PRÁCTICA 6:
Sistemas de Ecuaciones Lineales. Métodos Iterativos

6.1 Resumen de Teoría

6.1.1 Introducción

Para finalizar con los sistemas de ecuaciones, en esta práctica, estudiaremos brevemente los problemas de condicionamiento de sistemas lineales, para, a continuación, estudiar la resolución de sistemas lineales utilizando métodos iterativos.

Ahora centraremos nuestro estudio en aquellos que resultan ser una generalización del proceso de iteración del punto fijo, como son el método de Jacobi y el de Gauss-Seidel. Debemos resaltar que lo métodos directos vistos en la práctica anterior son muy caros computacionalmente. Estos métodos exigen una memoria de máquina proporcional al cuadrado del orden de la matriz de coeficiente *A*. De igual manera se producen grandes errores de redondeo como consecuencia del número de operaciones.

En los métodos iterativos necesitamos tener una aproximación inicial de la solución y no esperamos tener una solución exacta aun cuando todas las operaciones se realicen utilizando una aritmética exacta. Pero podemos decir que en muchos casos son más efectivos que los métodos directos por requerir mucho menos esfuerzo computacional. Además sus errores se reducen cuando la matriz es dispersa es decir cuando la matriz tiene un alto porcentaje de elementos nulos. Estos métodos han permitido resolver sistemas de hasta miles ecuaciones.

6.1.2 Condicionamiento de Sistemas Lineales

Una *norma* es una función que toma valores reales y que proporciona una medida del tamaño o "longitud" de entidades matemáticas multi-componentes, como los vectores y las matrices. La norma de una matriz cuadrada, *A*, puede ser definida en forma consistente con la definición de norma de un vector. A las normas matriciales más habituales se las designa del mismo modo que a las normas vectoriales con las que son compatibles. Ejemplos de normas matriciales:

- $\|A\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|$. *Norma columna-suma:* sumatorio de los valores absolutos de los coeficientes para cada columna, y el mayor de estos sumatorios se toma como la norma. Es compatible con la norma vectorial $\|x\|_1$

- $\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$. *Norma fila-suma:* sumatorio de los valores absolutos de los coeficientes para cada fila, y el mayor de estos sumatorios se toma como la norma. Es compatible con la norma vectorial $\|x\|_\infty$

- $\|A\|_2 = \lambda_{max}^{1/2}$. *Norma espectral*, donde λ_{max} es el mayor autovalor de A^tA. Es compatible con la norma vectorial $\|x\|_2$

Es importante destacar que aunque la norma $\|A\|_2$, es la norma mínima y, por lo tanto, proporciona la medida de tamaño más ajustada, la elección, algunas veces, está influenciada por consideraciones prácticas. Por ejemplo, la *norma uniforme o fila-suma* $\|A\|_\infty$ es ampliamente usada por la facilidad con que se calcula, y por el hecho de que usualmente proporciona una medida adecuada del tamaño de la matriz.

Recordemos que un número λ es un *autovalor* (o valor propio) de una matriz cuadrada de orden *n*, *A*, si existe un vector *x* no nulo de modo que $Ax = \lambda x$ y que a cada vector en esas condiciones se le denomina vector propio o autovector de A asociado al autovalor λ.

Es un resultado conocido de Álgebra Lineal que los valores propios de una matriz *A* coinciden con las raíces del polinomio característico de *A* definido como $|A- \lambda I|$, donde se representa por *I* a la matriz identidad de orden *n*.

También, recordando que se define el *radio espectral* de *A*, y se representa por $\rho(A)$, como el máximo de los módulos de los autovalores de *A*, podemos decir que:

$$\|A\|_2 = \rho(A^t A)^{1/2}$$

Ahora que se ha presentado el concepto de norma, se puede usar para definir:

$$cond(A) = \big||A|\big|.||A^{-1}||$$

donde *cond*(*A*) se llama *número de condición de una matriz.*

Este número permite saber si la solución de un sistema $Ax = b$ se verá afectada de forma importante, al realizar una pequeña modificación en los coeficientes de *A* o *b*. Esto es importante, por ejemplo, cuando los datos provienen de medidas experimentales, por lo que pueden contener errores. Dicho número de condición es siempre mayor o igual que 1.

- Los *sistemas bien condicionados* (*cond*(*A*) cercano a 1) son aquellos en los que un pequeño cambio en los coeficientes provoca un cambio pequeño en la solución.

- Los *sistemas mal condicionados* (*cond*(*A*) $\gg$ 1) son aquellos en donde pequeños cambios en los coeficientes generan grandes cambios en la solución.

