# Comparación de los tiempos de ejecución de algoritmos secuenciales iterativos para la resolución de sistemas de ecuaciones lineales programados en Julia y Python

Max B. Austria [1]

Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas (IIMAS), UNAM

[1] austriamax@ciencias.unam.mx

# Resumen

El presente documento expone los resultados de los tiempos de ejecución de algoritmos secuenciales para resolver sistemas de ecuaciones lineales iterativamente mediante Python y Julia: Gradiente Conjugado (GC), Gauss-Seidel(GS), Jacobi (Jac) y Sobre Relajación Sucesiva (SRS).

**Palabras clave:** Sistemas de Ecuaciones Lineales, Métodos Númericos, Ejecución Secuencial, Python, Julia, Tiempo de ejecución.

# Introducción

Los sistemas de ecuaciones lineales son de la forma

$$Ax = b.$$

Donde

- $x$ es un vector de dimensión $n$ (el número de incógnitas).
- $A$ es una matriz cuadrada de $n \times n$, con los coeficientes del sistema.
- $b$ un vector con los términos independientes.

Por lo común, los métodos de resolución de dichos sistemas suelen clasificarse como: Métodos directos y Métodos iterativos.

# Métodos iterativos

Los *métodos iterativos* están basados en construir una secuencia de soluciones aproximadas $\{x^{(k)}\}_{k=1,2,\ldots}$, que a medida que el contador de iteraciones $k$ aumente; se aproxime a la solución exacta.

Esta secuencia se construye usando *fórmulas de recurrencia*. Hay varias fórmulas de recurrencia y es posible obtenerlas de diferentes manereas. Por ejemplo:

$$x^{(k)} = Gx^{(k-1)+c}$$

# Métodos iterativos

Hay muchos métodos iterativos para resolver sistemas de ecuaciones. Algunos de los más clasicos son:

- Método de Jacobi.
- Método de Gauss-Seidel.
- Método de Sobre Relajación.

Por otro lado hay otros bassados en técnicas de optimización, como los métodos de iteración por subespacios, entre el cual se encuentra:

- Método del gradiente conjugado.

# Características

Algunas características a considerar cuando se quiere resolver un sistema de ecuaciones lineales son las siguientes:

- Los métodos directos, resultan caros de resolver para grandes sistemas de ecuaciones.
- Los métodos iterativos que se nombraron (Jacobi, Gauss-Seidel,SRS) son sencillos de implementar. Sin embargo su convergencia se considera lenta.
- El método del gradiente conjugado puede ser mirado como un método directo, ya que en un número finito de pasos conduce a la solución exacta, en la practica se utiliza como un método iterativo.

# Experimento

Para cumplir con los objetivos se diseño el siguiente experimento:

- Se implemento un sistema de ecuaciones $Ax = b$ de tamaño variable.
  - Se tomaron los tamaños $n = 20, 40, 50, 80, 100$.
  - La construcción de la matriz $A$ y el vector $b$ se encuentra en el anexo C.
  - La solución del sistema de ecuaciones siempre es $x = (1, 1, \ldots, 1)'_n$.
- Se implementaron los algoritmos mencionados en Julia y Python.
  - La implementación en Julia se encuentra en el anexo A y contiene ejemplos de su funcionamiento.
  - La implementación en Python se encuentra en el anexo B y también contiene ejemplos.

# Experimento

- La implementación del experimento se puede encontrar en el anexo D.
  - Todas las ejecuciones toman al origen como la inicialización de la solución ($x_0$).
  - Todas las ejecuciones tienen por objetivo alcanzar una precisión de $e = 0.000001$.
  - Todas las ejecuciones tienen un limite de iteraciones igual a 100 (itermax).
  - La implementación se realiza en Julia y realiza llamadas a Python a través del paquete PyCall.
  - Por cada cambio de $N = 20, 40, 50, 80, 100$ y cada algoritmo se realizaron 300 repeticiones.
  - El experimento finaliza cuando se escribe un archivo CSV con los datos de las simulaciones.
- El experimento se realizo en una computadora: Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz con 4 procesadores.

# Experimento

Al final de los anexos se encuentra el tabulado de los valores medios por algoritmo y parámetros.

# Resultados

- En general se observa una tendencia creciente entre el tiempo de ejecución y el tamaño de la entrada (T).

# Resultados

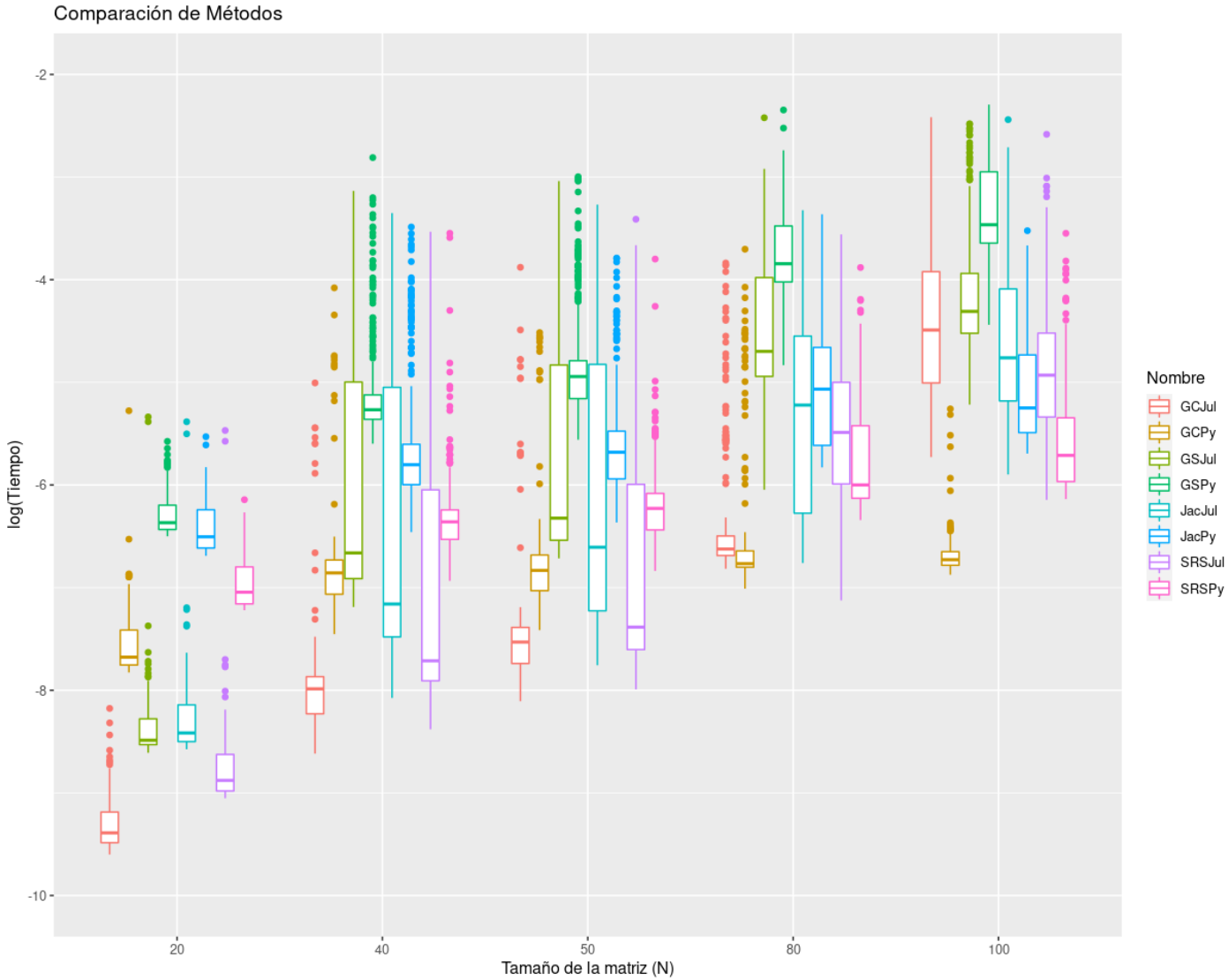



Figure: Tendencia creciente entre el tiempo y el tamaño de la entrada.

# Resultados

- Para sistemas con menos de 50 ecuaciones, Julia Domina a Python en practicamente todos los algoritmos con cualquier parámetro.
- Para 80 y 100 ecuaciones es necesario ir caso por caso.

# Resultados

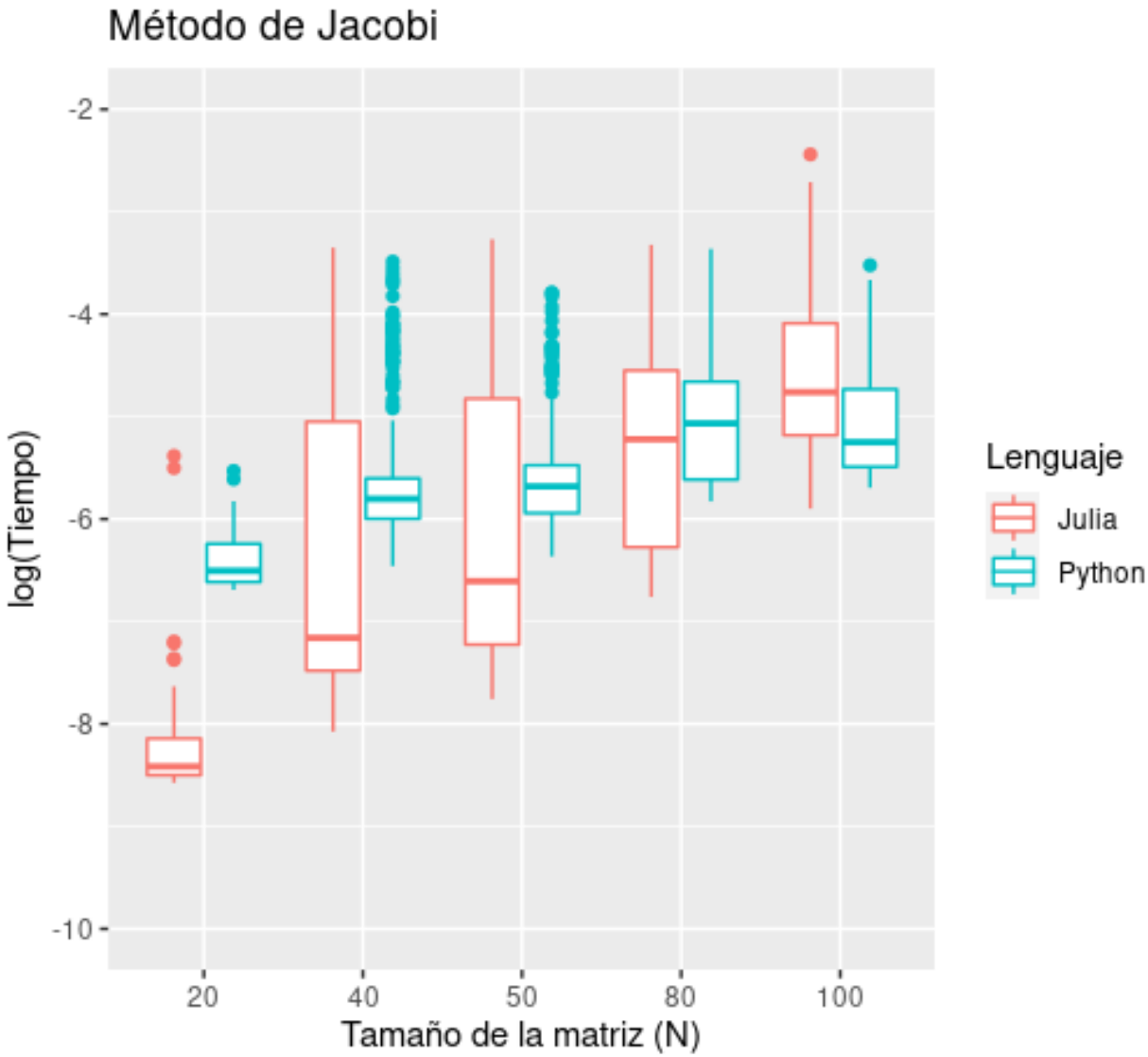



Figure: Diferencias entre las implementaciones en escala logaritmica.

