UNIVERSIDAD EAFIT
DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS
ELECCIÓN DEL PROYECTO
Primera Entrega

Asignatura: Análisis Numérico

Profesor responsable: Edwar Samir Posada Murillo

Semestre: 2020-2

Fecha de entrega del presente informe: 13-10-2020

Nombre del sistema (proyecto): Sitio Web Métodos Numéricos

Dirección Web del proyecto (o indique el repositorio desde donde trabajará):

https://github.com/eafit-201620030010/M-todos-Num-ricos

Integrantes:

José Alejandro Díaz Urrego

Jhon Jairo Chavarría Gaviria

Samuel Cadavid Pérez

Descripción del proyecto: El objetivo del proyecto es desarrollar una página web en la que se permita al usuario resolver problemas numéricos usando los métodos numéricos que se vean durante el semestre.

Para esta primera entrega se llevará a cabo un documento en el cual se evidencie cuáles fueron los códigos que se implementaron para los siguientes métodos correspondientes a la primera entrega del proyecto:

- Solución de ecuaciones de una variable
  - Bisección
  - Búsquedas incrementales
  - Regla falsa
  - Punto fijo
  - Newton
  - Secante
  - Raíces múltiples
- Solución de sistemas de ecuaciones lineales
  - Eliminación gaussiana
  - Pivoteo parcial
  - Pivoteo total

Método Bisección:

- Pseudocodigo:

```
Proceso Metodo de Biseccion

    Leer Xi, Xs, Tolerancia, Iter
    Yi = f(Xi)
    Ys = f(Xs)
    Si Yi = 0 Entonces
        Muestre: 'Xi es Raiz'
    Sino
        Si Ys = 0 Entonces
            Muestre: 'Xs es Raiz'
        Sino
            Si Yi*Ys < 0 Entonces
                Xm= (Xi + Xs)/2
                Contador = 1
                Ym = f(Xm)
                Error = Tolerancia + 1

                Mientras Error > Tolerancia & Ym ≠ 0 & Contador < Iter Hacer

                    Si Yi*Ym < 0 Entonces
                        Xs= Xm
                        Ys = Ym
                    Sino
                        Xi= Xm
                        Yi = Ym
                    Fin Si

                    Xaux = Xm
                    Xm = (Xi + Xs)/2
                    Ym = f(Xm)
                    Error = Abs(Xm - Xaux)
                    Contador = Contador + 1

                Fin Mientras

                Si Ym = 0 Entonces
                    Muestre: 'Xm es Raiz'
                Sino Si Error < Tolerancia Entonces
                    Muestre: 'Xm es aproximacion a una raiz con una toleracia 'Tolerancia''
                Sino
                    Muestre: 'Fracaso en 'Iter' iterraciones'
                Fin Si
                Fin Si
            Sino
                Muestre: 'El intervalo es inadecuado'

            Fin Si
        Fin Si
    Fin Si
Fin Proceso
```

- Como ejecutar:

```
biseccion(0,1,0.0000001,100,"log(sin(x)**2 + 1) - 1/2")
```