6.1.3 Métodos Iterativos para Sistemas Lineales

La aplicación de los *métodos de iteración del punto fij*o a la resolución de ecuaciones ya fue tratada en la Práctica 4 para el caso de una variable. En el caso de un sistema lineal de ecuaciones $Ax = b$, debemos resolver una ecuación vectorial $F(x) = 0$, con *x* un vector de dimensión *n* y donde:

$$F(x) = Ax - b = 0$$

Esta ecuación se transforma en una ecuación equivalente de punto fijo:

$$G(x) = x \quad \text{con} \quad G(x) = Bx + c$$

donde *G* es una función de iteración, *B* es una matriz cuadrada de orden *n* y *c* un vector de dimensión *n*. Obviamente, la elección de *B* y *c* debe realizarse de modo que la solución de $G(x)=x$ coincida con la de $F(x)=0$.

Llamaremos $x^{(m)} = \left[x_1^{(m)}, x_2^{(m)}, ..., x_n^{(m)}\right]^t$ y como en todo proceso de iteración de punto fijo:

1. Proponemos un vector inicial $x^{(0)}$ como la primera aproximación al vector solución *x.* Una forma simple para obtener los valores iniciales es suponer que todos son cero.

2. Calculamos la sucesión de vectores $x^{(1)}, x^{(2)}, x^{(3)}, ...$, que son soluciones aproximadas, usando:

$$x^{(m+1)} = G(x^{(m)}) = Bx^{(m)} + C \ , \quad m = 0,1,2,.....$$

A continuación veamos un teorema que proporciona una condición necesaria y suficiente de convergencia en términos del radio espectral de la matriz de iteración *B.* Y recordemos que el concepto de convergencia de un proceso de iteración del punto fijo se asocia siempre a que dicha convergencia no dependa del vector inicial elegido.

Teorema. Convergencia del Método del Punto Fijo
Sea B una matriz cuadrada de orden n, c un vector de dimensión n, G la función vectorial que, a cada x, le hace corresponde:

$$G(x) = Bx + c$$

y sea ρ(B) el radio espectral de B. Entonces, la condición ρ(B) < 1 es necesaria y suficiente para que la función G(x) posea un único punto fijo, es decir, exista un único x, tal que G(x) = x. Además, si fijamos un vector cualquiera $x^{(0)}$ *y definimos la sucesión:*

$$x^{(1)} = G(x^{(0)}), x^{(2)} = G(x^{(1)}), ..., x^{(k+1)} = G(x^{(k)}), ...$$

dicha sucesión converge al punto fijo de G.

Para determinar la sucesión del proceso iterativo de punto fijo veremos dos formas:

1. *Método de Jacobi* (Desplazamiento simultáneo)

2. *Método de Gauss –Seidel* (Desplazamiento sucesivo)

Método de Jacobi

Si todos los elementos de la diagonal principal de *A* son no nulos, entonces podemos despejar x_1 de la primera ecuación, x_2 de la segunda, etc. Si $x^{(m)}$ es el vector de aproximación a la solución *x* después de *m* iteraciones, entonces, tendremos la siguiente aproximación:

$$x^{(m+1)} = \begin{pmatrix} x_1^{(m+1)} \\ x_2^{(m+1)} \\ x_3^{(m+1)} \\ \dots \end{pmatrix} = \begin{pmatrix} \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(m)} - a_{13}x_3^{(m)} - \dots) \\ \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(m)} - a_{23}x_3^{(m)} - \dots) \\ \frac{1}{a_{33}}(b_3 - a_{31}x_1^{(m)} - a_{32}x_2^{(m)} - \dots) \\ \dots \end{pmatrix}$$

$$x^{(m+1)} = \begin{pmatrix} 0 & \frac{-a_{12}}{a_{11}} & \frac{-a_{13}}{a_{11}} & \dots \\ \frac{-a_{21}}{a_{22}} & 0 & \frac{-a_{23}}{a_{22}} & \dots \\ \frac{-a_{31}}{a_{33}} & \frac{-a_{31}}{a_{33}} & 0 & \dots \\ \dots & & & \end{pmatrix} \begin{pmatrix} x_1^{(m)} \\ x_2^{(m)} \\ x_3^{(m)} \\ \dots \end{pmatrix} + \begin{pmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \frac{b_3}{a_{33}} \\ \dots \end{pmatrix}$$

El conjunto anterior de ecuaciones puede escribirse abreviadamente como:

$$x^{(m+1)} = G(x^{(m)}) = Bx^{(m)} + C \ , \ \ m = 0,1,2,.....$$

$$x_i^{(m+1)} = -\frac{1}{a_{ii}}\left(-b_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(m)} \right), \ i = 1,2,.....,n$$