# Resultados

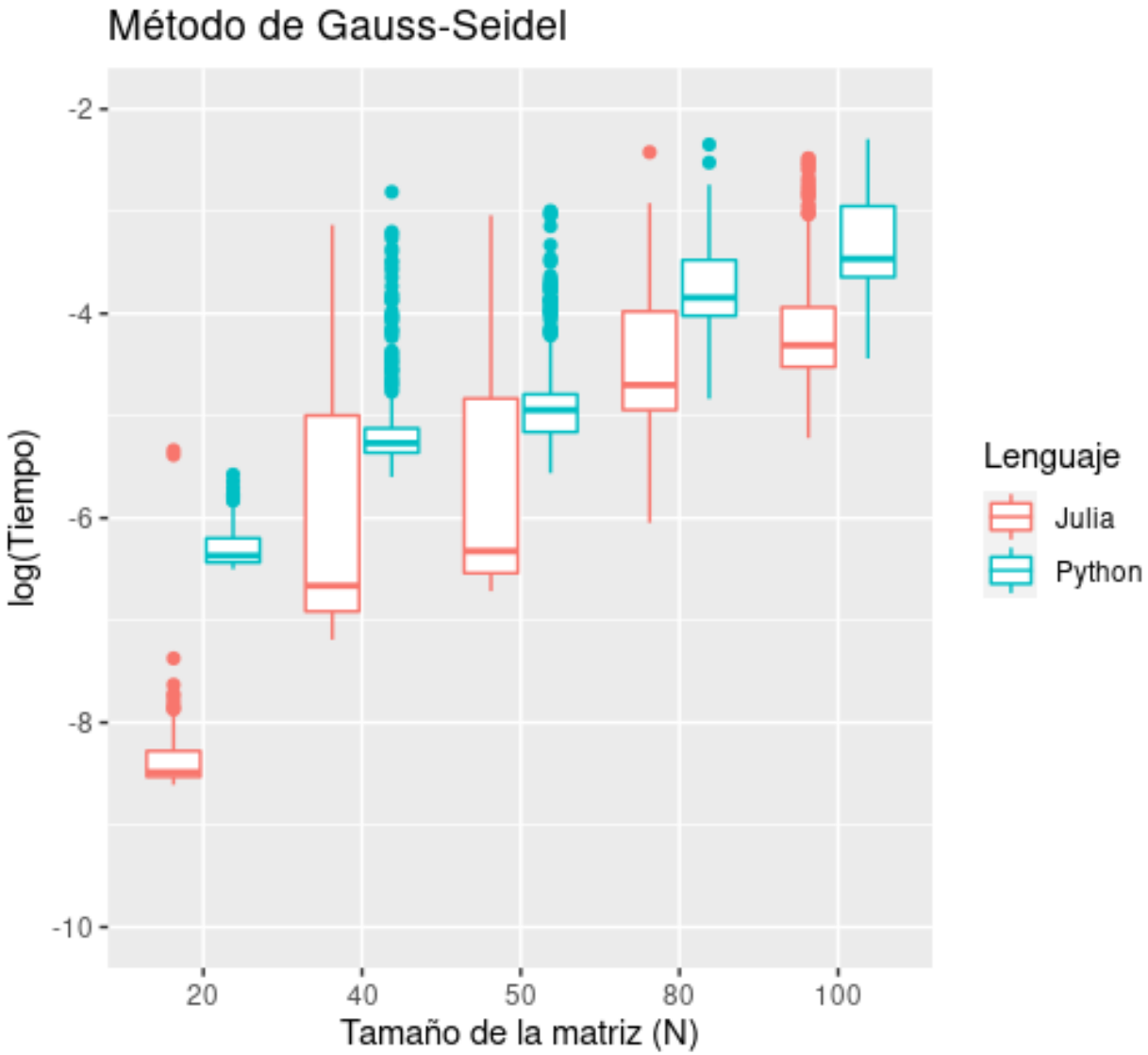



Figure: Diferencias entre las implementaciones en escala logaritmica

# Resultados

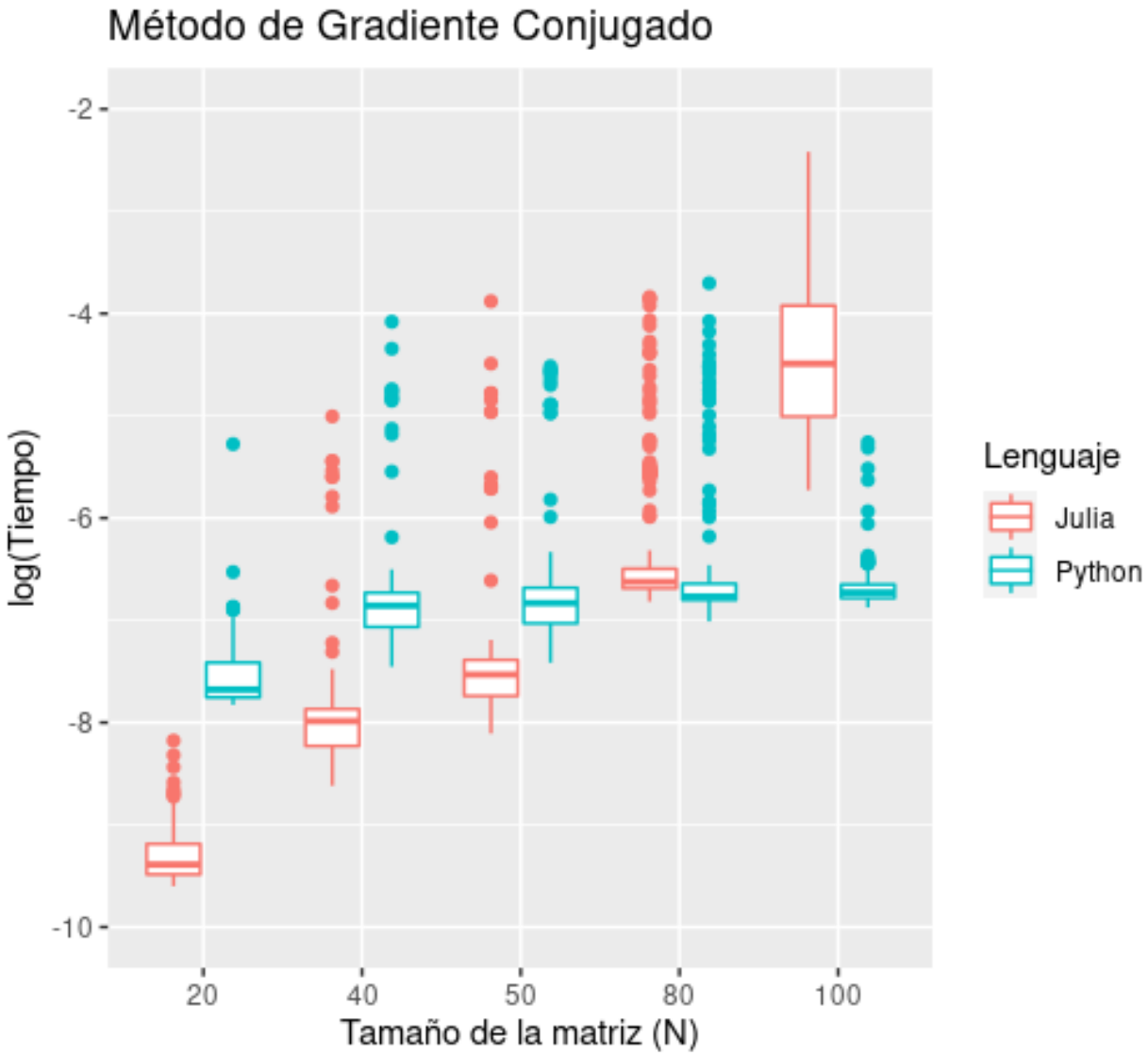



Figure: Diferencias entre las implementaciones en escala logaritmica.

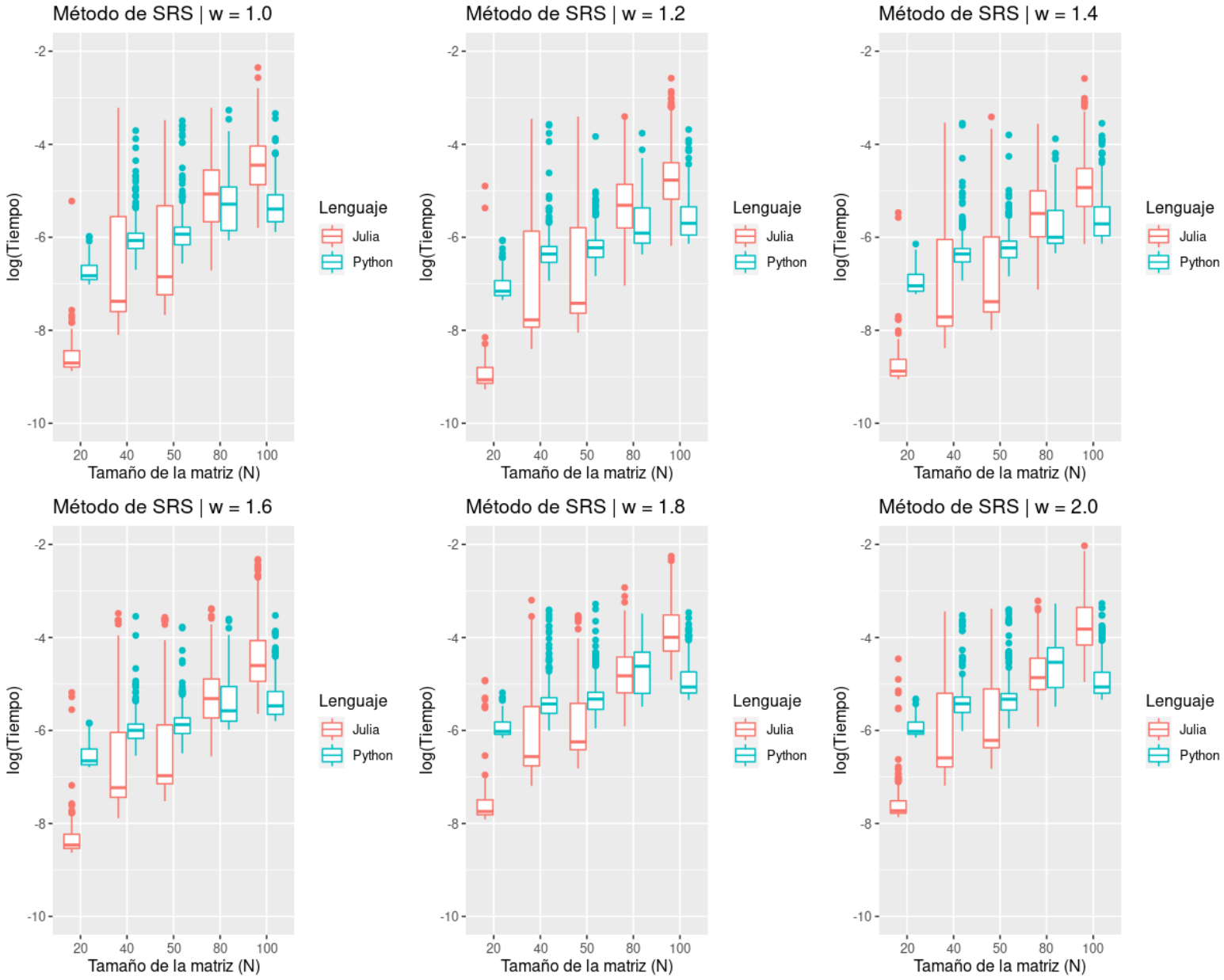



Figure: Diferencias entre las implementaciones en escala logaritmica.

# Conclusiones

Después de lo expuesto se puede concluir que:

- Julia es una buena opción para ejecutar algoritmos secuenciales.
- La implementación del algoritmo es relevante para el tiempo de ejecución.
- Conocer diferentes algoritmos de solución es necesario para manterner la calidad de los resultados en tiempo y forma.
- El método del gradiente conjugado es una gran opción para sistemas de ecuaciones lineales pequeños.
- Para sistemas grandes, hay algoritmos más apropiados.

# Conclusiones

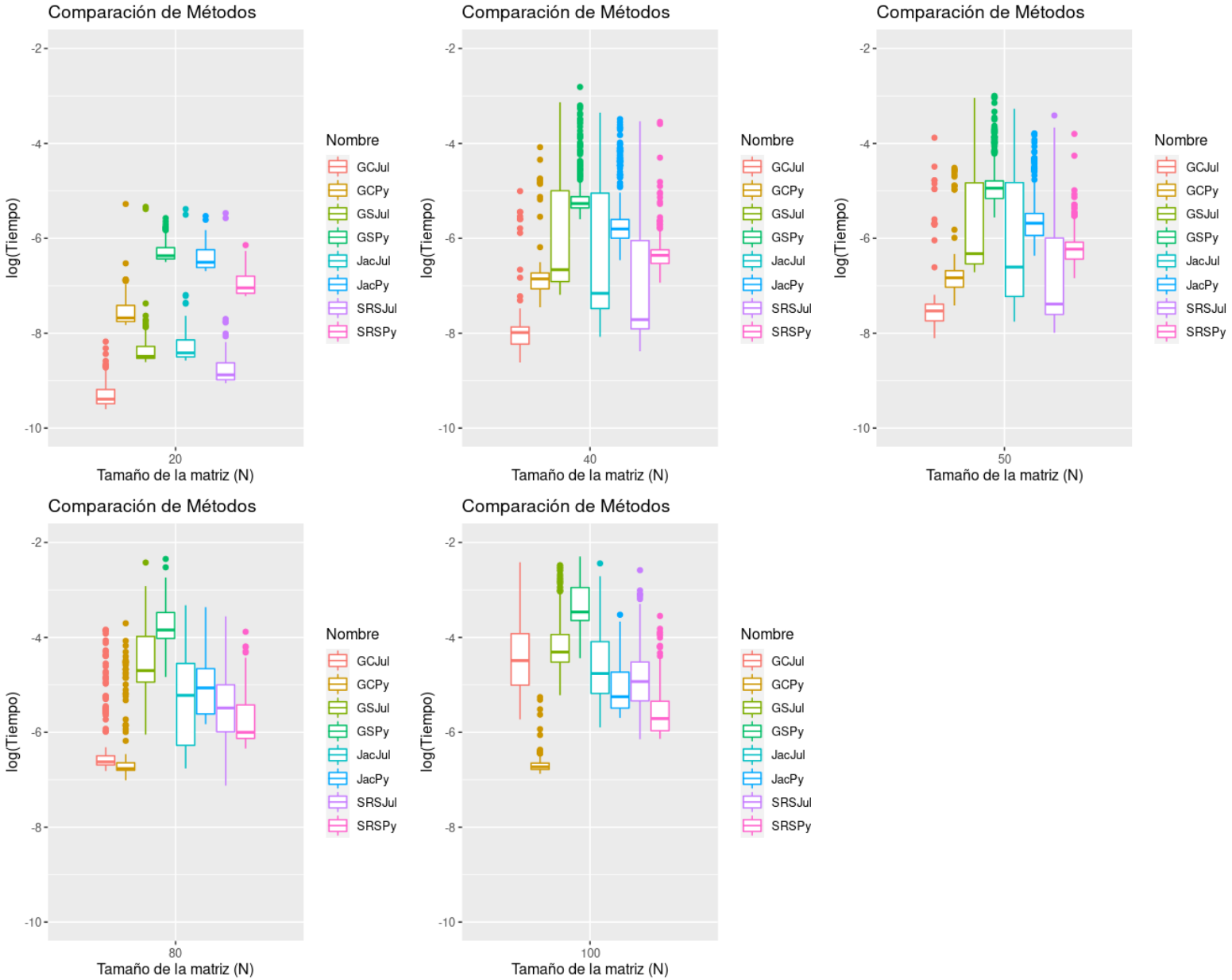



Figure: Diferencias entre las implementaciones en escala

# Bibliografía

[1] Sauer, Timothy, and Jesús Elmer Murrieta Murrieta. *Análisis numérico*. Pearson Educación, 2013.

[2] The Julia Programming Language [TheJuliaLanguage]. (2020, mayo 25). Julia for data science - video 2: Linear algebra, by Dr. Huda Nassar (for JuliaAcademy.Com). Youtube. https://www.youtube.com/watch?v=bndXPsRHPg0

[3] PyCall.jl: Package to call Python functions from the Julia language. (s/f).