- Resultado:

```
#+RESULTS:
:
: [[[1, 0, 1, 0.5, -0.2931087267313766, 0], [2, 0.5, 1, 0.75, -
0.11839639385347844, 0.25], [3, 0.75, 1, 0.875, -0.036817690757380395,
0.125], [4, 0.875, 1, 0.9375, 0.0006339161592386899, 0.0625], [5, 0.875,
0.9375, 0.90625, -0.017772289226861138, 0.03125], [6, 0.90625, 0.9375,
0.921875, -0.008486582211768012, 0.015625], [7, 0.921875, 0.9375,
0.9296875, -0.0039053586270640928, 0.0078125], [8, 0.9296875, 0.9375,
0.93359375, -0.0016304381170096915, 0.00390625], [9, 0.93359375, 0.9375,
0.935546875, -0.0004969353153195244, 0.001953125], [10, 0.935546875,
0.9375, 0.9365234375, 6.882244496264622e-05, 0.0009765625], [11,
0.935546875, 0.9365234375, 0.93603515625, -0.00021397350516394464,
0.00048828125], [12, 0.93603515625, 0.9365234375, 0.936279296875, -
7.255478812057126e-05, 0.000244140625], [13, 0.936279296875,
0.9365234375, 0.9364013671875, -1.8609849000705836e-06,
0.0001220703125], [14, 0.9364013671875, 0.9365234375, 0.93646240234375,
3.348202684883006e-05, 6.103515625e-05], [15, 0.9364013671875,
0.93646240234375, 0.936431884765625, 1.5810845160335596e-05,
3.0517578125e-05], [16, 0.9364013671875, 0.936431884765625,
0.9364166259765625, 6.975011174192858e-06, 1.52587890625e-05], [17,
0.9364013671875, 0.9364166259765625, 0.9364089965820312,
2.5570333977986692e-06, 7.62939453125e-06], [18, 0.9364013671875,
0.9364089965820312, 0.9364051818847656, 3.4802931392352576e-07,
3.814697265625e-06], [19, 0.9364013671875, 0.9364051818847656,
0.9364032745361328, -7.56476526753147e-07, 1.9073486328125e-06], [20,
0.9364032745361328, 0.9364051818847656, 0.9364042282104492, -
2.042232898902263e-07, 9.5367431640625e-07], [21, 0.9364042282104492,
0.9364051818847656, 0.9364047050476074, 7.190309125881811e-08,
4.76837158203125e-07], [22, 0.9364042282104492, 0.9364047050476074,
0.9364044666290283, -6.616007947046754e-08, 2.384185791015625e-07], [23,
0.9364044666290283, 0.9364047050476074, 0.9364045858383179,
2.8715108069121698e-09, 1.1920928955078125e-07], [24,
0.9364044666290283, 0.9364045858383179, 0.9364045262336731, -
3.164428308277678e-08, 5.960464477539063e-08]], ['0.9364045262336731 es
una aproximación a una raíz con una tolerancia = 1e-07', True]]
:
: ['0.9364045262336731 es una aproximación a una raíz con una tolerancia
= 1e-07', True]
```

Método Búsquedas incrementales

- Pseudocodigo:

```
Proceso Busquedas incrementadas

    Leer Xo,delta, Iter
    Yo = f(Xo)
    Si Yo = 0 Entonces
        Muestre: 'Xo es Raiz'
    Sino
        X1 = Xo + delta
        Contador = 1
        Y1 = f(X1)

        Mientras Yo*Y1 > 0  &  Contador < Iter Hacer

            Xo = X1
            Yo = Y1
            X1 = Xo + delta
            Y1 = f(X1)
            Contador = Contador + 1

        Fin Mientras

        Si Y1 = 0 Entonces
            Muestre: 'Xo es Raiz'
        Sino Si Yo*Y1 < 0 Entonces
                Muestre: 'Hay una Raiz entre Xo y X1'
            Sino
                Muestre: 'Fracaso en 'Iter' iterraciones'
            Fin Si

        Fin Si

    Fin Si

Fin Proceso
```

- Como ejecutar:

```
busquedasIncrementales(-3, 0.5, 100, "log(sin(x)**2 + 1) - 1/2")
```

- Resultado:

```
        #+RESULTS:
['Hay una raiz entre -2.5 y -2.0']
['Hay una raiz entre -1.0 y -0.5']
['Hay una raiz entre 0.5 y 1.0']
['Hay una raiz entre 2.0 y 2.5']
['Hay una raiz entre 4.0 y 4.5']
['Hay una raiz entre 5.0 y 5.5']
['Hay una raiz entre 7.0 y 7.5']
['Hay una raiz entre 8.0 y 8.5']
['Hay una raiz entre 10.0 y 10.5']
['Hay una raiz entre 11.5 y 12.0']
['Hay una raiz entre 13.5 y 14.0']
['Hay una raiz entre 14.5 y 15.0']
['Hay una raiz entre 16.5 y 17.0']
['Hay una raiz entre 17.5 y 18.0']
['Hay una raiz entre 19.5 y 20.0']
['Hay una raiz entre 21.0 y 21.5']
['Hay una raiz entre 22.5 y 23.0']
['Hay una raiz entre 24.0 y 24.5']
['Hay una raiz entre 26.0 y 26.5']
['Hay una raiz entre 27.0 y 27.5']
['Hay una raiz entre 29.0 y 29.5']
['Hay una raiz entre 30.0 y 30.5']
['Hay una raiz entre 32.0 y 32.5']
['Hay una raiz entre 33.5 y 34.0']
['Hay una raiz entre 35.0 y 35.5']
['Hay una raiz entre 36.5 y 37.0']
['Hay una raiz entre 38.5 y 39.0']
['Hay una raiz entre 39.5 y 40.0']
['Hay una raiz entre 41.5 y 42.0']
['Hay una raiz entre 43.0 y 43.5']
['Hay una raiz entre 44.5 y 45.0']
['Hay una raiz entre 46.0 y 46.5']
```

