 - \varphi_6) - K_2 Cos(\varphi_6 - \psi_6) - K_3 Cos\phi_6 - K_4 Cos\varphi_6 + K_5 Cos\psi_6 + K_6 = Cos(\phi_6 - \psi_6)$ (Equation 2.6.)

| | $\phi$ | $\varphi$ | $\psi$ |
|---|---|---|---|
| 1 | 15° | 20° | 35° |
| 2 | 30° | 40° | 60° |
| 3 | 45° | 60° | 90° |
| 4 | 60° | 90° | 110° |
| 5 | 75° | 110° | 145° |
| 6 | 90° | 140° | 180° |

The unknowns represent algebraic expressions from which the measures of the 5 bars can be found. However, the value of the K variables must be found first, so a set of 6 precision points is then proposed. These data are evidenced in table 2.1:

Table 2.1. Set of angles for the mechanism.

Having these values, we can replace them in the previous equations to build the matrix of coefficients $A$ and the vector of independent terms $b$:

$$A = \begin{bmatrix} \cos(15-20) & -\cos(20-35) & -\cos(15) & -\cos(20) & \cos(35) & 1 \\ \cos(30-40) & -\cos(40-60) & -\cos(30) & -\cos(40) & \cos(60) & 1 \\ \cos(45-60) & -\cos(60-90) & -\cos(45) & -\cos(60) & \cos(90) & 1 \\ \cos(60-90) & -\cos(90-110) & -\cos(60) & -\cos(90) & \cos(110) & 1 \\ \cos(75-110) & -\cos(110-145) & -\cos(75) & -\cos(110) & \cos(145) & 1 \\ \cos(90-140) & -\cos(140-180) & -\cos(90) & -\cos(140) & \cos(180) & 1 \end{bmatrix}$$

$b = [\cos(15-35) \quad \cos(30-60) \quad \cos(45-90) \quad \cos(60-110) \quad \cos(75-145) \quad \cos(90-180)]'$

**Numerical Analysis**

For Jacobi, Gauss-Seidel and SOR, an initial value of $x_0 = [2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2]'$ and a tolerance of 0.5e-5 will be employed. Also, the maximum number of iterations will be 100 for each method.

**Gaussian elimination method**

The gaussian elimination method requires the matrix of coefficients A and the vector of independent terms b to run. Both A and b can be viewed previously.

**With no pivoting (piv=0)**

The method gives as a result the following values for $K$:

$K_1$ = 0.161360158679686

$K_2$ =-1.238880641847576

$K_3$ =-1.122343299166891

$K_4$ =0.047696716851628

$K_5$ =-0.202129723776837

$K_6$ =-1.291425470747411

**With partial pivoting (piv=1)**

The method gives as a result the following values for $K$:

$K_1$ = 0.161360158679687

$K_2$ =-1.238880641847578

$K_3$ =-1.122343299166885

$K_4$ =0.047696716851626

$K_5$ =-0.202129723776837

$K_6$ =-1.291425470747411

**With total pivoting (piv=2)**

The method gives as a result the following values for $K$:

$K_1$ =0.161360158679686

$K_2$ =-1.238880641847578

$K_3$ =-1.122343299166887

$K_4$ =0.047696716851626

$K_5$ =-0.202129723776837

$K_6$ =-1.291425470747411

**Jacobi's method**

The Jacobi method requires both A and b, and, as mentioned above, must satisfy some requirements regarding the initial value vector, the tolerance, and the iterations. When entering this data into the code, we get that it does not converge.

```
los valores de error fueron:
     0
   NaN

La matriz T es:
   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN

El radio espectral de la matriz T es NaN
la tabla de iteraciones es:
    n    error n    vector solucion (con orden x1 x2 x3.....xn)
    _    _______    ___________________________________________

    0        0                   1×6 double
    1      NaN                   1×6 double
```

**Gauss-Seidel method**

The Gauss-Seidel method will also require the same inputs as the previous method. Based on this, we will obtain that the method does not converge.

```
los valores de error fueron:
     0
   NaN

La matriz T es:
   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN

El radio espectral de la matriz T es NaN
la tabla de iteraciones es:
    n    error n    vector solucion (con orden x1 x2 x3.....xn)
    _    _______    ___________________________________________

    0        0                   1×6 double
    1      NaN                   1×6 double
```