[4] Shaw, Zed A. *Learn Python the hard way: A very simple introduction to the terrifyingly beautiful world of computers and code*. Addison-Wesley, 2013.

# Anexo A: Algoritmos para resolver sistemas de ecuaciones lineales implementados en Julia

## Paquetes

In [1]:
```
using LinearAlgebra, BenchmarkTools, Plots
```

## Métodos

Implementación del método de Jacobi.

In [2]:
```
function JacJul(
        A::Array{Float64,2},
        b::Array{Float64,1},
        x0::Array{Float64,1},
        e::Float64,
        itermax::Int64)

    #A:= Matriz de coeficientes
    #b:= vector de constantes
    #x0:= inicialización
    #e:= presicion
    #itermax:= iteración maxima

    #Iteración
    it::Int64 = 1

    #Matriz diagonal
    D::Array{Float64,2} = Diagonal(A)

    #Inversa de la matriz diagonal
    Dinv::Array{Float64,2} = inv(D)

    #Matriz triangular inferior
    L ::Array{Float64,2}= tril(A,-1)

    #Matriz triangular superior
    U::Array{Float64,2} = triu(A,1)

    #Iterando la solucion
    x::Array{Float64,1} = Dinv*(b- (L+U)*x0)

    #Calculando el residuo
    r::Array{Float64,1} = A*x-b

    #Normas de los residuos
    normresJAC::Array{Float64,1} = [norm(r)];

    #Repetir hasta el maximo numero de
    #iteraciones o alcanzar la precision
    while ((normresJAC[it] > e) && (it <= itermax))

        #Iterando la solucion
        x = Dinv*(b- (L+U)*x)

        #Calculando el residuo
        r = A*x-b

        #Normas de los residuos
        normresJAC = [normresJAC;norm(r)];

        #Aumentando la iteracion
        it = it + 1
    end

    return x, normresJAC

end;
```

Implementación de método de Sobre Relajación Sucesiva.

In [3]:
```
function SRSJul(
        A::Array{Float64,2},
        b::Array{Float64,1},
        x0::Array{Float64,1},
        w::Float64,
        e::Float64,
        itermax::Int64)

    #A:= Matriz de coeficientes
    #b:= vector de constantes
    #x0:= inicialización
    #w:= ponderador
    #e:= presicion
    #itermax:= iteración maxima

    #Iteración
    it::Int64 = 1

    #Matriz diagonal
    D::Array{Float64,2} = Diagonal(A)

    #Matriz triangular superior
    U::Array{Float64,2} = triu(A, 1)

    #Matriz triangular inferior
    L::Array{Float64,2} = tril(A, -1)

    #alfa es una matriz auxiliar
    alfa::Array{Float64,2} = inv(w*L+D)

    #gama es un vector auxiliar
    gama::Array{Float64,1} = w*inv(D+w*L)*b

    #Iterando la solucion
    x::Array{Float64,1} = alfa*((1-w)*D*x0-w*U*x0) + gama

    #Calculando el residuo
    r::Array{Float64,1} = A*x-b

    #Normas de los residuos
    normresSCS::Array{Float64,1} = [norm(r)];

    #Repetir hasta el maximo numero de
    #iteraciones o alcanzar la precision
    while ((normresSCS[it] > e) && (it <= itermax))

        #Multiplicando
        x = alfa*((1-w)*D*x-w*U*x) + gama

        #Calculando la diferencia
        r = A*x-b

        #Normas de los residuos
        normresSCS = [normresSCS; norm(r)];

        #Aumentando la iteracion
        it = it + 1
    end

    return x, normresSCS

end;
```

Implementación de método de Gauss - Seidel.

In [4]:

```
function GSJul(
        A::Array{Float64,2},
        b::Array{Float64,1},
        x0::Array{Float64,1},
        e::Float64,
        itermax::Int64)

    #A:= Matriz de coeficientes
    #b:= vector de constantes
    #x0:= inicialización
    #e:= presicion
    #itermax:= iteración maxima

    #Iteración
    it::Int64 = 1

    #Matriz triangular inferior
    L::Array{Float64,2} = tril(A)

    #Matriz triangular superior
    U::Array{Float64,2} = triu(A,1)

    #Iterando la solucion
    x::Array{Float64,1} = inv(L)*(b-U*x0);

    #Calculando el residuo
    r::Array{Float64,1} = b-A*x

    #Normas de los residuos
    normresGAUSS::Array{Float64,1} = [norm(r)];

    #Repetir hasta el maximo numero de
    #iteraciones o alcanzar la precision
    while ((normresGAUSS[it] > e) && (it <= itermax))

        #Iterando la solucion
        x = inv(L)*(b-U*x);

        #Calculando el residuo
        r = b - A*x

        #Normas de los residuos
        normresGAUSS = [normresGAUSS;norm(r)];

        #Aumentando la iteracion
        it = it + 1

    end

    return x, normresGAUSS
end;
```

Implementación de método de gradiente conjugado.

In [5]:

```
function GCJul(
        A::Array{Float64,2},
        b::Array{Float64,1},
        x0::Array{Float64,1},
        e::Float64)

    #A:= Matriz de coeficientes
    #b:= vector de constantes
    #x0:= inicialización
    #e:= presicion

    #Iteración
    it::Int64 = 1

    #numeq := numero de ecuaciones
    numeq::Int64 = size(A,1)

    #Iterando la solucion
    x::Array{Float64,1} = x0;

    #Calculando el residuo
    r::Array{Float64,1} = b - A*x

    #Vectores auxiliares
    d::Array{Float64,1} = r;
    s::Array{Float64,1} = r;

    #Normas de los residuos
    normresGC::Array{Float64,1} = [norm(r)];

    #Constantes auxiliares
    a::Float64 = 0.0
    beta::Float64 = 0.0

    #Por cada ecuacion, hacer lo siguiente:
    for k in 1:numeq

        #Si la norma del vector es mayor a
        #la presicion entonces
        if normresGC[it] > e

            #a:= constante auxiliar
            a = r'r/(d'*A*d)

            #Iterando la solucion
            x = x + a*d

            #s:= vector auxiliar
            s = r - a*A*d

            #beta:= constante auxiliar
            beta = s's/r'r

            #d:= vector auxiliar
            d = s + beta*d

            #actualizando los residuos
            r = s

            #Normas de los residuos
            normresGC = [normresGC; norm(r)];

        else
            return  x, normresGC
        end

    end

    return x, normresGC
end;
```

## Ejemplo:

Matriz de coeficientes (A)

In [6]:

```
A = [3.0 -1.0  0.0  0.0  0.0  0.5;
    -1.0  3.0 -1.0  0.0  0.5  0.0;
     0.0 -1.0  3.0 -1.0  0.0  0.0;
     0.0  0.0 -1.0  3.0 -1.0  0.0;
     0.0  0.5  0.0 -1.0  3.0 -1.0;
     0.5  0.0  0.0  0.0 -1.0  3.0;]
```

Out[6]:
```
6×6 Array{Float64,2}:
  3.0  -1.0   0.0   0.0   0.0   0.5
 -1.0   3.0  -1.0   0.0   0.5   0.0
  0.0  -1.0   3.0  -1.0   0.0   0.0
  0.0   0.0  -1.0   3.0  -1.0   0.0
  0.0   0.5   0.0  -1.0   3.0  -1.0
  0.5   0.0   0.0   0.0  -1.0   3.0
```

Vector de constantes (b)

In [7]:
```
b = [5/2 3/2 1 1 3/2 5/2]
b = b[1,:]
```

Out[7]:
```
6-element Array{Float64,1}:
 2.5
 1.5
 1.0
 1.0
 1.5
 2.5
```

Inicialización de la solución ($x_0$)

In [8]:
```
x0 = zeros(size(b))
```

Out[8]:
```
6-element Array{Float64,1}:
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
```

Error máximo (e)

In [9]:
```
e = 0.000001
```

Out[9]:
```
1.0e-6
```

Número máximo de iteraciones (itermax)

In [10]:
```
itermax  = 100
```

Out[10]:
```
100
```

Solución

In [11]:
```
A\b
```

Out[11]:
```
6-element Array{Float64,1}:
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
```

## ** Algoritmo de Jacobi **

In [12]:
```
SolJac,histJac = JacJul(A,b,x0,e,itermax);
```

Solución

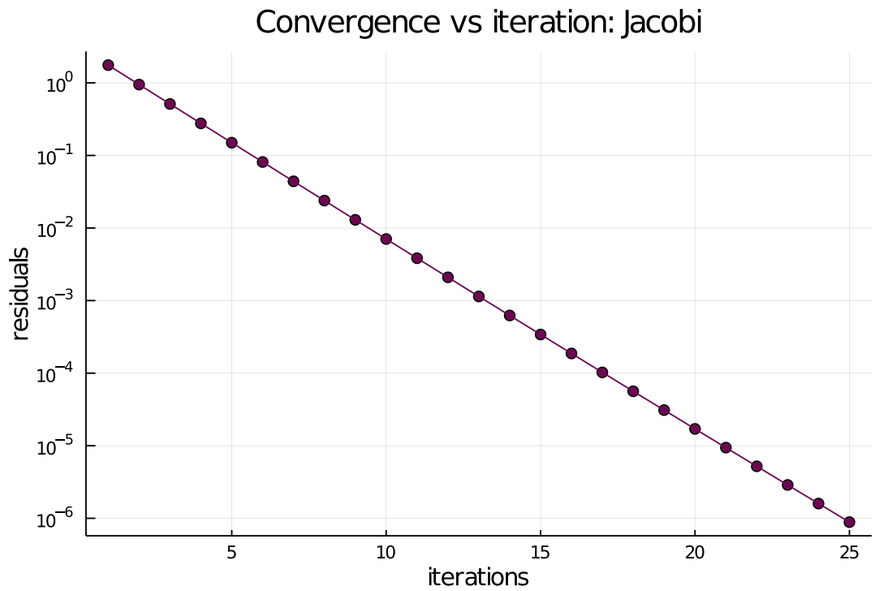
In [13]:
```
SolJac
```

Out[13]:
```
6-element Array{Float64,1}:
 0.9999999976171963
 0.9999997813867877
 0.9999997976579562
 0.9999997976579562
 0.9999997813867877
 0.9999999976171963
```

Precisión vs Iteraciones

In [14]:
```
scatter(histJac, markersize=4, c="#6b0851", leg=false)
plot!(histJac, color="#6b0851", leg=false )
plot!(xaxis=("iterations") , yaxis = ("residuals", :log))
plot!(title="Convergence vs iteration: Jacobi")
```

Out[14]:




## ** Algoritmo de Sobre-Relajación **

In [15]:
```
#Peso
w = 1.1;
SolSRS,histSRS = SRSJul(A,b,x0,w,e,itermax);
```

Solución

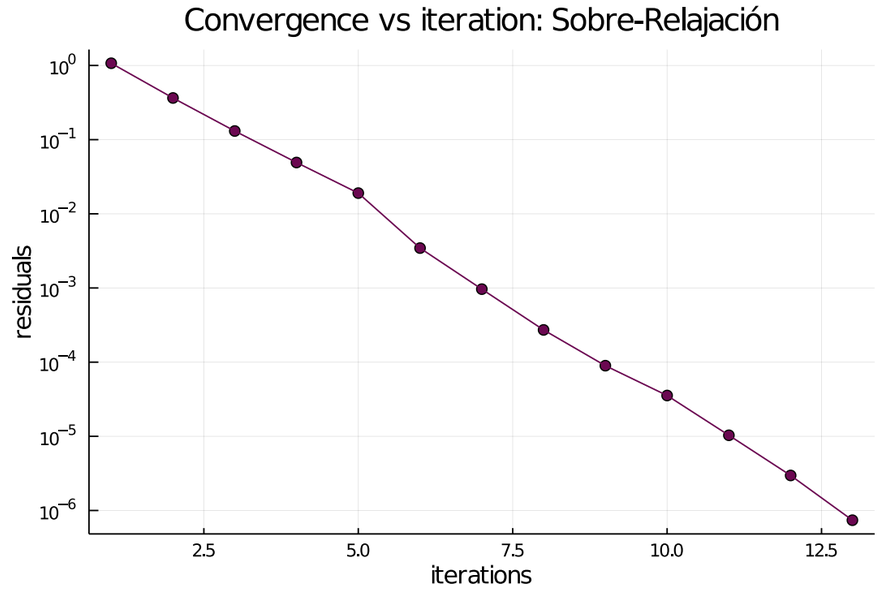
In [16]:
```
SolSRS
```

Out[16]:
```
6-element Array{Float64,1}:
 1.0000001279675337
 0.9999999438723751
 0.9999997391036273
 0.999999769408286
 0.9999998817019154
 0.99999996097816
```

Precisión vs Iteraciones

In [17]:
```
scatter(histSRS, markersize=4, c="#6b0851", leg=false)
plot!(histSRS, color="#6b0851", leg=false )
plot!(xaxis=("iterations") , yaxis = ("residuals", :log))
plot!(title="Convergence vs iteration: Sobre-Relajación")
```

Out[17]:




## ** Algoritmo de Gauss - Seidel **

In [18]:
```
SolGS,histGS = GSJul(A,b,x0,e,itermax);
```

Solución

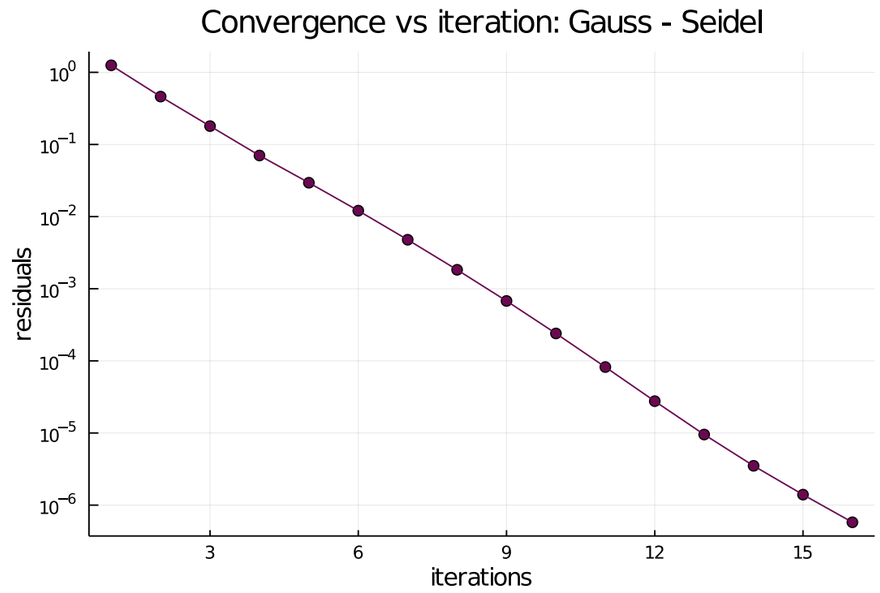
In [19]:
```
SolGS
```

Out[19]:
```
6-element Array{Float64,1}:
 1.0000002137455963
 1.0000002688269807
 1.0000002052803891
 1.000000079328917
 0.9999999631217477
 0.9999999520829831
```