Método de Gauss-Seidel

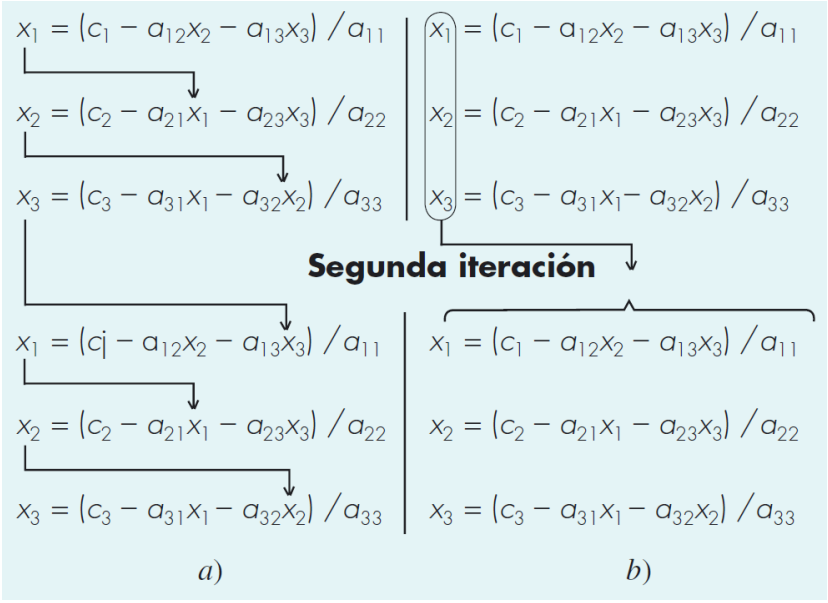
Este método se diferencia del anterior en que los valores que se van calculando en la iteración (m+1) se usan para calcular los valores restantes de esa misma iteración. Esto es:

$$x^{(m+1)} = \begin{pmatrix} x_1^{(m+1)} \\ x_2^{(m+1)} \\ x_3^{(m+1)} \\ \dots \end{pmatrix} = \begin{pmatrix} \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(m)} - a_{13}x_3^{(m)} - \dots) \\ \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(m+1)} - a_{23}x_3^{(m)} - \dots) \\ \frac{1}{a_{33}}(b_3 - a_{31}x_1^{(m+1)} - a_{32}x_2^{(m+1)} - \dots) \\ \dots \end{pmatrix}$$

En este caso tenemos:

$$x_i^{(m+1)} = -\frac{1}{a_{ii}}\left(-b_i + \sum_{\substack{j=i+1 \\ i \neq j}}^{n} a_{ij} x^{(m)} + \sum_{\substack{j=1 \\ j \neq i}}^{i-1} a_{ij} x^{(m+1)} \right), 1 \leq i \leq n$$

La siguiente figura muestra la diferencia entre los métodos de *a*) Gauss-Seidel y *b*) Jacobi.



Nota. La condición de convergencia que proporciona el Teorema anterior tiene utilidad práctica en el caso en que se puedan calcular o estimar los autovalores de la matriz lo cual no siempre es simple.

Por eso suele también utilizarse otra condición, según la cual, los procesos de Jacobi y Gauss–Seidel convergerán si en la matriz de coeficientes *A* cada elemento de la diagonal principal es mayor que la suma de los valores absolutos de todos los demás elementos de la misma fila o columna (*matriz estrictamente diagonal dominante*):

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| \quad 1 \le i \le n;$$

$$|a_{ii}| > \sum_{\substack{i=1 \\ i\neq j}} a_{ij}\,; \qquad 1 \le j \le n$$

Este criterio es suficiente pero no necesario para la convergencia. Como condición de parada, como siempre, utilizaremos:

$$\left|\left| \frac{x^{(n+1)} - x^{(n)}}{x^{(n)}} \right|\right| \le Tol$$

con la norma vectorial:

$$||x|| = \sqrt{x.x} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

6.2 Normas matriciales. Número de condición de una matriz

Vamos a calcular ahora las normas usuales de vectores y matrices con funciones de Maxima. Revisemos, en primer lugar, los comandos que incorpora Maxima para calcular el número de condición de una matriz y las normas matriciales. Tenemos:

- **mat_norm(M,p)** = devuelve la p-norma de la matriz *M*. Los valores permitidos son: **mat_norm(M,1), mat_norm(M,inf)**
- **mat_cond(M,p)** = devuelve el número de condición de la matriz M usando la p-norma. Los valores permitidos son: **mat_cond(M,1)** y **mat_cond(M,inf)**

```
(%i1)    kill(all)$
```

Si consideramos ahora una matriz cuadrada:

```
(%i1)    A:matrix([1,-1],[2,3]);
         [ 1  -1 ]
(A)      [       ]
         [ 2   3 ]
(%i2)    mat_norm(A,1);
0 errors, 0 warnings
  (%o2)    4
(%i3)    mat_norm(A,inf);
(%o3)    5
```