Método Regla Falsa

- Pseudocodigo:

```
Proceso Metodo de la regla falsa
    Leer Xi, Xs, Tolerancia, Iter
    Yi = f(Xi)
    Ys = f(Xs)
    Si Yi = 0 Entonces
        Muestre: 'Xi es Raiz'
    Sino
        Si Ys = 0 Entonces
            Muestre: 'Xs es Raiz'
        Sino
            Si Yi*Ys < 0 Entonces
                Xm= Xi – ((Yi*(Xs – Xi))/(Ys – Yi))
                Contador = 1
                Ym = f(Xm)
                Error = Tolerancia + 1

                Mientras Error > Tolerancia & Ym ≠ 0 & Contador < Iter Hacer

                    Si Yi*Ym < 0 Entonces
                        Xs= Xm
                        Ys = Ym
                    Sino
                        Xi= Xm
                        Yi = Ym
                    Fin Si

                    Xaux = Xm
                    Xm= Xi – ((Yi*(Xs – Xi))/(Ys – Yi))
                    Ym = f(Xm)
                    Error = Abs(Xm - Xaux)
                    Contador = Contador + 1

                Fin Mientras

                Si Ym = 0 Entonces
                    Muestre: 'Xm es Raiz'
                Sino Si Error < Tolerancia Entonces
                    Muestre: 'Xm es aproximacion a una raiz con una toleracia 'Tolerancia''
                Sino
                    Muestre: 'Fracaso en 'Iter' iterraciones'
                Fin Si
                Fin Si
            Sino
                Muestre: 'El intervalo es inadecuado'

            Fin Si
        Fin Si
    Fin Si
Fin Proceso
```

- Como ejecutar:

```
reglaFalsa(0,1,0.0000001,100,"log(sin(x)**2 + 1) - 1/2")
```

- Resultado:

```
#+RESULTS:


:


: [[[1, 0, 1, 0.9339403807182157, -0.0014290767036854723, 0], [2,
0.9339403807182157, 1, 0.9365060516656253, 5.8756008358140654e-05,
0.0025656709474095596], [3, 0.9339403807182157, 0.9365060516656253,
0.9364047307426412, 8.678254082017389e-08, 0.0001013209229840939], [4,
0.9339403807182157, 0.9364047307426412, 0.9364045811008692,
1.281542649778089e-10, 1.4964177197374084e-07], [5, 0.9339403807182157,
0.9364045811008692, 0.936404580879889, 1.8907098109366416e-13,
2.2098023411132317e-10]], ['0.936404580879889 es una aproximación a una
raíz con una tolerancia = 1e-07', True]]


: ['0.936404580879889 es una aproximación a una raíz con una tolerancia =
1e-07', True]
```

Método punto fijo

- Pseudocodigo:

```
Proceso Metodo punto Fijo

    Leer Xo, Tolerancia, Iter
    Yo = f(Xo)
    Contador = 0
    Error = Tolerancia + 1

    Mientras Yo ≠ 0 & Error > Tolerancia & Contador < Iter Hacer

        Xn = g(Xo)
        Yo = f(Xn)
        Error = abs ((Xn - Xo)/Xn)
        Xo = Xn
        Contador = Contador + 1

    Fin Mientras

        Si Yo = 0 Entonces
            Muestre: 'Xo es Raiz'
        Sino Si Error < Tolerancia Entonces
            Muestre: ''Xo' es una raiz aproximada con una tolerancia 'Tolerancia''
          Sino
            Muestre: 'Fracaso en 'Iter' iterraciones'
          Fin Si

        Fin Si


Fin Proceso
```