Precisión vs Iteraciones

In [20]:
```
scatter(histGS, markersize=4, c="#6b0851", leg=false)
plot!(histGS, color="#6b0851", leg=false )
plot!(xaxis=("iterations") , yaxis = ("residuals", :log))
plot!(title="Convergence vs iteration: Gauss - Seidel")
```

Out[20]:




## ** Algoritmo del Gradiente Conjugado **

In [21]:
```
SolGC,histGC = GCJul(A,b,x0,e);
```

Solución por el método de Gradiente Conjugado

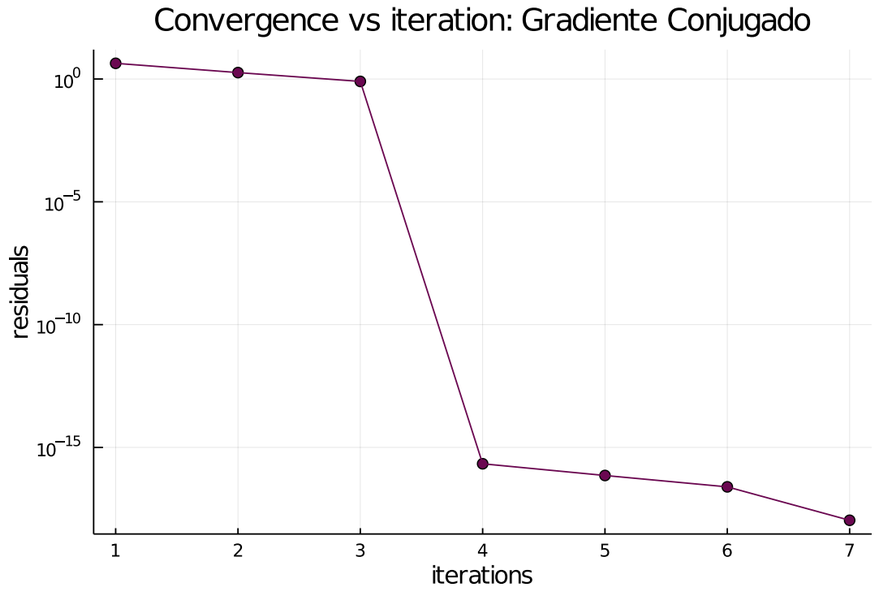
In [22]:
```
SolGC
```

Out[22]:
```
6-element Array{Float64,1}:
 1.0
 1.0
 0.9999999999999998
 0.9999999999999998
 1.0
 0.9999999999999999
```

Precisión vs Iteraciones

In [23]:
```
scatter(histGC, markersize=4, c="#6b0851", leg=false)
plot!(histGC, color="#6b0851", leg=false )
plot!(xaxis=("iterations") , yaxis = ("residuals", :log))
plot!(title="Convergence vs iteration: Gradiente Conjugado")
```

Out[23]:


Convergence vs iteration: Gradiente Conjugado
residuals
$10^{0}$
$10^{-5}$
$10^{-10}$
$10^{-15}$
1
2
3
4
5
6
7
iterations

# Anexo B: Algoritmos para resolver sistemas de ecuaciones lineales implementados en Python

## Paquetes

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

## Métodos

Implementación del método de Jacobi.

In [2]:
```
def JacPy(A,b,x0,e,itermax):

    #A:= Matriz de coeficientes
    #b:= vector de constantes
    #x0:= inicialización
    #e:= presicion
    #itermax:= iteración maxima

    #Iteración
    it = 1

    #Matriz diagonal
    D = np.diag(np.diag(A))

    #Inversa de la matriz diagonal
    Dinv = np.linalg.inv(D)

    #Matriz triangular superior
    U = np.triu(A, k=1)

    #Matriz triangular inferior
    L = np.tril(A, k=-1)

    #Iterando la solucion
    x = Dinv.dot(b - (L+U).dot(x0))

    #Calculando el residuo
    r = A.dot(x)- b

    #Normas de los residuos
    normresJAC = [np.linalg.norm(r,1)]

    #Repetir hasta el maximo numero de
    #iteraciones o alcanzar la precision
    while ((normresJAC[it-1] > e) and (it <= itermax)):

        #Iterando la solucion
        x = Dinv.dot(b - (L+U).dot(x))

        #Calculando la diferencia
        r = A.dot(x)- b

        #Normas de los residuos
        normresJAC = np.concatenate((normresJAC,
                                     np.linalg.norm(r,1)),
                                    axis = None)

        #Aumentando la iteracion
        it = it + 1

    return x, normresJAC
```

Implementación de método de Sobre Relajación Sucesiva.

In [3]:
```
def SRSPy(A,b,x0,w,e,itermax):

    #A:= Matriz
    #b:= vector
    #w:= ponderador
    #e:= presicion
    #itermax:= iteración máxima
    #x0:= inicialización

    #Iteración
    it = 1

    #Matriz diagonal
    D = np.diag(np.diag(A))

    #Matriz triangular superior
    U = np.triu(A, k=1)

    #Matriz triangular inferior
    L = np.tril(A, k=-1)

    #alfa es una matriz auxiliar
    alfa = np.linalg.inv(w*L+D)

    #gama es un vector auxiliar
    gama = w*np.linalg.inv(D+w*L).dot(b)

    #Iterando la solucion
    x = alfa.dot((1-w)*D.dot(x0)-w*U.dot(x0)) + gama

    #Calculando el residuo
    r = A.dot(x)-b

    #Normas de los residuos
    normresSCS = [np.linalg.norm(r,1)]

    #Repetir hasta el maximo numero de
    #iteraciones o alcanzar la precision
    while ((normresSCS[it-1] > e) and (it <= itermax)):

        #Multiplicando
        x = alfa.dot((1-w)*D.dot(x)-w*U.dot(x)) + gama

        #Calculando la diferencia
        r = A.dot(x)-b

        #Normas de los residuos
        normresSCS = np.concatenate((normresSCS,
                                     np.linalg.norm(r,1)),
                                    axis = None)

        #Aumentando la iteracion
        it = it + 1


    return x, normresSCS
```

Implementación de método de Gauss - Seidel.

In [4]:
```
def GSPy(A,b,x0,e,itermax):

    #A:= Matriz de coeficientes
    #b:= vector de constantes
    #x0:= inicialización
```

```
    #e:= presicion
    #itermax:= iteración maxima

    #Iteración
    it = 1

    #Matriz triangular inferior
    U = np.triu(A, k=1)

    #Matriz triangular superior
    L = np.tril(A)

    #Iterando la solución
    x = np.linalg.inv(L).dot(b-U.dot(x0));

    #Calculando el residuo
    r = b-A.dot(x)

    #Respaldando la norma
    normresGS = [np.linalg.norm(r,1)]

    #Repetir hasta el maximo numero de
    #iteraciones o alcanzar la precision
    while ((normresGS[it-1] > e) and (it <= itermax)):

        #Iterando la solucion
        x = np.linalg.inv(L).dot(b-U.dot(x));

        #Calculando el residuo
        r = b-A.dot(x)

        #Normas de los residuos
        normresGS = np.concatenate((normresGS,
                                    np.linalg.norm(r,1)),
                                   axis = None);

        #Aumentando la iteracion
        it = it + 1

    return x, normresGS
```

Implementación de método de gradiente conjugado.

In [5]:

```
def GCPy(A,b,x0,e):

    #A:= Matriz de coeficientes
    #b:= vector de constantes
    #x0:= inicialización
    #e:= presicion

    #Iteración
    it = 1

    #numeq := numero de ecuaciones
    numeq = A.shape[1]

    #Iterando la solucion
    x = x0;

    #Calculando el residuo
    r = b - A.dot(x)

    #Vectores auxiliares
    d = r;

    #Vector para guardar las convergencias
    normresGC = [np.linalg.norm(r,1)]

    #Por cada ecuacion, hacer lo siguiente:
    for k in range(numeq):

        #Si la norma del vector es mayor a
        #la presicion entonces
        if normresGC[it-1] > e:

            #a:= constante auxiliar
            a = np.transpose(r).dot(r)[0][0]/np.transpose(d).dot(A.dot(d))[0][0]

            #Iterando la solucion
            x = x + a*d

            #s:= vector auxiliar
            s = r - a*A.dot(d)

            #beta:= constante auxiliar
            beta = np.transpose(s).dot(s)[0][0]/np.transpose(r).dot(r)[0][0]

            #d:= vector auxiliar
            d = s + beta*d

            #actualizando los residuos
            r = s

            #Normas de los residuos
            normresGC = np.concatenate((normresGC,
                                        np.linalg.norm(r,1)),
                                       axis = None);

            it = it + 1

        else:

            return  x, normresGC


    return x, normresGC
```

# Ejemplo:

## Matriz de coeficientes (A)

In [6]:

```
A = np.array([[ 3.0, -1.0,  0.0,  0.0,  0.0,  0.5],
              [-1.0,  3.0, -1.0,  0.0,  0.5,  0.0],
              [ 0.0, -1.0,  3.0, -1.0,  0.0,  0.0],
              [ 0.0,  0.0, -1.0,  3.0, -1.0,  0.0],
              [ 0.0,  0.5,  0.0, -1.0,  3.0, -1.0],
              [ 0.5,  0.0,  0.0,  0.0, -1.0,  3.0]])
A
```

Out[6]:

```
array([[ 3. , -1. ,  0. ,  0. ,  0. ,  0.5],
       [-1. ,  3. , -1. ,  0. ,  0.5,  0. ],
       [ 0. , -1. ,  3. , -1. ,  0. ,  0. ],
       [ 0. ,  0. , -1. ,  3. , -1. ,  0. ],
       [ 0. ,  0.5,  0. , -1. ,  3. , -1. ],
       [ 0.5,  0. ,  0. ,  0. , -1. ,  3. ]])
```

## Vector de constantes (b)

In [7]:

```
b = np.array([[2.5], [1.5], [1], [1], [1.5], [2.5]])
b
```

Out[7]:

```
array([[2.5],
       [1.5],
       [1. ],
       [1. ],
       [1.5],
       [2.5]])
```

Inicialización de la solución ($x_0$)

In [8]:
```
x0 = np.zeros((A.shape[1], 1))
x0
```

Out[8]:
```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

Error máximo (e)

In [9]:
```
e = 0.000001
```

Número máximo de iteraciones (itermax)

In [10]:
```
itermax  = 100
```

Solución

In [11]:
```
np.linalg.solve(A, b)
```

Out[11]:
```
array([[1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.]])
```

## ** Algoritmo de Jacobi **

In [12]:
```
SolJac,histJac = JacPy(A,b,x0,e,itermax)
```

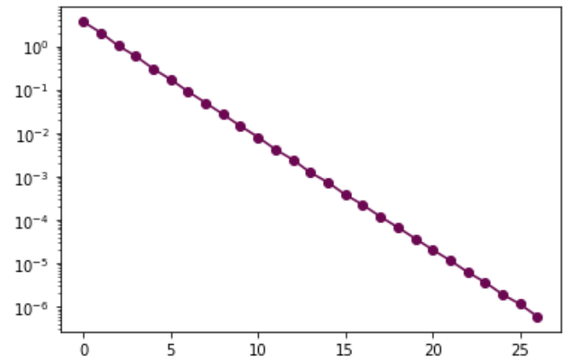
Solución

In [13]:
```
SolJac
```

Out[13]:
```
array([[1.        ],
       [0.99999993],
       [0.99999994],
       [0.99999994],
       [0.99999993],
       [1.        ]])
```

Precisión vs Iteraciones

In [14]:
```
plt.plot(histJac, color="#6b0851", marker='o')
plt.yscale('log')
plt.show()
```




## ** Algoritmo de Sobre-Relajación **

In [15]:
```
#Peso
w = 1.1;
SolSRS,histSRS = SRSPy(A,b,x0,w,e,itermax);
```

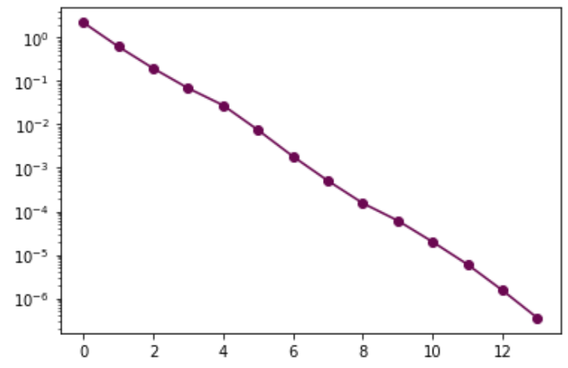
Solución

In [16]:
```
SolSRS
```

Out[16]:
```
array([[0.99999997],
       [0.99999992],
       [0.99999991],
       [0.99999995],
       [0.99999999],
       [1.00000001]])
```

Precisión vs Iteraciones

In [17]:
```
plt.plot(histSRS, color="#6b0851", marker='o')
plt.yscale('log')
plt.show()
```




## ** Algoritmo de Gauss - Seidel **

In [18]:
```
SolGS,histGS = GSPy(A,b,x0,e,itermax)
```

Solución

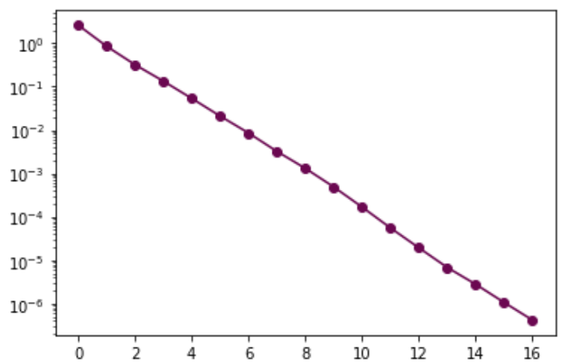
In [19]:
```
SolGS
```

Out[19]:
```
array([[1.0000001 ],
```

