SISTEMAS DE ECUACIONES

1. Métodos Directos
2. Sistemas lineales triangulares

* **TRIANGULAR SUPERIOR**
* **ALGORITMO**
* **CÓDIGO**

function sol=back(A,b)

N=length(A);

sol(N)=b(N)/A(N,N);

for k= N-1:-1:1

suma=0;

for i=k+1:N

suma=suma+A(k,i)\*sol(i);

end

sol(k)=(b(k)-suma)/A(k,k);

end

* **EJEMPLO**

Resolver el siguiente sistema:

Como se puede apreciar es una solución hacia atrás, si mis incógnitas fusen primero calcularía el , luego el programa usara el mismo para hallar el y después usara el y para hallar el , y así sucesivamente hasta hallar el .

* **TRIGUANGULAR INFERIOR**
* **ALGORITMO**
* **CODIGO**

function sol=next(A,b)

N=length(A);

sol(1)=b(1)/A(1,1);

for k=2:N

suma=0;

for i=1:k-1

suma=suma+A(k,i)\*sol(i);

end

sol(k)=(b(k)-suma)/A(k,k);

end

* **EJEMPLO**

En este programa podemos apreciar que es una solución hacia adelante, es decir si mis incógnitas fusen primero calcularía el y luego el programa usara el mismo para hallar el y después usara el y para hallar el , y así sucesivamente hasta hallar el .

1. Método de eliminación gaussiana

* **ELIMINACION GAUSSIANA INFERIOR**
* **ALGORITMO**
* **CODIGO**

function sol=elimagaus(A,b)

N=length(A);

C=[A b];

for i=1:N-1

for k=i+1:N

z=C(k,i)./C(i,i);

C(k,:)=C(k,:)-z.\*C(i,:);

end

end

A=C(:,1:N);

b=C(:,N+1);

sol=back(A,b);

* **EJEMPLO**

Este programa se utiliza para hallar las soluciones de un sistema de ecuaciones, sin la necesidad de que mi arreglo matricial sea triangular superior o inferior, ya que este programa consiste en triangulisar una matriz ya sea superiormente o inferiormente; primero por medio de sumas o restas de la primera fila, multiplicado por un factor, con las filas que se encuentran debajo de dicha fila, convertirá en ceros a todos los elementos que se encuentren por debajo de su segundo elemento de la primera columna y así sucesivamente a partir de la segunda fila como se muestra en el ejemplo, una vez triangulizado se pude aplicar el programa back y obtener la solución

* **ELIMINACION GASSIANA SUPERIORMENTE**
* **CODIGO**

function sol=elimgaussup(A,b)

A=input('ingresa tu matriz A=')

b=input('ingresa tu vector b=')

N=length(A);

C=[A b];

for j=N:-1:2

for i=j-1:-1:1

C(i,:)=C(i,:)-(C(i,j)/C(j,j))\*C(j,:)

end

C=C(:,1:N+1);

end

A=C(:,1:N);

b=C(:,N+1);

sol(1)=b(1)/A(1,1);

for k=2:N

suma=0;

for i=1:k-1

suma=suma+A(k,i)\*sol(i);

end

sol(k)=(b(k)-suma)/A(k,k);

end

* **EJEMPLO**

Pivoteo

* **P.PARCIAL**
* **ALGORITMO**
* **CODIGO**

function sol=elimagauspivo(A,b)

A=input('ingresa tu matriz A=')

[N,M]=size(A);

if M~=N

disp('La matriz no es cuadrada')

return

else

disp('La matriz es cuadrada')

end

b=input('dame tuscorrdenadas b=')

b=b'

C=[A b];

fori=1:N-1

[y,x]=max(abs(A(i:N,i)));

x;

H=C(i,:);

C(i,:)=C(i+x-1,:);

C(i+x-1,:)=H;

C

for k=i+1:N

if C(i,i)~=0

z=C(k,i)./C(i,i);

C(k,:)=C(k,:)-z.\*C(i,:);

else

disp('procesoerrado')

end

end

C

end

A=C(:,1:N);

b=C(:,N+1);

disp('la solucion es')

sol=back(A,b)