**SOR (w=0.8)**

The SOR method requires the same inputs as the previous methods, plus a value for the variable $w$ which must be contained in the interval (0, 2). In this case, $w$ will have a value of 0.8, giving that the method does not converge.

```
Fracasó en 100.000000 iteraciones, el vector:
   NaN
   NaN
   NaN
   NaN
   NaN
   NaN

la tabla de iteraciones es:
    n     error n     vector solucion (con orden x1 x2 x3.....xn)
    _     _______     ___________________________________________

    0        1                       1×6 double
    1       NaN                      1×6 double
```

## Chapter 3: Interpolation

### Stress-strain curve for A36 steel

Stress-strain curves are diagrams used to determine the relationship between the unitary strain of a material when subjected to specific loads that generate certain stresses in the solid. These graphs show the behavior of the material, both in its elastic and plastic zones, dividing the curve into four zones: elastic zone, yield zone, strain hardening zone, and necking zone (see Figure 3.1).

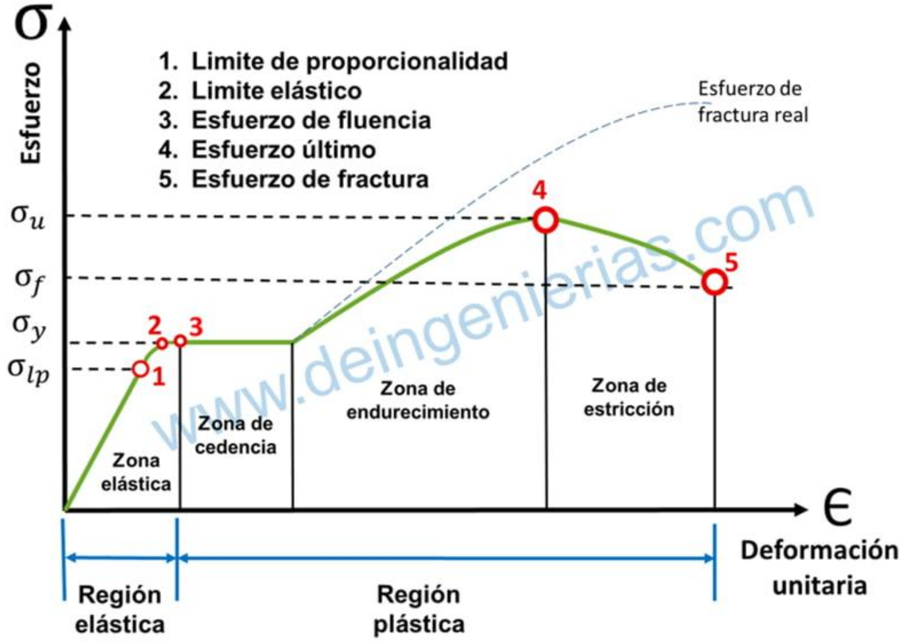



Figure 3.1 Stress-strain curve with critical points and zones shown. Note. De Ingenierías. (2019). Diagrama esfuerzo deformación [Diagram]. Retrieved from https://deingenierias.com/el-acero/diagrama-esfuerzo-deformacion/

Within these four zones we can find five critical points: the proportionality limit, elastic limit, yield strength, ultimate strength, and fracture strength. Among these, the most important are the last three, since they are the ones that indicate the resistance of the material and the limits that it reaches. Therefore, these three points will be taken into consideration to apply the different interpolation methods and generate the function that better describes the curve.

For this case, the stress-strain curve of ASTM A36 steel and the three critical points mentioned above will be analyzed. Taking as reference the following diagrams:

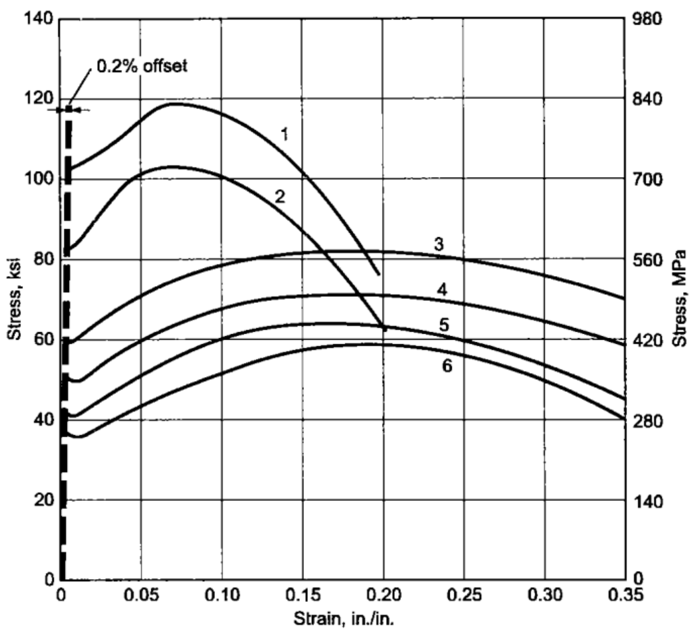



Figure 3.2. Stress-strain curve for ASTM A36 steel. Note. ASM INTERNATIONAL. (2002). Atlas of Stress-Strain Curves [Diagram]. Retrieved from shorturl.at/ftuO0