Una vez tenemos controlado como calcular normas de matrices, vamos a pasar a calcular el número de condición de una matriz. Este número indica si la matriz está bien condicionada o no con respecto a la resolución de sistemas lineales. Un número de condición alto es malo, ya que indica posibles problemas en la resolución numérica de un sistema lineal que tenga a esa matriz como matriz de coeficientes. Los condicionamientos respecto de las normas usuales, están dados por las funciones siguientes:

```
(%i4)    A:matrix([2,2,4],[2,3,6],[1,1,1]);
         [ 2  2  4 ]
         [         ]
(A)      [ 2  3  6 ]
         [         ]
         [ 1  1  1 ]
(%i5)    mat_cond(A,1);
(%o5)    44
(%i6)    mat_cond(A,inf);
(%o6)    55
```

Un ejemplo clásico de matrices mal condicionadas son las matrices de Vandermonde. Recordemos que una Matriz de Vandermonde es una matriz que presenta una progresión geométrica en cada fila. Estas matrices aparecen de manera natural en el problema de interpolación de Lagrange. El número de condición de estas matrices crece con la dimensión, y eso es perjudicial.

```
(%i7)    B:vandermonde_matrix([1,2,3,4,5]);
                [ 1  1   1    1    1  ]
                [                     ]
                [ 1  2   4    8   16  ]
                [                     ]
(B)             [ 1  3   9   27   81  ]
                [                     ]
                [ 1  4  16   64  256  ]
                [                     ]
                [ 1  5  25  125  625  ]
(%i8)    mat_cond(B,1);
(%o8)    44055
```

Vamos a resolver dos sistemas lineales con la anterior como matriz de coeficientes y con pequeños cambios en la columna de términos independientes, para ver cómo afecta a las soluciones.

```
(%i9)    linsolve([x+y+z+u+v=10,x+2*y+4*z+8*u+16*v=2,
         x+3*y+9*z+27*u+81*v=3,x+4*y+16*z+64*u+256*v=4,
         x+5*y+25*z+125*u+625*v=50],[x,y,z,u,v]),numer;
(%o9)    [x=90,y=-150.5,z=92.25,u=-24,v=2.25]
```

Veamos ahora las soluciones del sistema perturbado, cambiando ligeramente los términos independientes de las ecuaciones anteriores.

```
(%i10)   ratprint:false$
(%i8)    linsolve([x+y+z+u+v=10.02,x+2*y+4*z+8*u+16*v=2.01,
         x+3*y+9*z+27*u+81*v=2.9,x+4*y+16*z+64*u+256*v=3.99,
         x+5*y+25*z+125*u+625*v=50.03],[x,y,z,u,v]),numer;
(%o8)    [x=89.08,y=-148.6641666666667,z=91.09791666666666,
         u=-23.72083333333333,v=2.227083333333333]
```

Este efecto puede ser mucho más evidente. Consideremos los dos sistemas lineales siguientes:

$$\left.\begin{aligned} 0.1^{12}x + 0.1^{13}y &= 6.06\,10^{-13} \\ 0.9^{12}x + 0.9^{13}y &= 0.03577 \end{aligned}\right\}$$

$$\left.\begin{aligned} 0.1^{12}x + 0.1^{13}y &= 0.0001 \\ 0.9^{12}x + 0.9^{13}y &= 0.0356 \end{aligned}\right\}$$

La solución del primero es:

```
(%i9)    linsolve([0.1^12*x+0.1^13*y=6.06*10^-13,
                   0.9^12*x+0.9^13*y=0.03577],[x,y]),numer;
(%o9)    [x=0.6659186175755177,y=-0.599186175755177]
```

Veamos ahora las soluciones del segundo sistema, donde cambian ligeramente los términos independientes.

```
(%i10)  linsolve([0.1^12*x+0.1^13*y=0.0001,
                  0.9^12*x+0.9^13*y=0.0356],[x,y]),numer;
(%o10)  [x=1.124999999842439 10^8,y=-1.249999998424386 10^8]
```

Para ver el porqué de esta gran diferencia en las soluciones, por tan leve cambio en los términos independientes de estas ecuaciones, basta con ver el número de condición en norma 1 de la matriz de coeficientes de este sistema.

```
(%i12)  coefmatrix([0.1^12*x+0.1^13*y=6.06*10^-13,
                    0.9^12*x+0.9^13*y=0.03577],[x,y]),numer;
(%o12)  [     1.0 10^-12              1.0 10^-13          ]
        [                                                 ]
        [0.2824295364810007  0.2541865828329006           ]
```

Y su número de condición en norma 1 es:

```
(%i13)  mat_cond(%,1);
(%o13)  6.707701491447516 10^11
```

6.3 Método de Jacobi

Dentro de los métodos iterativos para la resolución de sistemas de ecuaciones lineales, el más clásico es el método de Jacobi. Consideremos el sistema $Ax = b$, siendo A una matriz cuadrada de orden n, descompuesta en la forma $A = D + L + U$, donde D es una matriz diagonal, L triangular inferior, con diagonal principal nula, y U triangular superior, también con diagonal principal nula.