```
       [1.00000011],
       [1.00000006],
       [1.00000001],
       [0.99999997],
       [0.99999997]])
```

Precisión vs Iteraciones

In [20]:
```
plt.plot(histGS, color="#6b0851", marker='o')
plt.yscale('log')
plt.show()
```




## ** Algoritmo del Gradiente Conjugado **

In [21]:
```
SolGC,histGC = GCPy(A,b,x0,e)
```

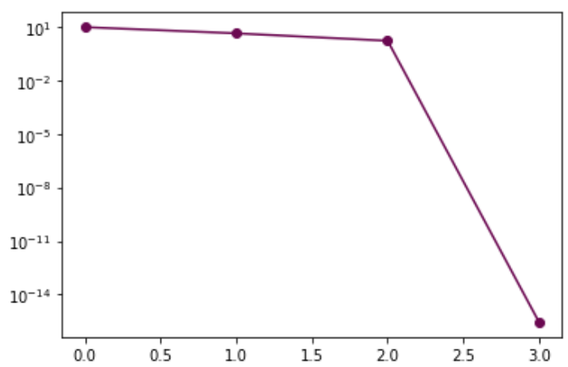
Solución

In [22]:
```
SolGC
```

Out[22]:
```
array([[1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.]])
```

Precisión vs Iteraciones

In [23]:
```
plt.plot(histGC, color="#6b0851", marker='o')
plt.yscale('log')
plt.show()
```



# Anexo C: Construcción de sistemas de ecuaciones lineales implementados en Julia

## Paquetes

In [1]:
```
using LinearAlgebra
```

## Métodos

Implementación del método de Jacobi.

In [2]:
```
function SistEQ(tam::Int64, x::Array{Float64,1})

    #tam:= tamanio del sistema de EQ
    #x:= solucion del sistema

    #Matriz de coeficientes (A)
    A::Array{Float64,2} = diagm(1 => -ones(tam-1),
                                0 => 3*ones(tam),
                               -1 => -ones(tam-1))

    for i in 1:tam
        if A[i,tam-i+1] == 0.0
            A[i,tam-i+1] = 0.5
        end
    end

    #Vector de constantes (b)
    b::Array{Float64,1} = A*x

    return A,b
end;
```

## Ejemplo:

### Tamaño de la entrada (tam)

In [3]:
```
tam = 5;
```

### Matriz de coeficientes (A)

In [4]:
```
A,b = SistEQ(tam,ones(tam))
A
```

Out[4]:
```
5×5 Array{Float64,2}:
  3.0  -1.0   0.0   0.0   0.5
 -1.0   3.0  -1.0   0.5   0.0
  0.0  -1.0   3.0  -1.0   0.0
  0.0   0.5  -1.0   3.0  -1.0
  0.5   0.0   0.0  -1.0   3.0
```

### Vector de constantes (b)

In [5]:
```
b
```

Out[5]:
```
5-element Array{Float64,1}:
 2.5
 1.5
 1.0
 1.5
 2.5
```

### Tamaño de la entrada (tam)

In [6]:
```
tam = 6;
```

### Matriz de coeficientes (A)

In [7]:
```
A,b = SistEQ(tam,ones(tam))
A
```

Out[7]:
```
6×6 Array{Float64,2}:
  3.0  -1.0   0.0   0.0   0.0   0.5
 -1.0   3.0  -1.0   0.0   0.5   0.0
  0.0  -1.0   3.0  -1.0   0.0   0.0
  0.0   0.0  -1.0   3.0  -1.0   0.0
  0.0   0.5   0.0  -1.0   3.0  -1.0
  0.5   0.0   0.0   0.0  -1.0   3.0
```

### Vector de constantes (b)

In [8]:
```
b
```

Out[8]:
```
6-element Array{Float64,1}:
 2.5
 1.5
 1.0
 1.0
 1.5
 2.5
```

### Tamaño de la entrada (tam)

In [9]:
```
tam = 10;
```

### Matriz de coeficientes (A)

In [10]:
```
A,b = SistEQ(tam,ones(tam))
A
```

Out[10]:
```
10×10 Array{Float64,2}:
  3.0  -1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.5
 -1.0   3.0  -1.0   0.0   0.0   0.0   0.0   0.0   0.5   0.0
  0.0  -1.0   3.0  -1.0   0.0   0.0   0.0   0.5   0.0   0.0
  0.0   0.0  -1.0   3.0  -1.0   0.0   0.5   0.0   0.0   0.0
  0.0   0.0   0.0  -1.0   3.0  -1.0   0.0   0.0   0.0   0.0
  0.0   0.0   0.0   0.0  -1.0   3.0  -1.0   0.0   0.0   0.0
  0.0   0.0   0.0   0.5   0.0  -1.0   3.0  -1.0   0.0   0.0
  0.0   0.0   0.5   0.0   0.0   0.0  -1.0   3.0  -1.0   0.0
  0.0   0.5   0.0   0.0   0.0   0.0   0.0  -1.0   3.0  -1.0
  0.5   0.0   0.0   0.0   0.0   0.0   0.0   0.0  -1.0   3.0
```

### Vector de constantes (b)

In [11]:
```
b
```

Out[11]:
```
10-element Array{Float64,1}:
 2.5
 1.5
 1.5
 1.5
 1.0
 1.0
 1.5
 1.5
 1.5
 2.5
```

# Anexo D: Implementación

## Paquetes

### Paquetes de Julia

PyCall nos permite llamar a Python desde Julia

In [1]:
```
#using Pkg
#Pkg.add(PackageSpec(name="PyCall", rev="master"))
#Pkg.build("PyCall")
#Pkg.add("CSV")
#Pkg.add("Tables")
```

In [3]:
```
using PyCall
```

In [4]:
```
using CSV, LinearAlgebra, Tables
```

### Paquetes de Python

In [5]:
```
py"""
import numpy as np
"""
```

## Métodos

A continuación se agregaran las funciones de Python y Julia, llamados desde Julia.

In [6]:
```
function JacJul(
        A::Array{Float64,2},
        b::Array{Float64,1},
        x0::Array{Float64,1},
        e::Float64,
        itermax::Int64)

    #A:= Matriz de coeficientes
    #b:= vector de constantes
    #x0:= inicialización
    #e:= presicion
    #itermax:= iteración maxima

    #Iteración
    it::Int64 = 1

    #Matriz diagonal
    D::Array{Float64,2} = Diagonal(A)

    #Inversa de la matriz diagonal
    Dinv::Array{Float64,2} = inv(D)

    #Matriz triangular inferior
    L ::Array{Float64,2}= tril(A,-1)

    #Matriz triangular superior
    U::Array{Float64,2} = triu(A,1)

    #Iterando la solucion
    x::Array{Float64,1} = Dinv*(b- (L+U)*x0)

    #Calculando el residuo
    r::Array{Float64,1} = A*x-b

    #Normas de los residuos
    normresJAC::Array{Float64,1} = [norm(r)];

    #Repetir hasta el maximo numero de
    #iteraciones o alcanzar la precision
    while ((normresJAC[it] > e) && (it <= itermax))

        #Iterando la solucion
        x = Dinv*(b- (L+U)*x)

        #Calculando el residuo
        r = A*x-b

        #Normas de los residuos
        normresJAC = [normresJAC;norm(r)];

        #Aumentando la iteracion
        it = it + 1

    end

    return x, normresJAC

end;
```

In [7]:
```
py"""
def JacPy(A,b,x0,e,itermax):

    #A:= Matriz de coeficientes
    #b:= vector de constantes
    #x0:= inicialización
    #e:= presicion
    #itermax:= iteración maxima

    #Iteración
    it = 1

    #Matriz diagonal
    D = np.diag(np.diag(A))

    #Inversa de la matriz diagonal
    Dinv = np.linalg.inv(D)

    #Matriz triangular superior
    U = np.triu(A, k=1)

    #Matriz triangular inferior
    L = np.tril(A, k=-1)

    #Iterando la solucion
    x = Dinv.dot(b - (L+U).dot(x0))

    #Calculando el residuo
    r = A.dot(x)- b

    #Normas de los residuos
    normresJAC = [np.linalg.norm(r,1)]

    #Repetir hasta el maximo numero de
    #iteraciones o alcanzar la precision
    while ((normresJAC[it-1] > e) and (it <= itermax)):

        #Iterando la solucion
        x = Dinv.dot(b - (L+U).dot(x))

        #Calculando la diferencia
        r = A.dot(x)- b

        #Normas de los residuos
        normresJAC = np.concatenate((normresJAC,
                                     np.linalg.norm(r,1)),
                                    axis = None)

        #Aumentando la iteracion
        it = it + 1

    return x, normresJAC
"""
```

In [8]:
```
function SRSJul(
        A::Array{Float64,2},
        b::Array{Float64,1},
        x0::Array{Float64,1},
        w::Float64,
        e::Float64,
        itermax::Int64)

    #A:= Matriz de coeficientes
    #b:= vector de constantes
    #x0:= inicialización
    #w:= ponderador
    #e:= presicion
```

```
    #itermax:= iteración maxima

    #Iteración
    it::Int64 = 1

    #Matriz diagonal
    D::Array{Float64,2} = Diagonal(A)

    #Matriz triangular superior
    U::Array{Float64,2} = triu(A, 1)

    #Matriz triangular inferior
    L::Array{Float64,2} = tril(A, -1)

    #alfa es una matriz auxiliar
    alfa::Array{Float64,2} = inv(w*L+D)

    #gama es un vector auxiliar
    gama::Array{Float64,1} = w*inv(D+w*L)*b

    #Iterando la solucion
    x::Array{Float64,1} = alfa*((1-w)*D*x0-w*U*x0) + gama

    #Calculando el residuo
    r::Array{Float64,1} = A*x-b

    #Normas de los residuos
    normresSCS::Array{Float64,1} = [norm(r)];

    #Repetir hasta el maximo numero de
    #iteraciones o alcanzar la precision
    while ((normresSCS[it] > e) && (it <= itermax))

        #Multiplicando
        x = alfa*((1-w)*D*x-w*U*x) + gama

        #Calculando la diferencia
        r = A*x-b

        #Normas de los residuos
        normresSCS = [normresSCS; norm(r)];

        #Aumentando la iteracion
        it = it + 1
    end

    return x, normresSCS

end;
```

In [9]:

```
py"""
def SRSPy(A,b,x0,w,e,itermax):

    #A:= Matriz
    #b:= vector
    #w:= ponderador
    #e:= presicion
    #itermax:= iteración máxima
    #x0:= inicialización

    #Iteración
    it = 1

    #Matriz diagonal
    D = np.diag(np.diag(A))

    #Matriz triangular superior
    U = np.triu(A, k=1)

    #Matriz triangular inferior
    L = np.tril(A, k=-1)

    #alfa es una matriz auxiliar
    alfa = np.linalg.inv(w*L+D)

    #gama es un vector auxiliar
    gama = w*np.linalg.inv(D+w*L).dot(b)

    #Iterando la solucion
    x = alfa.dot((1-w)*D.dot(x0)-w*U.dot(x0)) + gama

    #Calculando el residuo
    r = A.dot(x)-b

    #Normas de los residuos
    normresSCS = [np.linalg.norm(r,1)]

    #Repetir hasta el maximo numero de
    #iteraciones o alcanzar la precision
    while ((normresSCS[it-1] > e) and (it <= itermax)):

        #Multiplicando
        x = alfa.dot((1-w)*D.dot(x)-w*U.dot(x)) + gama

        #Calculando la diferencia
        r = A.dot(x)-b

        #Normas de los residuos
        normresSCS = np.concatenate((normresSCS,
                                     np.linalg.norm(r,1)),
                                    axis = None)

        #Aumentando la iteracion
        it = it + 1


    return x, normresSCS
"""
```

In [10]:

```
function GSJul(
        A::Array{Float64,2},
        b::Array{Float64,1},
        x0::Array{Float64,1},
        e::Float64,
        itermax::Int64)

    #A:= Matriz de coeficientes
    #b:= vector de constantes
    #x0:= inicialización
    #e:= presicion
    #itermax:= iteración maxima

    #Iteración
    it::Int64 = 1

    #Matriz triangular inferior
    L::Array{Float64,2} = tril(A)

    #Matriz triangular superior
    U::Array{Float64,2} = triu(A,1)

    #Iterando la solucion
    x::Array{Float64,1} = inv(L)*(b-U*x0);

    #Calculando el residuo
    r::Array{Float64,1} = b-A*x

    #Normas de los residuos
    normresGAUSS::Array{Float64,1} = [norm(r)];

    #Repetir hasta el maximo numero de
    #iteraciones o alcanzar la precision
    while ((normresGAUSS[it] > e) && (it <= itermax))

        #Iterando la solucion
        x = inv(L)*(b-U*x);

        #Calculando el residuo
        r = b - A*x

        #Normas de los residuos
        normresGAUSS = [normresGAUSS;norm(r)];

        #Aumentando la iteracion
        it = it + 1

    end

    return x, normresGAUSS
end;
```

In [11]:

```
py"""
def GSPy(A,b,x0,e,itermax):
```

```
    #A:= Matriz de coeficientes
    #b:= vector de constantes
    #x0:= inicialización
    #e:= presicion
    #itermax:= iteración maxima

    #Iteración
    it = 1

    #Matriz triangular inferior
    U = np.triu(A, k=1)

    #Matriz triangular superior
    L = np.tril(A)

    #Iterando la solución
    x = np.linalg.inv(L).dot(b-U.dot(x0));

    #Calculando el residuo
    r = b-A.dot(x)

    #Respaldando la norma
    normresGS = [np.linalg.norm(r,1)]

    #Repetir hasta el maximo numero de
    #iteraciones o alcanzar la precision
    while ((normresGS[it-1] > e) and (it <= itermax)):

        #Iterando la solucion
        x = np.linalg.inv(L).dot(b-U.dot(x));

        #Calculando el residuo
        r = b-A.dot(x)