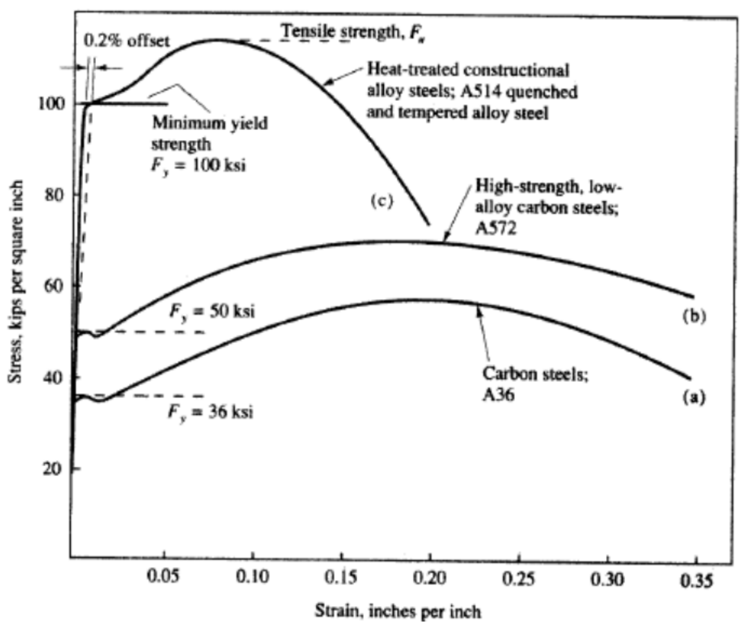



Figure 3.3. Stress-strain curve for ASTM A36 steel. Note. Dr. Hesham A. Numan. (n.d). Design of Steel Structures [Diagram]. Retrieved from shorturl.at/uQY16

And starting from the fact that the yield strength and ultimate strength of this steel is 250 Mpa and 400 Mpa respectively (Aceros Crea, 2022), a set of three points can be proposed as shown in the following table:

Table 3.1. Strain-stress values.

| Strain [in/in] | Stress [Mpa] |
|---|---|
| 0.023 | 250 |
| 0.2 | 400 |
| 0.35 | 280 |

The first two pairs of values represent the aforementioned critical points. The last pair represents the fracture strength of the material.

**Newton polynomial**

The Newton polynomial method calculates the n-1 degree Newton's divided differences polynomial. The method uses an initial value of [0.023, 0.2, 0.35] for $x$ and [250, 400, 280] for $y$.

The coefficients obtained for the polynomial are:

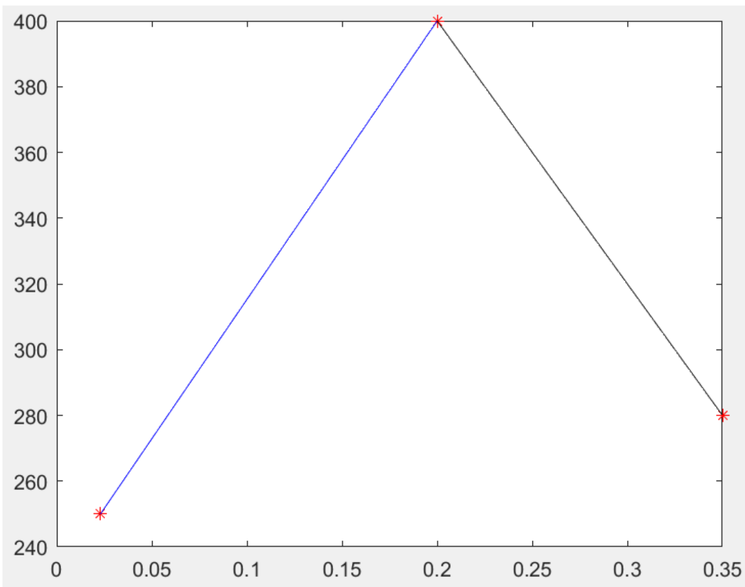
```
1.0e+03 *

-5.0381    1.9710    0.2073
```

**Linear Spline**

The spline method employs the same inputs as the Newton polynomial method but, in this case, a variable $d$ is introduced, which has a value of 1.