Suponemos que los elementos de la diagonal de A son todos no nulos, con lo que D es regular, y definimos la matriz de iteración:

$$G(x) = Bx + c; \text{ con } B = -D^{-1}(L + U); c = D^{-1}b$$

Como vimos, esto equivale a despejar x_1 de la primera ecuación, x_2 de la segunda, etc...

El método es convergente si y sólo si $\rho(B) < 1$. En particular, si A es estrictamente diagonal dominante el método es convergente. Vamos a ver cómo aplicar dicho método en el supuesto de que sea convergente. Consideramos la matriz de coeficientes:

```
(%i1)    kill(all)$
(%i1)    A:matrix([1,6,-1,2],[4,1,0,1],[0,-1,10,4],[3,-1,-2,7]);
```

$$(A) \quad \begin{bmatrix} 1 & 6 & -1 & 2 \\ 4 & 1 & 0 & 1 \\ 0 & -1 & 10 & 4 \\ 3 & -1 & -2 & 7 \end{bmatrix}$$

Y los términos independientes lo introducimos ahora como una matriz columna:

```
(%i2)    b:matrix([6],[6],[-2],[19]);
```

$$(b) \quad \begin{bmatrix} 6 \\ 6 \\ -2 \\ 19 \end{bmatrix}$$

Debido a que es mejor trabajar con una matriz estrictamente diagonal dominante, ya que esto es condición suficiente para la convergencia de este método, vamos a intercambiar la primera con la segunda ecuación (que se traduce en el intercambio de esas filas en las matrices A y b)

```
(%i3)    A:rowswap(A,1,2);
0 errors, 0 warnings
```

$$(A) \quad \begin{bmatrix} 4 & 1 & 0 & 1 \\ 1 & 6 & -1 & 2 \\ 0 & -1 & 10 & 4 \\ 3 & -1 & -2 & 7 \end{bmatrix}$$

```
(%i4)   b:rowswap(b,1,2);
```

$$(b)\quad \begin{bmatrix} 6 \\ 6 \\ -2 \\ 19 \end{bmatrix}$$

Antes de construir un bloque para el método de Jacobi veamos como programar este ejemplo. Para empezar inicializamos las variables, damos el número máximo de iteraciones, hallamos la dimensión del sistema y elegimos los valores iniciales de las incógnitas, que tomamos iguales a cero.

```
(%i5)   maxiter:100;
(maxiter) 100
(%i7)   n:matrix_size(b);
        n:n[1];
(n)     [4,1]
  (n)     4
(%i8)   xant:matrix([0],[0],[0],[0])$
(%i9)   x:matrix([0],[0],[0],[0])$
(%i10)  k:1;
(k)     1
(%i11)  while (k <= maxiter) do
        (
        for i:1 thru n do(
        x[i]:1/A[i,i]*(b[i]-sum(A[i,j]*xant[j],j,1,i-1)
        -sum(A[i,j]*xant[j],j,i+1,n))
        ),
        xant:ev(x),
        k:k+1)$
(%i12)  print("La solución aproximada tras ", maxiter, " pasos es ",
            float(x))$
```

$$\textit{La solución aproximada tras}\ \ 100\ \ \textit{pasos es}\quad \begin{bmatrix} 1.0 \\ 2.071261776039495\,10^{-43} \\ -1.0 \\ 2.0 \end{bmatrix}$$

Como vemos la última aproximación obtenida de la solución es una buena aproximación de la solución exacta dada por $x = (1, 0, -1, 2)$.

```
(%i13)  linsolve([x1+6*x2-x3+2*x4=6,4*x1+x2+x4=6,
            -x2+10*x3+4*x4=-2,3*x1-x2-2*x3+7*x4=19],[x1,x2,x3,x4]);
(%o13)  [x1=1,x2=0,x3=-1,x4=2]
```

Hemos llamado *x* al vector de incógnitas y usamos una variable auxiliar llamada *xant* para almacenar los valores de *x* en la iteración anterior. Obsérvese que al finalizar cada iteración (bucle *k*) los valores de *x* los pasamos a *xant* para ser tomados en la iteración siguiente. El comando **ev** debe usarse para que *xant* sea la evaluación en ese momento de *x*, y no debe usarse **xant:x**. El bucle *i* recorre las filas despejando la *x* de esa fila, mientras que el sumatorio de *j* evita el valor de *i*.