        #Normas de los residuos
        normresGS = np.concatenate((normresGS,
                                    np.linalg.norm(r,1)),
                                   axis = None);

        #Aumentando la iteracion
        it = it + 1

    return x, normresGS
"""
```

In [12]:

```
function GCJul(
        A::Array{Float64,2},
        b::Array{Float64,1},
        x0::Array{Float64,1},
        e::Float64)

    #A:= Matriz de coeficientes
    #b:= vector de constantes
    #x0:= inicialización
    #e:= presicion

    #Iteración
    it::Int64 = 1

    #numeq := numero de ecuaciones
    numeq::Int64 = size(A,1)

    #Iterando la solucion
    x::Array{Float64,1} = x0;

    #Calculando el residuo
    r::Array{Float64,1} = b - A*x

    #Vectores auxiliares
    d::Array{Float64,1} = r;
    s::Array{Float64,1} = r;

    #Normas de los residuos
    normresGC::Array{Float64,1} = [norm(r)];

    #Constantes auxiliares
    a::Float64 = 0.0
    beta::Float64 = 0.0

    #Por cada ecuacion, hacer lo siguiente:
    for k in 1:numeq

        #Si la norma del vector es mayor a
        #la presicion entonces
        if normresGC[it] > e

            #a:= constante auxiliar
            a = r'r/(d'*A*d)

            #Iterando la solucion
            x = x + a*d

            #s:= vector auxiliar
            s = r - a*A*d

            #beta:= constante auxiliar
            beta = s's/r'r

            #d:= vector auxiliar
            d = s + beta*d

            #actualizando los residuos
            r = s

            #Normas de los residuos
            normresGC = [normresGC; norm(r)];

        else
            return  x, normresGC
        end

    end

    return x, normresGC
end;
```

In [13]:

```
py"""
def GCPy(A,b,x0,e):

    #A:= Matriz de coeficientes
    #b:= vector de constantes
    #x0:= inicialización
    #e:= presicion

    #Iteración
    it = 1

    #numeq := numero de ecuaciones
    numeq = A.shape[1]

    #Iterando la solucion
    x = x0;

    #Calculando el residuo
    r = b - A.dot(x)

    #Vectores auxiliares
    d = r;

    #Vector para guardar las convergencias
    normresGC = [np.linalg.norm(r,1)]

    #Por cada ecuacion, hacer lo siguiente:
    for k in range(numeq):

        #Si la norma del vector es mayor a
        #la presicion entonces
        if normresGC[it-1] > e:

            #a:= constante auxiliar
            a = np.transpose(r).dot(r)/np.transpose(d).dot(A.dot(d))

            #Iterando la solucion
            x = x + a*d

            #s:= vector auxiliar
            s = r - a*A.dot(d)

            #beta:= constante auxiliar
            beta = np.transpose(s).dot(s)/np.transpose(r).dot(r)

            #d:= vector auxiliar
            d = s + beta*d

            #actualizando los residuos
            r = s
```

```
            #Normas de los residuos
            normresGC = np.concatenate((normresGC,
                                np.linalg.norm(r,1)),
                               axis = None);

            it = it + 1

        else:

            return  x, normresGC


    return x, normresGC
"""
```

In [14]:
```
function SistEQ(tam::Int64, x::Array{Float64,1})

    #tam:= tamanio del sistema de EQ
    #x:= solucion del sistema

    #Matriz de coeficientes (A)
    A::Array{Float64,2} = diagm(1 => -ones(tam-1),
                                0 => 3*ones(tam),
                               -1 => -ones(tam-1))

    for i in 1:tam
        if A[i,tam-i+1] == 0.0
            A[i,tam-i+1] = 0.5
        end
    end

    #Vector de constantes (b)
    b::Array{Float64,1} = A*x

    return A,b
end;
```

## Evaluando los algoritmos e implementaciones

### Precisión fija (e)

In [15]:
```
e = 0.000001;
```

### Número máximo de iteraciones (itermax)

In [16]:
```
itermax = 100;
```

### Número de simulaciones

In [17]:
```
NSim = 100;
```

## Ejemplo

Se ejecutaron algunos ejemplos para que Julia constriya sus archivos temporales.

Primero se construiyen los parámetros de los algoritmos:

In [18]:
```
tam = 10;
A,b = SistEQ(tam,ones(tam));
x0 = zeros(tam);
w = 1.1;
```

Ejecutando los ejemplos:

In [19]:
```
@time sol,hist = JacJul(A,b,x0,e,itermax);
```

```
  0.078937 seconds (151.76 k allocations: 7.691 MiB)
```

In [20]:
```
@time sol,hist = py"JacPy"(A,b,x0,e,itermax);
```

```
  0.914357 seconds (2.15 M allocations: 109.166 MiB, 2.72% gc time)
```

In [21]:
```
@time sol,hist = SRSJul(A,b,x0,w,e,itermax);
```

```
  0.000092 seconds (414 allocations: 66.359 KiB)
```

In [22]:
```
@time sol,hist = py"SRSPy"(A,b,x0,w,e,itermax);
```

```
  0.023760 seconds (29.40 k allocations: 1.519 MiB)
```

In [23]:
```
@time sol,hist = GSJul(A,b,x0,e,itermax);
```

```
  0.000094 seconds (401 allocations: 45.031 KiB)
```

In [24]:
```
@time sol,hist = py"GSPy"(A,b,x0,e,itermax);
```

```
  0.000998 seconds (99 allocations: 4.672 KiB)
```

In [25]:
```
@time sol,hist = GCJul(A,b,x0,e);
```

```
  0.000058 seconds (254 allocations: 27.703 KiB)
```

In [26]:
```
@time sol,hist = py"GCPy"(A,b,x0,e);
```

```
  0.025329 seconds (29.36 k allocations: 1.518 MiB)
```

## Simulando

### Creando una tabla para respaldar los datos

Id:= Id de la simulación.

Nombre:= Nombre del algoritmo y simulación.

Ejecucion:= Julia ó Python.

N:= Tamaño de la entrada 20,40,50,80 ó 100.

Tiempo:= Tiempo de ejecución.

NumIte:= Número de iteraciones.

w:= Peso del método de sobre relajación.

In [27]:
```
Datos = Array{Any}(undef,5*18*NSim, 7);
```

Simulando

In [28]:
```
#Id del ejercicio.
id = 1

#n corre sobre el tamanio de la entrada
for n in [20,40,50,80,100]

    #Incializando la solución
    x0 = zeros(n)

    #Solución real
    x = ones(n)

    #Construyendo el sistema de ecuaciones
    A,b = SistEQ(n,x);

    #s cuenta el numero de simulaciones
    for s in 1:NSim

         #Ejecutando JacJul
        x0 = zeros(n)
        tiempo = @elapsed sol,hist = JacJul(A,b,x0,e,itermax);
        Datos[id,:] = [id, "JacJul", "Julia", n, tiempo, size(hist,1),0.0]
        id = id + 1

        #Ejecutando JacPy
        tiempo = @elapsed sol,hist = py"JacPy"(A,b,x0,e,itermax);
        Datos[id,:] = [id, "JacPy", "Python", n, tiempo, size(hist,1),0.0]
        id = id + 1
```

```
        for w in [1.0,1.2,1.4,1.6,1.8,2.0]

            #Ejecutando SRSJul
            tiempo = @elapsed sol,hist = SRSJul(A,b,x0,w,e,itermax);
            Datos[id,:] = [id, "SRSJul", "Julia", n, tiempo, size(hist,1),w]
            id = id + 1

            #Ejecutando SRSPy
            tiempo = @elapsed sol,hist = py"SRSPy"(A,b,x0,w,e,itermax);
            Datos[id,:] = [id, "SRSPy", "Python", n, tiempo, size(hist,1),w]
            id = id + 1

        end

        #Ejecutando GSJul
        tiempo = @elapsed sol,hist = GSJul(A,b,x0,e,itermax);
        Datos[id,:] = [id, "GSJul", "Julia", n, tiempo, size(hist,1),0.0]
        id = id + 1

        #Ejecutando GSPy
        tiempo = @elapsed sol,hist = py"GSPy"(A,b,x0,e,itermax);
        Datos[id,:] = [id, "GSPy", "Python", n, tiempo, size(hist,1),0.0]
        id = id + 1

        #Ejecutando GCJul
        tiempo = @elapsed sol,hist = GCJul(A,b,x0,e);
        Datos[id,:] = [id, "GCJul", "Julia", n, tiempo, size(hist,1),0.0]
        id = id + 1

        #Ejecutando GCJul
        tiempo = @elapsed sol,hist = py"GCPy"(A,b,x0,e);
        Datos[id,:] = [id, "GCPy", "Python", n, tiempo, size(hist,1),0.0]
        id = id + 1

        #Imprimiendo progreso
        println("Tamanio N = ",n," Núm de Sim s = ",s)
    end

end
```

```
Tamanio N = 20 Núm de Sim s = 1
Tamanio N = 20 Núm de Sim s = 2
Tamanio N = 20 Núm de Sim s = 3
Tamanio N = 20 Núm de Sim s = 4
Tamanio N = 20 Núm de Sim s = 5
Tamanio N = 20 Núm de Sim s = 6
Tamanio N = 20 Núm de Sim s = 7
Tamanio N = 20 Núm de Sim s = 8
Tamanio N = 20 Núm de Sim s = 9
Tamanio N = 20 Núm de Sim s = 10
Tamanio N = 20 Núm de Sim s = 11
Tamanio N = 20 Núm de Sim s = 12
Tamanio N = 20 Núm de Sim s = 13
Tamanio N = 20 Núm de Sim s = 14
Tamanio N = 20 Núm de Sim s = 15
Tamanio N = 20 Núm de Sim s = 16
Tamanio N = 20 Núm de Sim s = 17
Tamanio N = 20 Núm de Sim s = 18
Tamanio N = 20 Núm de Sim s = 19
Tamanio N = 20 Núm de Sim s = 20
Tamanio N = 20 Núm de Sim s = 21
Tamanio N = 20 Núm de Sim s = 22
Tamanio N = 20 Núm de Sim s = 23
Tamanio N = 20 Núm de Sim s = 24
Tamanio N = 20 Núm de Sim s = 25
Tamanio N = 20 Núm de Sim s = 26
Tamanio N = 20 Núm de Sim s = 27
Tamanio N = 20 Núm de Sim s = 28
Tamanio N = 20 Núm de Sim s = 29
Tamanio N = 20 Núm de Sim s = 30
Tamanio N = 20 Núm de Sim s = 31
Tamanio N = 20 Núm de Sim s = 32
Tamanio N = 20 Núm de Sim s = 33
Tamanio N = 20 Núm de Sim s = 34
Tamanio N = 20 Núm de Sim s = 35
Tamanio N = 20 Núm de Sim s = 36
Tamanio N = 20 Núm de Sim s = 37
Tamanio N = 20 Núm de Sim s = 38
Tamanio N = 20 Núm de Sim s = 39
Tamanio N = 20 Núm de Sim s = 40
Tamanio N = 20 Núm de Sim s = 41
Tamanio N = 20 Núm de Sim s = 42
Tamanio N = 20 Núm de Sim s = 43
Tamanio N = 20 Núm de Sim s = 44
Tamanio N = 20 Núm de Sim s = 45
Tamanio N = 20 Núm de Sim s = 46
Tamanio N = 20 Núm de Sim s = 47
Tamanio N = 20 Núm de Sim s = 48
Tamanio N = 20 Núm de Sim s = 49
Tamanio N = 20 Núm de Sim s = 50
Tamanio N = 20 Núm de Sim s = 51
Tamanio N = 20 Núm de Sim s = 52
Tamanio N = 20 Núm de Sim s = 53
Tamanio N = 20 Núm de Sim s = 54
Tamanio N = 20 Núm de Sim s = 55
Tamanio N = 20 Núm de Sim s = 56
Tamanio N = 20 Núm de Sim s = 57
Tamanio N = 20 Núm de Sim s = 58
Tamanio N = 20 Núm de Sim s = 59
Tamanio N = 20 Núm de Sim s = 60
Tamanio N = 20 Núm de Sim s = 61
Tamanio N = 20 Núm de Sim s = 62
Tamanio N = 20 Núm de Sim s = 63
Tamanio N = 20 Núm de Sim s = 64
Tamanio N = 20 Núm de Sim s = 65
Tamanio N = 20 Núm de Sim s = 66
Tamanio N = 20 Núm de Sim s = 67
Tamanio N = 20 Núm de Sim s = 68
Tamanio N = 20 Núm de Sim s = 69
Tamanio N = 20 Núm de Sim s = 70
Tamanio N = 20 Núm de Sim s = 71
Tamanio N = 20 Núm de Sim s = 72
Tamanio N = 20 Núm de Sim s = 73
Tamanio N = 20 Núm de Sim s = 74
Tamanio N = 20 Núm de Sim s = 75
Tamanio N = 20 Núm de Sim s = 76
Tamanio N = 20 Núm de Sim s = 77
Tamanio N = 20 Núm de Sim s = 78
Tamanio N = 20 Núm de Sim s = 79
Tamanio N = 20 Núm de Sim s = 80
Tamanio N = 20 Núm de Sim s = 81
Tamanio N = 20 Núm de Sim s = 82
Tamanio N = 20 Núm de Sim s = 83
Tamanio N = 20 Núm de Sim s = 84
Tamanio N = 20 Núm de Sim s = 85
Tamanio N = 20 Núm de Sim s = 86
Tamanio N = 20 Núm de Sim s = 87
Tamanio N = 20 Núm de Sim s = 88
Tamanio N = 20 Núm de Sim s = 89
Tamanio N = 20 Núm de Sim s = 90
Tamanio N = 20 Núm de Sim s = 91
Tamanio N = 20 Núm de Sim s = 92
Tamanio N = 20 Núm de Sim s = 93
Tamanio N = 20 Núm de Sim s = 94
Tamanio N = 20 Núm de Sim s = 95
Tamanio N = 20 Núm de Sim s = 96
Tamanio N = 20 Núm de Sim s = 97
Tamanio N = 20 Núm de Sim s = 98
Tamanio N = 20 Núm de Sim s = 99
Tamanio N = 20 Núm de Sim s = 100
Tamanio N = 40 Núm de Sim s = 1
Tamanio N = 40 Núm de Sim s = 2
Tamanio N = 40 Núm de Sim s = 3
Tamanio N = 40 Núm de Sim s = 4
Tamanio N = 40 Núm de Sim s = 5
Tamanio N = 40 Núm de Sim s = 6
Tamanio N = 40 Núm de Sim s = 7
Tamanio N = 40 Núm de Sim s = 8
Tamanio N = 40 Núm de Sim s = 9
Tamanio N = 40 Núm de Sim s = 10
Tamanio N = 40 Núm de Sim s = 11
Tamanio N = 40 Núm de Sim s = 12
Tamanio N = 40 Núm de Sim s = 13
Tamanio N = 40 Núm de Sim s = 14
Tamanio N = 40 Núm de Sim s = 15
Tamanio N = 40 Núm de Sim s = 16
Tamanio N = 40 Núm de Sim s = 17
Tamanio N = 40 Núm de Sim s = 18
Tamanio N = 40 Núm de Sim s = 19
Tamanio N = 40 Núm de Sim s = 20
Tamanio N = 40 Núm de Sim s = 21
Tamanio N = 40 Núm de Sim s = 22
Tamanio N = 40 Núm de Sim s = 23
Tamanio N = 40 Núm de Sim s = 24
Tamanio N = 40 Núm de Sim s = 25
Tamanio N = 40 Núm de Sim s = 26
Tamanio N = 40 Núm de Sim s = 27
```