* **EJEMPLO**

La matriz es cuadrada

Dame tu vector

* **P.TOTAL**
* **ALGORITMO**
* **CODIGO**

function sol=elimgauspivototal(A,b)

A=input('ingresa tu matriz A=')

[N,M]=size(A);

if M~=N

disp('La matriz no es cuadrada')

return

else

disp('La matriz es cuadrada')

end

b=input('dame tuscorrdenadas b=')

b=b';

C=[A b];

fori=1:N-1

for l=i:N

S(l)=max(abs(C(l,i:N)));

end

S(i:N)

[y,x]=max(S(i:N));

x;

H=C(i,:);

C(i,:)=C(i+x-1,:);

C(i+x-1,:)=H;

C;

for m=i:N

T(m)=max(abs(C(i:N,m)));

end

T(i:N)

[p,q]=max(T(i:N));

q;

R=C(:,i);

C(:,i)=C(:,i+q-1);

C(:,i+q-1)=R;

C;

for k=i+1:N

if C(i,i)~=0

z=C(k,i)./C(i,i);

C(k,:)=C(k,:)-z.\*C(i,:);

else

disp('procesoerrado')

end

end

C;

end

A=C(:,1:N);

b=C(:,N+1);

disp('la solucion es')

sol=back(A,b)

* **EJEMPLO**
* **P.ESCALAR**
* **ALGORITMO**
* **CODIGO**

function sol=elimagauspivoescalar(A,b)

A=input('ingresa tu matriz A=')

[N,M]=size(A);

if M~=N

disp('La matriz no es cuadrada')

return

else

disp('La matriz es cuadrada')

end

b=input('dame tuscorrdenadas b=')

b=b';

C=[A b];

fori=1:N-1

for l=1:N

s(l)=max(abs(C(l,1:N)));

end

a=s(i:N)';

[y,x]=max(abs(C(i:N,i)./a));

x;

H=C(i,:);

C(i,:)=C(i+x-1,:);

C(i+x-1,:)=H;

C

for k=i+1:N

if C(i,i)~=0

z=C(k,i)./C(i,i);

C(k,:)=C(k,:)-z.\*C(i,:);

else

disp('procesoerrado')

end

end

C

end

A=C(:,1:N);

b=C(:,N+1);

disp('la solucion es')

sol=back(A,b)

* **EJEMPLO**

**ADICIONALES:**

* **INVERSA DE UNA MATRIZ NXN APLICANDO PIVOTEO PARCIAL**
* **CODIGO**

functioninver=inver(A)

A=input('ingresa tu matriz A=')

[N,M]=size(A);

if M~=N

disp('La matriz no es cuadrada')

return

else

disp('La matriz es cuadrada')

end

if deter(A)~=0

D=eye(N);

Q=A;

inver=[];

for j=1:N

C=[Q D(:,j)];

for i=1:N-1

[y,x]=max(abs(A(i:N,i)));

x;

H=C(i,:);

C(i,:)=C(i+x-1,:);

C(i+x-1,:)=H;

C;

for k=i+1:N

z=C(k,i)./C(i,i);

C(k,:)=C(k,:)-z.\*C(i,:);

end

C;

end

A=C(:,1:N);

b=C(:,N+1);

sol=back(A,b);

inver=[inver , sol'];

end

disp('La Matriz Inversa es')

IA=inver

else

disp('La Matriz A no posee inversa')

return

end

* **EJEMPLO**

El programa aplica eliminación Gaussiana hacia atrás a las matrices A1, A2, A3, todas esas operaciones se omitirá en este programa ya que se mostrara al principio luego la inversa es justamente el desarrollo de las ecuaciones de los sistemas A1, A2 y A3, esto es:

NOTA: se puede comprobar este programa con la “inv(A)” del mismo matlab

* **SOLUCION PARA N SISTEMAS LINEALES APLICANDO PIVOTEO PARCIAL**

function resol=sistemasNXN(A,D)