- Como ejecutar:

```
puntoFijo(-0.5,0.0000001,100,"log(sin(x)**2 + 1) - 1/2","log(sin(x)**2 + 1)
- 1/2")
```

- Resultado:

```
#+RESULTS:

:

: [[[0, -0.5, -0.2931087267313766, 0], [1, -0.2931087267313766, -
0.41982154360625734, 0.2068912732686234], [2, -0.41982154360625734, -
0.3463045191776651, 0.12671281687488073], [3, -0.3463045191776651, -
0.3909584565423095, 0.07351702442859226], [4, -0.3909584565423095, -
0.3644050348941392, 0.0446539373646444], [5, -0.3644050348941392, -
0.3804263031679563, 0.02655342164817026], [6, -0.3804263031679563, -
0.37083679528020885, 0.016021268273817058], [7, -0.37083679528020885, -
0.3766056453635812, 0.009589507887747428], [8, -0.3766056453635812, -
0.373145417607189, 0.005768850083372357], [9, -0.373145417607189, -
0.3752246411870562, 0.003460227756392209], [10, -0.3752246411870562, -
0.37397658604830963, 0.002079223579867173], [11, -0.37397658604830963, -
0.3747262157084321, 0.0012480551387465 4], [12, -0.3747262157084321, -
0.37427613331045395, 0.0007496296601224861], [13, -0.37427613331045395, -
0.3745464284580923, 0.0004500823979781 6874], [14, -0.3745464284580923, -
0.3743841264348447, 0.0002702951476383 2196], [15, -0.3743841264348447, -
0.3744815908319551, 0.0001623020232475736], [16, -0.3744815908319551, -
0.37442306518389706, 9.746439711039168e-05], [17, -0.37442306518389706, -
0.37445820986270584, 5.852564805802 7624e-05], [18, -0.37445820986270584, -
0.3744371058494556, 3.514467880877392e-05], [19, -0.3744371058494556, -
0.37444977872741303, 2.110401325022826e-05], [20, -0.37444977872741303, -
0.37444216876320036, 1.267287795742 0337e-05], [21, -0.37444216876320036, -
0.3744467385052047, 7.609964212673681e-06], [22, -0.3744467385052047, -
0.37444399440652526, 4.569742004356 6694e-06], [23, -0.37444399440652526, -
0.37444564222126353, 2.744098679452467e-06], [24, -0.37444564222126353, -
0.37444465271927385, 1.647814738270359e-06], [25, -0.37444465271927385, -
0.3744452469090602, 9.895019896788426e-07], [26, -0.3744452469090602, -
0.37444489010190096, 5.941897863737111e-07], [27, -0.37444489010190096, -
0.37444510436235334, 3.568071592630062e-07], [28, -0.37444510436235334, -
0.3744449757003151, 2.142604523802660 3e-07], [29, -0.3744449757003151, -
0.37444505296105535, 1.28662038245686e-07], [30, -0.37444505296105535, -
0.3744450065664714, 7.726074024994034e-08]], ['-0.37444505296105535 es una
aproximación con tolerancia 1e-07', True]]

:

: ['-0.37444505296105535 es una aproximación con tolerancia 1e-07', True]
```

Método de newton

- Pseudocodigo:

```
Proceso Metodo de Newton

    Leer Xo, Tolerancia, Iter
    Yo = f(Xo)
    Do = f '(Xo)
    Contador = 0
    Error = Tolerancia + 1

    Mientras Yo ≠ 0 & Do ≠ 0 & Error > Tolerancia & Contador < Iter Hacer

        X1 = Xo − (Yo/Do)
        Yo = f(X1)
        Do = f '(X1)
        Error = abs ((X1 - Xo)/X1)
        Xo = X1
        Contador = Contador + 1

    Fin Mientras

        Si Yo = 0 Entonces
            Muestre: 'Xo es Raiz'
        Sino Si Error < Tolerancia Entonces
            Muestre: ''Xo' es una raiz aproximada con una tolerancia 'Tolerancia''
          Sino Si Do = 0 Entonces
                Muestre: ''Xo' es posiblemente una raiz múltiple'
              Sino
                Muestre: 'Fracaso en 'Iter' iterraciones'
              Fin Si
          Fin Si

        Fin Si
Fin Proceso
```