Veamos ahora un bloque para el método de Jacobi que incorpora el criterio de salida basado en la tolerancia relativa entre dos iteraciones consecutivas. En este caso debemos hallar las normas de los vectores *x* de incógnitas. El resto es muy similar a los ya vistos en este curso.

```
(%i1)    kill(all)$
```

```
(%i1)    /* Función jacobi(A,b,tolr,maxiter) */
         /* Resuelve un sistema de n ecuaciones lineales Ax=b,
            mediante el método de Jacobi */
         /* ARGUMENTOS DE ENTRADA: */
         /* A ..... Matriz de coeficientes */
         /* b ..... Matriz columna de términos independientes */
         /* tolr .. Tolerancia relativa entre dos iteraciones */
         /* maxiter .Número máximo de iteraciones */
         /* ARGUMENTOS DE SALIDA: */
         /* x ... Vector de incógnitas x[n] */
         /* N ... Número de Iteraciones */;
         jacobi(A,b,tolr,maxiter):=block([numer],numer:true,
         n:matrix_size(b)[1],
         x:zeromatrix(n,1),
         xant:zeromatrix(n,1),
             for k:1 thru maxiter do
             (
                 for i:1 thru n do(
                 x[i]:1/A[i,i]*(b[i]-sum(A[i,j]*xant[j],j,1,i-1)
                 -sum(A[i,j]*xant[j],j,i+1,n))
                     ),
             if abs((mat_norm(x,1)-mat_norm(xant,1))/mat_norm(x,1))
                  < tolr then
             return((print("La aproximación es =",x,
                           "en", k, "iteraciones",""))),
             xant:ev(x),
             if k=maxiter then
             print("No lograda la aproximación deseada en ",
             maxiter," iteraciones")
             )
         )$
```

```
(%i2)    A:matrix([1,6,-1,2],[4,1,0,1],[0,-1,10,4],[3,-1,-2,7]);
```

(A)
$$\begin{bmatrix} 1 & 6 & -1 & 2 \\ 4 & 1 & 0 & 1 \\ 0 & -1 & 10 & 4 \\ 3 & -1 & -2 & 7 \end{bmatrix}$$

```
(%i3)    b:matrix([6],[6],[-2],[19]);
```

$$(b)\quad \begin{bmatrix} 6 \\ 6 \\ -2 \\ 19 \end{bmatrix}$$

```
(%i4)    A:rowswap(A,1,2);
0 errors, 0 warnings
```

$$(A)\quad \begin{bmatrix} 4 & 1 & 0 & 1 \\ 1 & 6 & -1 & 2 \\ 0 & -1 & 10 & 4 \\ 3 & -1 & -2 & 7 \end{bmatrix}$$

```
(%i5)    b:rowswap(b,1,2);
```

$$(b)\quad \begin{bmatrix} 6 \\ 6 \\ -2 \\ 19 \end{bmatrix}$$

```
(%i6)    jacobi(A,b,10^-10,200);
```

$$La\ aproximación\ es = \begin{bmatrix} 1.00000000003545 \\ 3.588507269114416\,10^{-11} \\ -0.9999999999490641 \\ 1.999999999974545 \end{bmatrix} \quad en\ 25\ iteraciones$$

```
  (%o6)
```

Es importante ver ahora que ocurre si no permutamos las filas, dejando la matriz *A* original, que no es de diagonal dominante.

```
(%i7)    A:matrix([1,6,-1,2],[4,1,0,1],[0,-1,10,4],[3,-1,-2,7])$

(%i8)    b:matrix([6],[6],[-2],[19])$

(%i9)    jacobi(A,b,10^-10,200);
No lograda la aproximación deseada en  200  iteraciones
  (%o9)    done
```

El método no es convergente.

6.4 Método de Gauss-Seidel

Veamos seguidamente otro de los métodos iterativos usuales para la resolución de sistemas de ecuaciones lineales, el método de Gauss-Seidel, cuya matriz de iteración es:

$$G(x) = Bx + c; \text{ con } B = -(D + L)^{-1} U; \;\; c = (D + L)^{-1} b$$

Igual que Jacobi el método es convergente si y sólo si $\rho(B) < 1$, y también converge para sistemas lineales $Ax = b$ con A estrictamente diagonal dominante. Como vimos esto equivale a despejar x_1 de la primera ecuación, x_2 de la segunda, etc. pero usando los nuevos valores tan pronto como se obtengan.