A=input('ingresa tu matriz A=')

D=input('dame tu cordenadas D=')

[N,M]=size(A);

[N,M]=size(D);

if M~=N;

disp('La matriz no es cuadrada')

return

else

disp('La matriz es cuadrada')

end;

resol=[];

Q=A;

for j=1:N

C=[Q D(:,j)]

for i=1:N-1

[y,x]=max(abs(A(i:N,i)));

x;

H=C(i,:);

C(i,:)=C(i+x-1,:);

C(i+x-1,:)=H;

C;

for k=i+1:N

z=C(k,i)./C(i,i);

C(k,:)=C(k,:)-z.\*C(i,:);

C

end

end

A=C(:,1:N);

b=C(:,N+1);

sol=back(A,b)

resol=[resol,sol'];

end

disp('la soluciones de cada sistema son:')

R=resol

* **EJEMPLO**

ingresa tu matriz

Dame tu coordenadas

Las matrices son cuadradas

Las soluciones de cada sistema son:

1. Método de descomposición LU

* **FACTORIZACION LU (ELIMINACION GAUSSINA)**
* **ALGORITMO**

|  |
| --- |
|  |

* **CODIGO**

function[L,U]=lluu(A)

A=input('ingresa tu matriz A=')

[N,M]=size(A);

if M~=N

disp('La matriz no es cuadrada')

return

else

disp('La matriz es cuadrada')

end

M=eye(N);

fori=1:N-1

for k=i+1:N

if A(i,i)~=0

M(k,i)=(A(k,i)./A(i,i));

A(k,:)=A(k,:)-M(k,i).\*A(i,:);

else

disp('La matriz no posee factorización')

end

end

end

disp('la factorizaciones')

L=M

U=A

* **EJEMPLO**

*A=*

VAMOS A FACTORIZAR LA MATRIZ A =LU

L=

U=

L=

U=

*L=*

*U=*

*L=*

*U=*

VALORES FINALES DE LA MATRIZ L Y U

*L=*

*U=*

B=L\*U= A*=*

* **FACTORIZACION LU PARA UNA TRIAGONAL**
* **ALGORITMO**

|  |
| --- |
|  |

* **CODIGO**

functiontridiagonal(A,b)

A=input('ingresa tu matriz A=')

[N,M]=size(A);

if M~=N

disp('La matriz no es cuadrada')

return

else

disp('La matriz es cuadrada')

end;

b=input('dame tuscordenadas b=')

N=length(A);

U(1,1)=A(1,1);

L(2,1)=A(2,1)/U(1,1);

U(2,2)=A(2,2)-A(2,1)\*A(1,2)/A(1,1);

L(3,2)=A(3,2)/U(2,2);

fori=3:N

U(i,i)=A(i,i)-A(i-1,i)\*A(i,i-1)/U(i-1,i-1);

L(i,i-1)=A(i,i-1)/U(i-1,i-1);

end

for j=1:N-1

U(j,j+1)=A(j,j+1);

end

for k=1:N

L(k,k)=1;

end

disp('La factorizacion es')

display(L)

display(U)

y=elimagauspivo1(L,b);

B=L\*U;

disp('la comprobacion es')

display(B)

disp('la solucion del sistema es')

x=elimagauspivo1(U,y)

* **EJEMPLO**

A=

VEAMOS LA FACTORIZACION:

U=

L=

U=

U=

U=

finalmente se tiene :

L=

U=

luego comprobando:

B=L\*U=

**NOTA:** este algoritmo esta exclusivamente hecho para matrices tridiagonales, con su respectiva solución, solamente insertamos nuestro “b” y nos botara las soluciones, solamente hemos puesto su factorización respectiva.