- Como ejecutar:

```
newton(0.5,0.0000001,100,"log(sin(x)**2 + 1) -
1/2","(2*sin(x)*cos(x))/(sin(x)**2 + 1)")
```

- Resultado:

```
#+RESULTS:

:

: [[[0, 0.5, -0.2931087267313766, 0.6842068330717285, 0], [1,
0.9283919899125719, -0.004662157097372055, 0.5846147284064962,
0.4283919899125719], [2, 0.9363667412673313, -2.1912619882713535e-05,
0.5791052537949999, 0.007974751354759446], [3, 0.9364045800189902, -
4.98339092214195e-10, 0.5790789133390185, 3.783875165885853e-05], [4,
0.9364045808795621, -1.1102230246251565e-16, 0.5790789127399327,
8.605719470367035e-10]], ['0.9364045808795621 es una aproximación con
tolerancia 1e-07', True]]

:

: ['0.9364045808795621 es una aproximación con tolerancia 1e-07', True]
```

Método de secante

- Pseudocodigo:

```
Proceso Metodo de la Secante

    Leer X1, Xo, Tolerancia, Iter
    Yo = f(Xo)
    Si Yo = 0 Entonces
        Muestre: 'Xo es Raiz'
    Sino
        Y1 = f(X1)
        Contador = 0
        Error = Tolerancia + 1
        Den = Y1 − Yo

        Mientras Error > Tolerancia & Y1 ≠ 0 & Den ≠ 0 & Contador < Iter  Hacer

            X2 = X1 - ((Y1*(X1−Xo))/Den)
            Error = Abs((X2 - X1)/ X2)
            Xo = X1
            Yo = Y1
            X1 = X2
            Y1 = f(X1)
            Den = Y1 − Yo
            Contador = Contador + 1

        Fin Mientras

        Si Y1 = 0 Entonces
            Muestre: 'X1 es Raiz'
        Sino Si Error < Tolerancia  Entonces
            Muestre: ''X1' es una raiz aproximada con una tolerancia 'Tolerancia'"
            Sino Si Den = 0  Entonces
                Muestre: 'Hay posiblemente una raiz múltiple'
            Sino
                Muestre: 'Fracaso en 'Iter' iterraciones'
            Fin Si
        Fin Si

        Fin Si
    Fin Si
Fin Proceso
```

- Como ejecutar:

```
secante(0.5,1,0.0000001,100,"log(sin(x)**2 + 1) - 1/2")
```

- Resultado:

```
#+RESULTS:

:

: [[[0, 0.5, -0.2931087267313766, 0], [1, 1, 0.03536607938024017, 0], [2,
0.946166222306525, 0.005619392737863826, 0.05383377769347497], [3,
0.9359965807911726, -0.00023632217470054284, 0.010169641515352379], [4,
0.9364070023767038, 1.4022358909571153e-06, 0.00041042158553117325], [5,
0.9364045814731196, 3.4371649970665885e-10, 2.420903584265943e-06], [6,
0.9364045808795615, -4.996003610813204e-16, 5.935580915661376e-10]],
['0.9364045808795615 es una aproximación a una raíz con tolerancia=1e-07',
True]]

:

: ['0.9364045808795615 es una aproximación a una raíz con tolerancia=1e-
07', True]
```