A continuación presentamos un **block** para resolver un sistema por el método de Gauss-Seidel (en el supuesto de que sea convergente) con un número máximo de iteraciones y de nuevo usando como criterio de parada el basado en la tolerancia relativa entre dos iteraciones consecutivas. Para hallar las normas de los vectores *x* de incógnitas, hemos elegido igual que en Jacobi la norma 1 y la calculamos con el comando de Maxima **mat_norm(x,1)**.

La única diferencia, pero sustancial, con Jacobi en la programación es que en la fórmula que calcula los valores de las incógnitas no se usa *xant* sino *x*. El valor de *xant* se sigue almacenando pero ahora exclusivamente para poder aplicar el criterio de parada.

```
(%i1)    kill(all)$
(%i1)    /* Función gauss_seidel(A,b,tolr,maxiter) */
         /* Resuelve un sistema de n ecuaciones lineales Ax=b,
            mediante el método de Gauss-Seidel */
         /* ARGUMENTOS DE ENTRADA: */
         /* A ..... Matriz de coeficientes */
         /* b ..... Matriz columna de términos independientes */
         /* tolr .. Tolerancia relativa entre dos iteraciones */
         /* maxiter .Número máximo de iteraciones */
         /* ARGUMENTOS DE SALIDA: */
         /* x ... Vector de incógnitas x[n] */
         /* N ... Número de Iteraciones */;
         gauss_seidel(A,b,tolr,maxiter):=block([numer],numer:true,
         n:matrix_size(b)[1],
         x:zeromatrix(n,1),
         xant:zeromatrix(n,1),
             for k:1 thru maxiter do
             (
                 for i:1 thru n do(
                 x[i]:1/A[i,i]*(b[i]-sum(A[i,j]*x[j],j,1,i-1)
                 -sum(A[i,j]*x[j],j,i+1,n))
                     ),
             if abs((mat_norm(x,1)-mat_norm(xant,1))/mat_norm(x,1))
                  < tolr then
             return((print("La aproximación es =",x,
                           "en", k, "iteraciones","")),
             xant:ev(x),
             if k=maxiter then
             print("No lograda la aproximación deseada en ",
             maxiter," iteraciones")
             )
         )$
```

Ahora lo aplicamos al sistema lineal de seis ecuaciones (estrictamente diagonal dominante):

```
(%i3)   A: matrix([10,-3,2,1,1,0],[1,8,-1,3,0,1],[0,1,12,-5,0,2],
        [2,-2,3,20,1,1],[-2,-1,1,3,15,2],[1,-2,1,-1,1,9]);
        b: matrix([16],[-1],[0],[29],[38],[31]);
```

(A)

$$\begin{bmatrix} 10 & -3 & 2 & 1 & 1 & 0 \\ 1 & 8 & -1 & 3 & 0 & 1 \\ 0 & 1 & 12 & -5 & 0 & 2 \\ 2 & -2 & 3 & 20 & 1 & 1 \\ -2 & -1 & 1 & 3 & 15 & 2 \\ 1 & -2 & 1 & -1 & 1 & 9 \end{bmatrix}$$

(b)

$$\begin{bmatrix} 16 \\ -1 \\ 0 \\ 29 \\ 38 \\ 31 \end{bmatrix}$$

La solución aproximada tras 22 iteraciones es:

```
(%i4)   gauss_seidel(A,b,10^-12,100)$
0 errors, 0 warnings
```

$$La\ aproximación\ es = \begin{bmatrix} 1.000000000001566 \\ -1.000000000001314 \\ 4.881280564935271\ 10^{-13} \\ 0.999999999999522 \\ 1.999999999999914 \\ 2.999999999999436 \end{bmatrix} en\ 22\ iteraciones$$

Que es una buena aproximación a la solución exacta dada por x = (1,-1, 0, 1, 2, 3).

Para comparar la velocidad de convergencia de Jacobi y Gauss-Seidel vamos a usar este ejemplo, tomando la misma tolerancia relativa en ambos métodos (volviendo a cargar Jacobi).

```
(%i6)   jacobi(A,b,10^-12,100)$
```

$$La\ aproximación\ es = \begin{bmatrix} 1.000000000000399 \\ -0.9999999999988944 \\ -1.757575566567008\ 10^{-13} \\ 1.000000000000512 \\ 2.000000000000265 \\ 3.000000000000376 \end{bmatrix} en\ 28\ iteraciones$$

Comprobamos que es más rápido Gauss-Seidel, como era de esperar, dado que actualiza inmediatamente los valores obtenidos. Podemos ver lo mismo si consideramos el sistema:

$$\left.\begin{array}{rl} 3x_1 + 3x_2 + 7x_3 & = 1 \\ 3x_1 + 6x_2 + 2x_3 & = 2 \\ 3x_1 - x_2 + x_3 & = -1 \end{array}\right\}$$