```
Tamanio N = 40 Núm de Sim s = 28
Tamanio N = 40 Núm de Sim s = 29
Tamanio N = 40 Núm de Sim s = 30
Tamanio N = 40 Núm de Sim s = 31
Tamanio N = 40 Núm de Sim s = 32
Tamanio N = 40 Núm de Sim s = 33
Tamanio N = 40 Núm de Sim s = 34
Tamanio N = 40 Núm de Sim s = 35
Tamanio N = 40 Núm de Sim s = 36
Tamanio N = 40 Núm de Sim s = 37
Tamanio N = 40 Núm de Sim s = 38
Tamanio N = 40 Núm de Sim s = 39
Tamanio N = 40 Núm de Sim s = 40
Tamanio N = 40 Núm de Sim s = 41
Tamanio N = 40 Núm de Sim s = 42
Tamanio N = 40 Núm de Sim s = 43
Tamanio N = 40 Núm de Sim s = 44
Tamanio N = 40 Núm de Sim s = 45
Tamanio N = 40 Núm de Sim s = 46
Tamanio N = 40 Núm de Sim s = 47
Tamanio N = 40 Núm de Sim s = 48
Tamanio N = 40 Núm de Sim s = 49
Tamanio N = 40 Núm de Sim s = 50
Tamanio N = 40 Núm de Sim s = 51
Tamanio N = 40 Núm de Sim s = 52
Tamanio N = 40 Núm de Sim s = 53
Tamanio N = 40 Núm de Sim s = 54
Tamanio N = 40 Núm de Sim s = 55
Tamanio N = 40 Núm de Sim s = 56
Tamanio N = 40 Núm de Sim s = 57
Tamanio N = 40 Núm de Sim s = 58
Tamanio N = 40 Núm de Sim s = 59
Tamanio N = 40 Núm de Sim s = 60
Tamanio N = 40 Núm de Sim s = 61
Tamanio N = 40 Núm de Sim s = 62
Tamanio N = 40 Núm de Sim s = 63
Tamanio N = 40 Núm de Sim s = 64
Tamanio N = 40 Núm de Sim s = 65
Tamanio N = 40 Núm de Sim s = 66
Tamanio N = 40 Núm de Sim s = 67
Tamanio N = 40 Núm de Sim s = 68
Tamanio N = 40 Núm de Sim s = 69
Tamanio N = 40 Núm de Sim s = 70
Tamanio N = 40 Núm de Sim s = 71
Tamanio N = 40 Núm de Sim s = 72
Tamanio N = 40 Núm de Sim s = 73
Tamanio N = 40 Núm de Sim s = 74
Tamanio N = 40 Núm de Sim s = 75
Tamanio N = 40 Núm de Sim s = 76
Tamanio N = 40 Núm de Sim s = 77
Tamanio N = 40 Núm de Sim s = 78
Tamanio N = 40 Núm de Sim s = 79
Tamanio N = 40 Núm de Sim s = 80
Tamanio N = 40 Núm de Sim s = 81
Tamanio N = 40 Núm de Sim s = 82
Tamanio N = 40 Núm de Sim s = 83
Tamanio N = 40 Núm de Sim s = 84
Tamanio N = 40 Núm de Sim s = 85
Tamanio N = 40 Núm de Sim s = 86
Tamanio N = 40 Núm de Sim s = 87
Tamanio N = 40 Núm de Sim s = 88
Tamanio N = 40 Núm de Sim s = 89
Tamanio N = 40 Núm de Sim s = 90
Tamanio N = 40 Núm de Sim s = 91
Tamanio N = 40 Núm de Sim s = 92
Tamanio N = 40 Núm de Sim s = 93
Tamanio N = 40 Núm de Sim s = 94
Tamanio N = 40 Núm de Sim s = 95
Tamanio N = 40 Núm de Sim s = 96
Tamanio N = 40 Núm de Sim s = 97
Tamanio N = 40 Núm de Sim s = 98
Tamanio N = 40 Núm de Sim s = 99
Tamanio N = 40 Núm de Sim s = 100
Tamanio N = 50 Núm de Sim s = 1
Tamanio N = 50 Núm de Sim s = 2
Tamanio N = 50 Núm de Sim s = 3
Tamanio N = 50 Núm de Sim s = 4
Tamanio N = 50 Núm de Sim s = 5
Tamanio N = 50 Núm de Sim s = 6
Tamanio N = 50 Núm de Sim s = 7
Tamanio N = 50 Núm de Sim s = 8
Tamanio N = 50 Núm de Sim s = 9
Tamanio N = 50 Núm de Sim s = 10
Tamanio N = 50 Núm de Sim s = 11
Tamanio N = 50 Núm de Sim s = 12
Tamanio N = 50 Núm de Sim s = 13
Tamanio N = 50 Núm de Sim s = 14
Tamanio N = 50 Núm de Sim s = 15
Tamanio N = 50 Núm de Sim s = 16
Tamanio N = 50 Núm de Sim s = 17
Tamanio N = 50 Núm de Sim s = 18
Tamanio N = 50 Núm de Sim s = 19
Tamanio N = 50 Núm de Sim s = 20
Tamanio N = 50 Núm de Sim s = 21
Tamanio N = 50 Núm de Sim s = 22
Tamanio N = 50 Núm de Sim s = 23
Tamanio N = 50 Núm de Sim s = 24
Tamanio N = 50 Núm de Sim s = 25
Tamanio N = 50 Núm de Sim s = 26
Tamanio N = 50 Núm de Sim s = 27
Tamanio N = 50 Núm de Sim s = 28
Tamanio N = 50 Núm de Sim s = 29
Tamanio N = 50 Núm de Sim s = 30
Tamanio N = 50 Núm de Sim s = 31
Tamanio N = 50 Núm de Sim s = 32
Tamanio N = 50 Núm de Sim s = 33
Tamanio N = 50 Núm de Sim s = 34
Tamanio N = 50 Núm de Sim s = 35
Tamanio N = 50 Núm de Sim s = 36
Tamanio N = 50 Núm de Sim s = 37
Tamanio N = 50 Núm de Sim s = 38
Tamanio N = 50 Núm de Sim s = 39
Tamanio N = 50 Núm de Sim s = 40
Tamanio N = 50 Núm de Sim s = 41
Tamanio N = 50 Núm de Sim s = 42
Tamanio N = 50 Núm de Sim s = 43
Tamanio N = 50 Núm de Sim s = 44
Tamanio N = 50 Núm de Sim s = 45
Tamanio N = 50 Núm de Sim s = 46
Tamanio N = 50 Núm de Sim s = 47
Tamanio N = 50 Núm de Sim s = 48
Tamanio N = 50 Núm de Sim s = 49
Tamanio N = 50 Núm de Sim s = 50
Tamanio N = 50 Núm de Sim s = 51
Tamanio N = 50 Núm de Sim s = 52
Tamanio N = 50 Núm de Sim s = 53
Tamanio N = 50 Núm de Sim s = 54
Tamanio N = 50 Núm de Sim s = 55
Tamanio N = 50 Núm de Sim s = 56
Tamanio N = 50 Núm de Sim s = 57
Tamanio N = 50 Núm de Sim s = 58
Tamanio N = 50 Núm de Sim s = 59
Tamanio N = 50 Núm de Sim s = 60
Tamanio N = 50 Núm de Sim s = 61
Tamanio N = 50 Núm de Sim s = 62
Tamanio N = 50 Núm de Sim s = 63
Tamanio N = 50 Núm de Sim s = 64
Tamanio N = 50 Núm de Sim s = 65
Tamanio N = 50 Núm de Sim s = 66
Tamanio N = 50 Núm de Sim s = 67
Tamanio N = 50 Núm de Sim s = 68
Tamanio N = 50 Núm de Sim s = 69
Tamanio N = 50 Núm de Sim s = 70
Tamanio N = 50 Núm de Sim s = 71
Tamanio N = 50 Núm de Sim s = 72
Tamanio N = 50 Núm de Sim s = 73
Tamanio N = 50 Núm de Sim s = 74
Tamanio N = 50 Núm de Sim s = 75
Tamanio N = 50 Núm de Sim s = 76
Tamanio N = 50 Núm de Sim s = 77
Tamanio N = 50 Núm de Sim s = 78
Tamanio N = 50 Núm de Sim s = 79
Tamanio N = 50 Núm de Sim s = 80
Tamanio N = 50 Núm de Sim s = 81
Tamanio N = 50 Núm de Sim s = 82
Tamanio N = 50 Núm de Sim s = 83
Tamanio N = 50 Núm de Sim s = 84
Tamanio N = 50 Núm de Sim s = 85
Tamanio N = 50 Núm de Sim s = 86
Tamanio N = 50 Núm de Sim s = 87
Tamanio N = 50 Núm de Sim s = 88
Tamanio N = 50 Núm de Sim s = 89
Tamanio N = 50 Núm de Sim s = 90
Tamanio N = 50 Núm de Sim s = 91
Tamanio N = 50 Núm de Sim s = 92
Tamanio N = 50 Núm de Sim s = 93
Tamanio N = 50 Núm de Sim s = 94
Tamanio N = 50 Núm de Sim s = 95
```