Método de las raíces múltiples

- Pseudocodigo:

```
Proceso Metodo de las Raices Múltiples

    Leer Xo, Tolerancia, Iter
    Yo = f(Xo)
    Do = f ´(Xo)
    D2o = f ´´(Xo)
    Deno= Do² − (Yo * D2o)
    Contador = 0
    Error = Tolerancia + 1

    Mientras Yo ≠ 0 & Deno ≠ 0 & Error > Tolerancia & Contador < Iter Hacer

        X1 = Xo − ((Yo*Do) / Deno)
        Yo = f(X1)
        Do = f ´(X1)
        D2o = f ´´(X1)
        Error = abs ((X1 - Xo)/X1)
        Deno= Do² − (Yo * D2o)
        Xo = X1
        Contador = Contador + 1

    Fin Mientras

        Si Yo = 0 Entonces
            Muestre: 'Xo es Raiz'
        Sino Si Error < Tolerancia Entonces
            Muestre: ''Xo' es una raiz aproximada con una tolerancia 'Tolerancia''
          Sino Si Deno = 0 Entonces
                Muestre: 'El denominador se hace cero'
              Sino
                Muestre: 'Fracaso en 'Iter' iterraciones'
              Fin Si
          Fin Si

        Fin Si
Fin Proceso
```

- Como ejecutar:

```
raicesMultiples(1,0.0000001,100,"exp(x) - x - 1","exp(x) - x","exp(x)")
```

- Resultado:

```
#+RESULTS:

: [[[0, 1, 0.7182818284590451, 1.718281828459045, 2.718281828459045, 0], [1,
-0.23421061355351425, 0.025405775475345838, -0.20880483807816852,
0.7911951619218315, 1.2342106135535142], [2, -0.00845827991076109,
3.567060801401567e-05, -0.008422609302746964, 0.991577390697253,
0.22575233364275316], [3, -1.1890183808588653e-05, 7.068789997788372e-11, -
1.1890113120638368e-05, 0.9999881098868794, 0.008446389726952502], [4, -
4.218590698935789e-11, 0.0, -4.218592142279931e-11, 0.9999999999578141,
1.1890141622681664e-05]], ['-4.218590698935789e-11 es una raíz.', True]]


:


: ['-4.218590698935789e-11 es una raíz.', True]
```

Eliminación gaussiana simple

- Pseudocodigo:

```
Proceso Eliminacion Gaussiana Simple
    Leer A, b
    (n,m)= Tamaño(A)
    a = FormaMatrizAumentada (A,b)

    Si n = m Entonces
        Para k=1 Hasta n-1
            Para i=k+1 Hasta n
                m_ik = a_ik / a_kk
                Para j=k Hasta n+1
                    a_ij = a_ij - m_ik a_kj
                Fin Para
            Fin Para
        Fin Para

        Para i=n Hasta 1
            Suma = 0
            Para p=i+1 Hasta n
                Suma = Suma + a_ip x_p
            Fin Para
            x_i = ( b_i – Suma ) / a_ii
        Fin Para
    Sino
        Muestre: 'La matriz no es Cuadrada'
    Fin Si
    Imprimir a
    Imprimir x
Fin Proceso
```

- Como ejecutar:

```
gaussianaSimple(Ab, len(Ab), True)
```

- Resultado:

```
      #+RESULTS:
#+begin_example
Matriz A:
[2, -1, 0, 3]
[1, 0.5, 3, 8]
[0, 13, -2, 11]
[14, 5, -2, 3]
Vector b:
1
1
1
1
Matriz Ab:
[2, -1, 0, 3, 1]
[1, 0.5, 3, 8, 1]
[0, 13, -2, 11, 1]
[14, 5, -2, 3, 1]
```

```
Después de aplicar sustitución Gaussiana simple:

Etapa 0 :

[2, -1, 0, 3, 1]

[1, 0.5, 3, 8, 1]

[0, 13, -2, 11, 1]

[14, 5, -2, 3, 1]



Etapa 1 :

[2, -1, 0, 3, 1]

[0.0, 1.0, 3.0, 6.5, 0.5]

[0.0, 13.0, -2.0, 11.0, 1.0]

[0.0, 12.0, -2.0, -18.0, -6.0]



Etapa 2 :

[2, -1, 0, 3, 1]

[0.0, 1.0, 3.0, 6.5, 0.5]

[0.0, 0.0, -41.0, -73.5, -5.5]

[0.0, 0.0, -38.0, -96.0, -12.0]



Etapa 3 :

[2, -1, 0, 3, 1]

[0.0, 1.0, 3.0, 6.5, 0.5]

[0.0, 0.0, -41.0, -73.5, -5.5]
```

```
[0.0, 0.0, 0.0, -27.878048780487802, -6.902439024390244]



resultados x:

[0.03849518810148722, -0.18022747156605434, -0.30971128608923887,
0.24759405074365706]

#+end_example
```