Vamos a aplicar Jacobi y Gauss-Seidel a ver si convergen a la solución antes y después de escribirlo en forma diagonalmente dominante estrictamente.

```
(%i8)    A: matrix([3,3,7],[3,6,2],[3,-1,1]);
         b: matrix([1],[2],[-1]);
```

$$(A)\quad \begin{bmatrix} 3 & 3 & 7 \\ 3 & 6 & 2 \\ 3 & -1 & 1 \end{bmatrix}$$

$$(b)\quad \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$$

```
(%i9)    jacobi(A,b,10^-6,200)$
No lograda la aproximación deseada en  200  iteraciones
(%i10)   gauss_seidel(A,b,10^-6,200)$
No lograda la aproximación deseada en  200  iteraciones
(%i12)   A:rowswap(A,1,3);
         b:rowswap(b,1,3)$
```

$$(A)\quad \begin{bmatrix} 3 & -1 & 1 \\ 3 & 6 & 2 \\ 3 & 3 & 7 \end{bmatrix}$$

```
(%i13)   jacobi(A,b,10^-6,200)$
```

$$\text{La aproximación es} = \begin{bmatrix} -0.210526306696677 \\ 0.4210527305139183 \\ 0.05263168961567881 \end{bmatrix} \text{ en } 16 \text{ iteraciones}$$

```
(%i14)   gauss_seidel(A,b,10^-6,200)$
```

$$\text{La aproximación es} = \begin{bmatrix} -0.2105262950250222 \\ 0.4210525918044635 \\ 0.05263158709452517 \end{bmatrix} \text{ en } 10 \text{ iteraciones}$$

Nota: Se puede probar también que si un sistema tiene la matriz de coeficientes estrictamente diagonalmente dominante se puede realizar el método de Eliminación de Gauss sin tener que permutar filas.

6.5 Ejercicios Propuestos

Ejercicio 1. Se considera el sistema *Ax* = *b*, donde:

$$A = \begin{pmatrix} -4 & 1 \\ 1 & -4 \end{pmatrix}; \quad b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

a) Calcular el número de condición de *A*.
b) Denotando *z* a la solución del sistema y *z*1 a la del sistema resultante al sumar una milésima al último elemento de *A*, hallar, en porcentaje, el error relativo cometido al hacer *z*1 ≈ *z*, así como el mayor error absoluto cometido en las incógnitas, determinando en cuál de ellas se produce dicho error máximo.

Ejercicio 2. Repítase el ejemplo anterior con el sistema *Ax* = *b*, donde:

$$A = \begin{pmatrix} 3.021 & 2.714 & 6.913 \\ 1.031 & -4.273 & 1.121 \\ 5.084 & -5.832 & 9.155 \end{pmatrix}; \quad b = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Ejercicio 3. Resolver el sistema:

$$\left.\begin{array}{rl} x_1 + 3x_2 + x_3 & = 1 \\ 4x_1 + x_2 - x_3 + 2x_4 & = 1 \\ x_1 + 2x_2 + 5x_3 + x_4 & = 1 \\ x_1 - x_2 + 2x_3 + 6x_4 & = 1 \end{array}\right\}$$

por el método de Jacobi, con tolerancia relativa 10^{-6}, comprobando previamente que el método es aplicable y convergente. Comprobar la solución.

Ejercicio 4. Resolver el sistema:

$$\left.\begin{array}{rl} 5x_1 + 2x_2 + x_3 & = 1 \\ -x_1 + x_2 + 4x_3 & = -1 \\ 3x_1 - 6x_2 + 2x_3 & = 2 \end{array}\right\}$$

por el método de Jacobi, con *x*(0)= (0, 0, 0) y con tolerancia relativa 10^{-6}. Comprobar la solución. Arranca la iteración a partir de otro valor inicial, por ejemplo *x*(0) = (1000, 1000, 1000) y comprueba si la iteración también converge a la solución.

Ejercicio 5. Resolver el mismo sistema anterior, pero ahora con el método de Gauss-Seidel. Comparar la velocidad de convergencia para distintos valores de tolerancia relativa. ¿Qué observas?

Ejercicio 6. Pon en forma adecuada los sistemas:

$$(a)\ \left.\begin{array}{rl} -y + 2z & = 2 \\ -x + 2y - z & = 0 \\ 2x - y & = 1 \end{array}\right\}; \quad (b)\ \left.\begin{array}{rl} -x - z + 3t & = 10 \\ x + y + 4z - t & = 0 \\ x + 5y + z - t & = 2 \\ 6x + 2y + z - t & = 0 \end{array}\right\}$$

y aplica los métodos de Jacobi y Gauss Seidel a partir del vector nulo (0, 0, 0). Comprueba la solución.