```
Tamanio N = 50 Núm de Sim s = 96
Tamanio N = 50 Núm de Sim s = 97
Tamanio N = 50 Núm de Sim s = 98
Tamanio N = 50 Núm de Sim s = 99
Tamanio N = 50 Núm de Sim s = 100
Tamanio N = 80 Núm de Sim s = 1
Tamanio N = 80 Núm de Sim s = 2
Tamanio N = 80 Núm de Sim s = 3
Tamanio N = 80 Núm de Sim s = 4
Tamanio N = 80 Núm de Sim s = 5
Tamanio N = 80 Núm de Sim s = 6
Tamanio N = 80 Núm de Sim s = 7
Tamanio N = 80 Núm de Sim s = 8
Tamanio N = 80 Núm de Sim s = 9
Tamanio N = 80 Núm de Sim s = 10
Tamanio N = 80 Núm de Sim s = 11
Tamanio N = 80 Núm de Sim s = 12
Tamanio N = 80 Núm de Sim s = 13
Tamanio N = 80 Núm de Sim s = 14
Tamanio N = 80 Núm de Sim s = 15
Tamanio N = 80 Núm de Sim s = 16
Tamanio N = 80 Núm de Sim s = 17
Tamanio N = 80 Núm de Sim s = 18
Tamanio N = 80 Núm de Sim s = 19
Tamanio N = 80 Núm de Sim s = 20
Tamanio N = 80 Núm de Sim s = 21
Tamanio N = 80 Núm de Sim s = 22
Tamanio N = 80 Núm de Sim s = 23
Tamanio N = 80 Núm de Sim s = 24
Tamanio N = 80 Núm de Sim s = 25
Tamanio N = 80 Núm de Sim s = 26
Tamanio N = 80 Núm de Sim s = 27
Tamanio N = 80 Núm de Sim s = 28
Tamanio N = 80 Núm de Sim s = 29
Tamanio N = 80 Núm de Sim s = 30
Tamanio N = 80 Núm de Sim s = 31
Tamanio N = 80 Núm de Sim s = 32
Tamanio N = 80 Núm de Sim s = 33
Tamanio N = 80 Núm de Sim s = 34
Tamanio N = 80 Núm de Sim s = 35
Tamanio N = 80 Núm de Sim s = 36
Tamanio N = 80 Núm de Sim s = 37
Tamanio N = 80 Núm de Sim s = 38
Tamanio N = 80 Núm de Sim s = 39
Tamanio N = 80 Núm de Sim s = 40
Tamanio N = 80 Núm de Sim s = 41
Tamanio N = 80 Núm de Sim s = 42
Tamanio N = 80 Núm de Sim s = 43
Tamanio N = 80 Núm de Sim s = 44
Tamanio N = 80 Núm de Sim s = 45
Tamanio N = 80 Núm de Sim s = 46
Tamanio N = 80 Núm de Sim s = 47
Tamanio N = 80 Núm de Sim s = 48
Tamanio N = 80 Núm de Sim s = 49
Tamanio N = 80 Núm de Sim s = 50
Tamanio N = 80 Núm de Sim s = 51
Tamanio N = 80 Núm de Sim s = 52
Tamanio N = 80 Núm de Sim s = 53
Tamanio N = 80 Núm de Sim s = 54
Tamanio N = 80 Núm de Sim s = 55
Tamanio N = 80 Núm de Sim s = 56
Tamanio N = 80 Núm de Sim s = 57
Tamanio N = 80 Núm de Sim s = 58
Tamanio N = 80 Núm de Sim s = 59
Tamanio N = 80 Núm de Sim s = 60
Tamanio N = 80 Núm de Sim s = 61
Tamanio N = 80 Núm de Sim s = 62
Tamanio N = 80 Núm de Sim s = 63
Tamanio N = 80 Núm de Sim s = 64
Tamanio N = 80 Núm de Sim s = 65
Tamanio N = 80 Núm de Sim s = 66
Tamanio N = 80 Núm de Sim s = 67
Tamanio N = 80 Núm de Sim s = 68
Tamanio N = 80 Núm de Sim s = 69
Tamanio N = 80 Núm de Sim s = 70
Tamanio N = 80 Núm de Sim s = 71
Tamanio N = 80 Núm de Sim s = 72
Tamanio N = 80 Núm de Sim s = 73
Tamanio N = 80 Núm de Sim s = 74
Tamanio N = 80 Núm de Sim s = 75
Tamanio N = 80 Núm de Sim s = 76
Tamanio N = 80 Núm de Sim s = 77
Tamanio N = 80 Núm de Sim s = 78
Tamanio N = 80 Núm de Sim s = 79
Tamanio N = 80 Núm de Sim s = 80
Tamanio N = 80 Núm de Sim s = 81
Tamanio N = 80 Núm de Sim s = 82
Tamanio N = 80 Núm de Sim s = 83
Tamanio N = 80 Núm de Sim s = 84
Tamanio N = 80 Núm de Sim s = 85
Tamanio N = 80 Núm de Sim s = 86
Tamanio N = 80 Núm de Sim s = 87
Tamanio N = 80 Núm de Sim s = 88
Tamanio N = 80 Núm de Sim s = 89
Tamanio N = 80 Núm de Sim s = 90
Tamanio N = 80 Núm de Sim s = 91
Tamanio N = 80 Núm de Sim s = 92
Tamanio N = 80 Núm de Sim s = 93
Tamanio N = 80 Núm de Sim s = 94
Tamanio N = 80 Núm de Sim s = 95
Tamanio N = 80 Núm de Sim s = 96
Tamanio N = 80 Núm de Sim s = 97
Tamanio N = 80 Núm de Sim s = 98
Tamanio N = 80 Núm de Sim s = 99
Tamanio N = 80 Núm de Sim s = 100
Tamanio N = 100 Núm de Sim s = 1
Tamanio N = 100 Núm de Sim s = 2
Tamanio N = 100 Núm de Sim s = 3
Tamanio N = 100 Núm de Sim s = 4
Tamanio N = 100 Núm de Sim s = 5
Tamanio N = 100 Núm de Sim s = 6
Tamanio N = 100 Núm de Sim s = 7
Tamanio N = 100 Núm de Sim s = 8
Tamanio N = 100 Núm de Sim s = 9
Tamanio N = 100 Núm de Sim s = 10
Tamanio N = 100 Núm de Sim s = 11
Tamanio N = 100 Núm de Sim s = 12
Tamanio N = 100 Núm de Sim s = 13
Tamanio N = 100 Núm de Sim s = 14
Tamanio N = 100 Núm de Sim s = 15
Tamanio N = 100 Núm de Sim s = 16
Tamanio N = 100 Núm de Sim s = 17
Tamanio N = 100 Núm de Sim s = 18
Tamanio N = 100 Núm de Sim s = 19
Tamanio N = 100 Núm de Sim s = 20
Tamanio N = 100 Núm de Sim s = 21
Tamanio N = 100 Núm de Sim s = 22
Tamanio N = 100 Núm de Sim s = 23
Tamanio N = 100 Núm de Sim s = 24
Tamanio N = 100 Núm de Sim s = 25
Tamanio N = 100 Núm de Sim s = 26
Tamanio N = 100 Núm de Sim s = 27
Tamanio N = 100 Núm de Sim s = 28
Tamanio N = 100 Núm de Sim s = 29
Tamanio N = 100 Núm de Sim s = 30
Tamanio N = 100 Núm de Sim s = 31
Tamanio N = 100 Núm de Sim s = 32
Tamanio N = 100 Núm de Sim s = 33
Tamanio N = 100 Núm de Sim s = 34
Tamanio N = 100 Núm de Sim s = 35
Tamanio N = 100 Núm de Sim s = 36
Tamanio N = 100 Núm de Sim s = 37
Tamanio N = 100 Núm de Sim s = 38
Tamanio N = 100 Núm de Sim s = 39
Tamanio N = 100 Núm de Sim s = 40
Tamanio N = 100 Núm de Sim s = 41
Tamanio N = 100 Núm de Sim s = 42
Tamanio N = 100 Núm de Sim s = 43
Tamanio N = 100 Núm de Sim s = 44
Tamanio N = 100 Núm de Sim s = 45
Tamanio N = 100 Núm de Sim s = 46
Tamanio N = 100 Núm de Sim s = 47
Tamanio N = 100 Núm de Sim s = 48
Tamanio N = 100 Núm de Sim s = 49
Tamanio N = 100 Núm de Sim s = 50
Tamanio N = 100 Núm de Sim s = 51
Tamanio N = 100 Núm de Sim s = 52
Tamanio N = 100 Núm de Sim s = 53
Tamanio N = 100 Núm de Sim s = 54
Tamanio N = 100 Núm de Sim s = 55
Tamanio N = 100 Núm de Sim s = 56
Tamanio N = 100 Núm de Sim s = 57
Tamanio N = 100 Núm de Sim s = 58
Tamanio N = 100 Núm de Sim s = 59
Tamanio N = 100 Núm de Sim s = 60
Tamanio N = 100 Núm de Sim s = 61
Tamanio N = 100 Núm de Sim s = 62
Tamanio N = 100 Núm de Sim s = 63
```

```
Tamanio N = 100 Núm de Sim s = 64
Tamanio N = 100 Núm de Sim s = 65
Tamanio N = 100 Núm de Sim s = 66
Tamanio N = 100 Núm de Sim s = 67
Tamanio N = 100 Núm de Sim s = 68
Tamanio N = 100 Núm de Sim s = 69
Tamanio N = 100 Núm de Sim s = 70
Tamanio N = 100 Núm de Sim s = 71
Tamanio N = 100 Núm de Sim s = 72
Tamanio N = 100 Núm de Sim s = 73
Tamanio N = 100 Núm de Sim s = 74
Tamanio N = 100 Núm de Sim s = 75
Tamanio N = 100 Núm de Sim s = 76
Tamanio N = 100 Núm de Sim s = 77
Tamanio N = 100 Núm de Sim s = 78
Tamanio N = 100 Núm de Sim s = 79
Tamanio N = 100 Núm de Sim s = 80
Tamanio N = 100 Núm de Sim s = 81
Tamanio N = 100 Núm de Sim s = 82
Tamanio N = 100 Núm de Sim s = 83
Tamanio N = 100 Núm de Sim s = 84
Tamanio N = 100 Núm de Sim s = 85
Tamanio N = 100 Núm de Sim s = 86
Tamanio N = 100 Núm de Sim s = 87
Tamanio N = 100 Núm de Sim s = 88
Tamanio N = 100 Núm de Sim s = 89
Tamanio N = 100 Núm de Sim s = 90
Tamanio N = 100 Núm de Sim s = 91
Tamanio N = 100 Núm de Sim s = 92
Tamanio N = 100 Núm de Sim s = 93
Tamanio N = 100 Núm de Sim s = 94
Tamanio N = 100 Núm de Sim s = 95
Tamanio N = 100 Núm de Sim s = 96
Tamanio N = 100 Núm de Sim s = 97
Tamanio N = 100 Núm de Sim s = 98
Tamanio N = 100 Núm de Sim s = 99
Tamanio N = 100 Núm de Sim s = 100
```

## Exportando los datos

In [29]:
```
CSV.write("DatosProy2.csv",Tables.table(Datos),writeheader = false)
```

```
┌ Warning: `writeheader=false` is deprecated in favor of `header=false`
│   caller = ip:0x0
└ @ Core :-1
```

Out[29]: "DatosProy2.csv"

Tabulado

| Nombre | N | w | Freq | media |
|---|---|---|---|---|
| GCJul | 20 | 0 | 300 | 9,22606033333333e-05 |
| GCJul | 40 | 0 | 300 | 0,000431778556666667 |
| GCJul | 50 | 0 | 300 | 0,000808637813333333 |
| GCJul | 80 | 0 | 300 | 0,00234513241 |
| GCJul | 100 | 0 | 300 | 0,01749389442 |
| GCPy | 20 | 0 | 300 | 0,00053680772 |
| GCPy | 40 | 0 | 300 | 0,00128655791 |
| GCPy | 50 | 0 | 300 | 0,00137983773333333 |
| GCPy | 80 | 0 | 300 | 0,00203670412 |
| GCPy | 100 | 0 | 300 | 0,00128548714 |
| GSJul | 20 | 0 | 300 | 0,000260962096666667 |
| GSJul | 40 | 0 | 300 | 0,00601398226666667 |
| GSJul | 50 | 0 | 300 | 0,00750617396333333 |
| GSJul | 80 | 0 | 300 | 0,0148410223633333 |
| GSJul | 100 | 0 | 300 | 0,01960627934 |
| GSPy | 20 | 0 | 300 | 0,00186896014333333 |
| GSPy | 40 | 0 | 300 | 0,00729056066666667 |
| GSPy | 50 | 0 | 300 | 0,00961538102333333 |
| GSPy | 80 | 0 | 300 | 0,0262778092633333 |
| GSPy | 100 | 0 | 300 | 0,0388188607966667 |
| JacJul | 20 | 0 | 300 | 0,000281577266666667 |
| JacJul | 40 | 0 | 300 | 0,00456452801 |
| JacJul | 50 | 0 | 300 | 0,00535245062666667 |
| JacJul | 80 | 0 | 300 | 0,00689606937666667 |
| JacJul | 100 | 0 | 300 | 0,01446811983 |
| JacPy | 20 | 0 | 300 | 0,00167588550333333 |
| JacPy | 40 | 0 | 300 | 0,00640967394333333 |
| JacPy | 50 | 0 | 300 | 0,00445353861333333 |
| JacPy | 80 | 0 | 300 | 0,00753485980333333 |
| JacPy | 100 | 0 | 300 | 0,00704131403666667 |
| SRSJul | 20 | 1 | 300 | 0,00020518001 |
| SRSJul | 40 | 1 | 300 | 0,00383789634333333 |
| SRSJul | 50 | 1 | 300 | 0,00437237013333333 |
| SRSJul | 80 | 1 | 300 | 0,00743250781666667 |
| SRSJul | 100 | 1 | 300 | 0,0153907449033333 |
| SRSJul | 20 | 1.2 | 300 | 0,00016993276 |
| SRSJul | 40 | 1.2 | 300 | 0,00256953991 |
| SRSJul | 50 | 1.2 | 300 | 0,00278189531333333 |
| SRSJul | 80 | 1.2 | 300 | 0,00611042574666667 |
| SRSJul | 100 | 1.2 | 300 | 0,0115132217766667 |
| SRSJul | 20 | 1.4 | 300 | 0,000183953226666667 |
| SRSJul | 40 | 1.4 | 300 | 0,00197200166 |
| SRSJul | 50 | 1.4 | 300 | 0,00213549355 |
| SRSJul | 80 | 1.4 | 300 | 0,00501545947 |
| SRSJul | 100 | 1.4 | 300 | 0,00995183219 |
| SRSJul | 20 | 1.6 | 300 | 0,000283491763333333 |
| SRSJul | 40 | 1.6 | 300 | 0,00211698022666667 |
| SRSJul | 50 | 1.6 | 300 | 0,00277673954 |

| | | | | |
|---|---|---|---|---|
| SRSJul | 80 | 1.6 | 300 | 0,00597113000666667 |
| SRSJul | 100 | 1.6 | 300 | 0,01644522381 |
| SRSJul | 20 | 1.8 | 300 | 0,00059829406 |
| SRSJul | 40 | 1.8 | 300 | 0,00350849693 |
| SRSJul | 50 | 1.8 | 300 | 0,00376747304333333 |
| SRSJul | 80 | 1.8 | 300 | 0,00952306247 |
| SRSJul | 100 | 1.8 | 300 | 0,0285359874133333 |
| SRSJul | 20 | 2 | 300 | 0,000619826326666667 |
| SRSJul | 40 | 2 | 300 | 0,00418149639333333 |
| SRSJul | 50 | 2 | 300 | 0,00480777538 |
| SRSJul | 80 | 2 | 300 | 0,00991465015 |
| SRSJul | 100 | 2 | 300 | 0,0301907120833333 |
| SRSPy | 20 | 1 | 300 | 0,00119027406 |
| SRSPy | 40 | 1 | 300 | 0,00280286734333333 |
| SRSPy | 50 | 1 | 300 | 0,00341490623 |
| SRSPy | 80 | 1 | 300 | 0,0058141328 |
| SRSPy | 100 | 1 | 300 | 0,00553670714333333 |
| SRSPy | 20 | 1.2 | 300 | 0,000873064306666667 |
| SRSPy | 40 | 1.2 | 300 | 0,00217876849 |
| SRSPy | 50 | 1.2 | 300 | 0,00223823256666667 |
| SRSPy | 80 | 1.2 | 300 | 0,00377460367666667 |
| SRSPy | 100 | 1.2 | 300 | 0,00406908695333333 |
| SRSPy | 20 | 1.4 | 300 | 0,000967647826666667 |
| SRSPy | 40 | 1.4 | 300 | 0,00208828312 |
| SRSPy | 50 | 1.4 | 300 | 0,00222538768666667 |
| SRSPy | 80 | 1.4 | 300 | 0,00365580196666667 |
| SRSPy | 100 | 1.4 | 300 | 0,00431145816 |
| SRSPy | 20 | 1.6 | 300 | 0,00144421756333333 |
| SRSPy | 40 | 1.6 | 300 | 0,00276943333 |
| SRSPy | 50 | 1.6 | 300 | 0,00315693455 |
| SRSPy | 80 | 1.6 | 300 | 0,00522035894666667 |
| SRSPy | 100 | 1.6 | 300 | 0,00511308587666667 |
| SRSPy | 20 | 1.8 | 300 | 0,00267412998 |
| SRSPy | 40 | 1.8 | 300 | 0,00534601931666667 |
| SRSPy | 50 | 1.8 | 300 | 0,00566975328 |
| SRSPy | 80 | 1.8 | 300 | 0,0107731143566667 |
| SRSPy | 100 | 1.8 | 300 | 0,00821597288666667 |
| SRSPy | 20 | 2 | 300 | 0,00264281893 |
| SRSPy | 40 | 2 | 300 | 0,00529279843 |
| SRSPy | 50 | 2 | 300 | 0,00621833765333333 |
| SRSPy | 80 | 2 | 300 | 0,01148236515 |
| SRSPy | 100 | 2 | 300 | 0,00858874309666667 |