Eliminación gaussiana con pivoteo parcial

- Pseudocodigo:

```
Proceso Pivoteo Parcial
    Leer A, b
    (n,m)= Tamaño(A)
    a = FormaMatrizAumentada (A,b)
    Si n = m Entonces
        Para k=1 Hasta n-1
            mayor = 0
            filam = k
            Para p=k Hasta n
                Si mayor < |a_pk| Entonces
                    mayor = |a_pk|
                    filam = p
                Fin si
            Fin Para
            Si mayor = 0 Entonces
                Muestre: 'Suspendido el proceso, Infinitas soluciones'
            Sino
                Si filam ≠ k Entonces
                    Para j=1 Hasta n+1
                        Aux = a(k, j)
                        a(k, j) = a(filam, j)
                        a(filam ,j) = Aux
                    Fin para
                Fin Si
            Fin Si
            Para i=k+1 Hasta n
                m_ik = a_ik / a_kk
                Para j=k Hasta n+1
                    a_ij = a_ij - m_ik a_kj
                Fin Para
            Fin Para
        Fin Para
        Para i=n Hasta 1
            Suma = 0
            Para p=i+1 Hasta n
                Suma = Suma + a_ip x_p
            Fin Para
            x_i = ( b_i – Suma ) / a_ii
        Fin Para
    Sino
        Muestre: 'La matriz no es Cuadrada'
    Fin Si
    Imprimir a
    Imprimir x
Fin Proceso
```

- Como ejecutar:

```
gaussianaPivoteoParcial(Ab, len(Ab), True)
```

- Resultado:

```
  #+RESULTS:

#+begin_example

Matriz A:

[2, -1, 0, 3]

[1, 0.5, 3, 8]

[0, 13, -2, 11]

[14, 5, -2, 3]

Vector b:

1

1

1

1

Matriz Ab:

[2, -1, 0, 3, 1]

[1, 0.5, 3, 8, 1]

[0, 13, -2, 11, 1]

[14, 5, -2, 3, 1]

Después de aplicar sustitución Gaussiana con pivoteo parcial:
```

```
Etapa 0 :

[14, 5, -2, 3, 1]

[1, 0.5, 3, 8, 1]

[0, 13, -2, 11, 1]

[2, -1, 0, 3, 1]



Etapa 1 :

[14, 5, -2, 3, 1]

[0.0, 13.0, -2.0, 11.0, 1.0]

[0.0, 0.1428571428571429, 3.142857142857143, 7.785714285714286,
0.9285714285714286]

[0.0, -1.7142857142857142, 0.2857142857142857, 2.5714285714285716,
0.8571428571428572]



Etapa 2 :

[14, 5, -2, 3, 1]

[0.0, 13.0, -2.0, 11.0, 1.0]

[0.0, 0.0, 3.1648351648351647, 7.664835164835164, 0.9175824175824177]

[0.0, 2.220446049250313e-16, 0.021978021978021955, 4.021978021978022,
0.989010989010989]



Etapa 3 :

[14, 5, -2, 3, 1]

[0.0, 13.0, -2.0, 11.0, 1.0]

[0.0, 0.0, 3.1648351648351647, 7.664835164835164, 0.9175824175824177]
```

```
[0.0, 2.220446049250313e-16, 0.0, 3.96875, 0.982638888888889]



0.0384951881014873

-0.18022747156605426

-0.30971128608923887

0.24759405074365706

#+end_example
```