The coefficients obtained for the polynomials are:

| | |
|---|---|
| 847.4576 | 230.5085 |
| -800.0000 | 560.0000 |



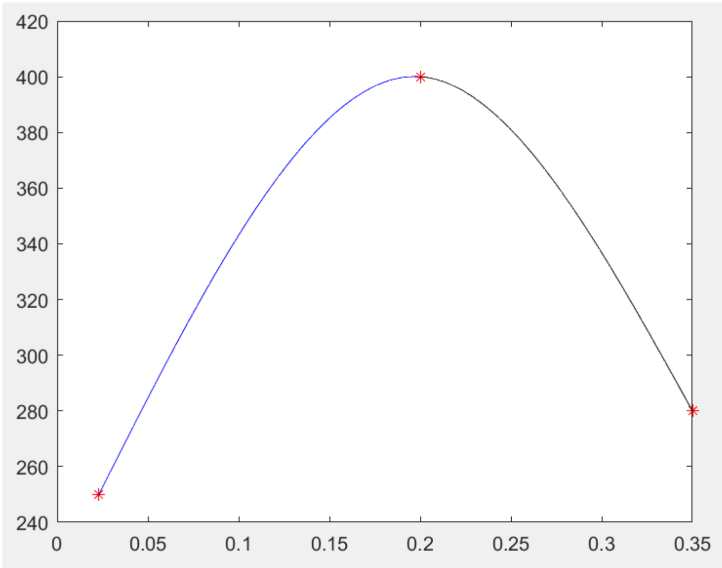

**Cubic Spline:**

The cubic spline uses the same parameters as the linear one, the only one that changes is the value of $d$, being in this case equal to 3.

The coefficients obtained for the polynomials are:

1.0e+04 *

| | | | |
|---|---|---|---|
| -1.4232 | 0.0982 | 0.1271 | 0.0220 |
| 1.6794 | -1.7633 | 0.4994 | -0.0028 |




## Chapter 4: Differential equations

### Mixtures problem

The mixing problem presents a situation where there is a constant flow of a combination of substances into and out of a container. To find the input to output ratio of the mixture, the following differential equation is used:

$$\frac{dQ}{dt} = [razón\ de\ entrada] - [razón\ de\ salida] \text{ (Equation 4.1)}$$

For this report, the following case will be analyzed: A tank initially contains 50 gallons of brine in which 10 lb of salt has dissolved. Brine, containing 2 lb of dissolved salt per gallon, enters the tank at a rate of 5 gal/min. The mixture is kept uniform, and comes out simultaneously at a rate of 3 gal/min. Based on formula 4.1, we can express the input to output ratio as follows:

$$\frac{dQ}{dt} = Q_i \times V_E - \frac{Q_s}{V+(V_E-V_S)t} V_S \text{ (Equation 4.2)}$$

Where $Q_i$ is the initial concentration, $V_E$ is the input rate, $Q_s$ is the output concentration (which we shall call x), $V$ is the initial volume of the mixture in the container, and $V_S$ is the output rate.

Taking the formula and the available data into account, we can set up our differential equation as follows:

$$\frac{dx}{dt} = (2 \times 5) - \frac{x}{50+(5-3)t} \times 3 \text{ (Equation 4.3)}$$

Thus, obtaining the following expression:

$$\frac{dx}{dt} + \frac{3}{50+2t}x = 10 \text{ (Equation 4.4)}$$

## Conclusions

### Chapter 1: Numerical solutions for nonlinear equations

Newton's method, regula falsi, and the secant method were the fastest in finding the answer with just 4 iterations, nevertheless, Newton was the most exact method with an error of just 0.000000000000006. On the other hand, the fixed-point method took the most iterations with 29 of them.

### Chapter 2: Numerical solutions for systems of equations

In the second chapter, neither Gauss-Seidel nor Jacobi and SOR converged due to the matrix A and the different requirements they need to converge. However, the gaussian elimination method did function, so we were able to obtain the values for $K$.

### Chapter 3: Interpolation

For the proposed problem, the method that best interpolates is the cubic spline because is the one that offers the most accurate values regarding the original stress-strain curve.

## References

De Ingenierías. (2019). Diagrama esfuerzo deformación. Retrieved from https://deingenierias.com/el-acero/diagrama-esfuerzo-deformacion/

ASM INTERNATIONAL. (2002). Atlas of Stress-Strain Curves. Retrieved from shorturl.at/ftuO0

Dr. Hesham A. Numan. (n.d). Design of Steel Structures. Retrieved from shorturl.at/uQY16