Eliminación gaussiana con pivoteo total

- Pseudocodigo:

```
Proceso Pivoteo Total
    Leer A, b
    (n,m)= Tamaño(A)
    a = FormaMatrizAumentada (A,b)
    Si n = m Entonces
        Para i=1 Hasta n
            marca(i)= i
        Fin Para
        Para k=1 Hasta n-1
            mayor = 0
            filam = k
            columnam = k
            Para p=k Hasta n
                Para r=k Hasta n
                    Si mayor < |a_pr| Entonces
                        mayor = |a_pr|
                        filam = p
                        columnam = r
                    Fin si
                Fin Para
            Fin Para
            Si mayor = 0 Entonces
                Muestre: 'Suspendido el proceso, Infinitas soluciones'
            Sino
                Si filam ≠ k Entonces
                    Para j=1 Hasta n+1
                        Aux = a(k, j)
                        a(k, j) = a(filam, j)
                        a(filam ,j) = Aux
                    Fin para
                Fin Si
                Si columnam ≠k Entonces
                    Para i=1 Hasta n
                        Aux = a(i, k)
                        a(i, k) = a(i, columnam)
                        a(i, columnam) = Aux
                    Fin para
                    Aux = marca (k)
                    marca (k) = marca (columnam)
                    marca (columnam) = Aux
                Fin Si
            Fin Si
            Para i=k+1 Hasta n
                m_ik = a_ik / a_kk
                Para j=k Hasta n+1
                    a_ij = a_ij - m_ik a_kj
                Fin Para
            Fin Para
        Fin Para
```

```
        Para  i=n  Hasta  1
                Suma = 0
                Para  p=i+1 Hasta  n
                        Suma = Suma + a_ip x_p
                Fin Para
                x_i = ( b_i – Suma ) / a_ii
        Fin Para
        Para i=n Hasta 1
                Para  j=1 Hasta n
                        Si  marca(j) = i Entonces
                                k = j
                        Fin Si
                Fin Para
                Aux = x(k)
                x(k) = x(i)
                x(i) = Aux
                Aux = marca(k)
                marca(k) = marca(i)
                marca(i) = Aux
        Fin Para
    Sino
        Muestre: 'La matriz no es Cuadrada'
    Fin Si
     Imprimir a
     Imprimir x
Fin Proceso
```

- Como ejecutar:

```
gaussianaPivoteoTotal(Ab, len(Ab), True)
```

- Resultado:

```
#+RESULTS:

#+begin_example

Matriz A:

[2, -1, 0, 3]

[1, 0.5, 3, 8]

[0, 13, -2, 11]

[14, 5, -2, 3]
```

```
Vector b:

1

1

1

1

Matriz Ab:

[2, -1, 0, 3, 1]

[1, 0.5, 3, 8, 1]

[0, 13, -2, 11, 1]

[14, 5, -2, 3, 1]

**********

Después de aplicar sustitución Gaussiana con pivoteo total:

Etapa 0 :

[14, 5, -2, 3, 1]

[1, 0.5, 3, 8, 1]

[0, 13, -2, 11, 1]

[2, -1, 0, 3, 1]

Etapa 1 :

[14, 5, -2, 3, 1]

[0.0, 13.0, -2.0, 11.0, 1.0]

[0.0, 0.1428571428571429, 3.142857142857143, 7.785714285714286,
0.9285714285714286]

[0.0, -1.7142857142857142, 0.2857142857142857, 2.5714285714285716,
0.8571428571428572]
```

```
Etapa 2 :

[14, 5, 3, -2, 1]

[0.0, 13.0, 11.0, -2.0, 1.0]

[0.0, 0.0, 7.664835164835164, 3.1648351648351647, 0.9175824175824177]

[0.0, 2.220446049250313e-16, 4.021978021978022, 0.021978021978021955,
0.989010989010989]

Etapa 3 :

[14, 5, 3, -2, 1]

[0.0, 13.0, 11.0, -2.0, 1.0]

[0.0, 0.0, 7.664835164835164, 3.1648351648351647, 0.9175824175824177]

[0.0, 2.220446049250313e-16, 0.0, -1.638709677419355, 0.5075268817204301]

Resultados x:

0.038495188101487325

-0.18022747156605423

0.24759405074365703

-0.3097112860892388

#+end_example
```