1. Factorización Matricial

* **METODO DE CROUT**
* **ALGORITMO**

|  |
| --- |
|  |

* **CODIGO**

function CROUT(A)

A=input('ingresatumatriz A=')

[N,M]=size(A);

if M~=N

disp('La matriz no es cuadrada')

return

else

disp('La matriz es cuadrada')

end

U=eye(N);

for k=1:N

fori=1:N

suma1=0;

for s=1:k-1

suma1=suma1+L(i,s)\*U(s,k);

end

L(i,k)=(A(i,k)-suma1)/U(k,k)

end

if L(k,k)~=0

for j=1:N

suma2=0;

for s=1:k-1

suma2=suma2 +L(k,s)\*U(s,j)

end

U(k,j)=(A(k,j)-suma2)/L(k,k)

end

else

disp ('la matriz no tiene solucion')

end

end

L

U

B=L\*U

* **EJEMPLO**

A=

U=

U=

U=

U=

U=

U=

U=

U=

U=

L=

U=

FINALMENTE COMPROBAMOS

B=L\*U=

* **METODO DE DOOLITTE**
* **ALGORITMO**
* **CODIGO**

function [L,U]=COUT1(A)

A=input('ingresa tu matriz A=')

[N,M]=size(A);

if M~=N

disp('La matriz no es cuadrada')

return

else

disp('La matriz es cuadrada')

end

L=eye(N); U=[];

U(1,1)=A(1,1);

for p=2:N

if U(1,1)~=0

U(1,p)=A(1,p);

L(p,1)=A(p,1)/U(1,1);

else ('la matriz no posee factorizacion')

end

end

fori=2:N

SUMA1=0;

for k=1:i-1

SUMA1=SUMA1+L(i,k)\*U(k,i);

end

U(i,i)=A(i,i)-SUMA1;

for j=i+1:N

SUMA1=0;

SUMA2=0;

for k=1:i-1

SUMA1=SUMA1+L(i,k)\*U(k,j);

SUMA2=SUMA2+L(j,k)\*U(k,i);

end

U(i,j)=A(i,j)-SUMA1;

L(j,i)=(A(j,i)-SUMA2)/U(i,i);

end

end

display(U)

display(L)

B=L\*U

* **EJEMPLO**

* **FACTORIZACION LU PARA UN PENTADIAGONAL**
* **ALGORITMO**

|  |
| --- |
|  |

* **CODIGO**

function pentadiagonal4(A)

A=input('ingresa tu matrizpentadiagolnal')

N=length(A)

L=eye(N)

for i=1:2

U(1,i)=A(1,i)

end

for k=1:N-2

U(k,k+2)=A(k,k+2)

end

L(2,1)=A(2,1)/U(1,1)

U(2,2)=A(2,2)-L(2,1)\*U(1,2)

U(2,3)=A(2,3)-L(2,1)\*U(1,3)

for i=3:N-1

L(i,i-2)=A(i,i-2)/U(i-2,i-2)

L(i,i-1)=(A(i,i-1)-L(i,i-2)\*U(i-2,i-1))./U(i-1,i-1)

U(i,i)=A(i,i)-(L(i,i-2)\*U(i-2,i)+L(i,i-1)\*U(i-1,i))

U(i,i+1)=A(i,i+1)-L(i,i-1)\*U(i-1,i+1)

end

L(N,N-2)=A(N,N-2)/U(N-2,N-2)

L(N,N-1)=(A(N,N-1)-L(N,N-2)\*U(N-2,N-1))./U(N-1,N-1)

U(N,N)=A(N,N)-(L(N,N-2)\*U(N-2,N)+L(N,N-1)\*U(N-1,N))

B=L\*U

* **EJEMPLO**
* **METODO DE CHOLESKY**
* **ALGORITMO**
* **CODIGO**

function LLT(A)

A=input('ingresa tu matriz simetrica A=')

[N,M]=size(A);

fori=1:N

deter=det(A(1:i,1:i));

if deter<=0

disp('la matriz A no esta definida positiva')

disp('la matriz no factorizable A=L\*LT')

return

end

end

disp('la matriz A espositiva')

L(1,1)=(A(1,1))^(1/2);

for j=2:N

L(j,1)=A(j,1)/L(1,1);

end

for k=2:N-1

suma1=0;

for t=1:k-1

suma1=suma1+L(k,t)^2;

end

L(k,k)=(A(k,k)-suma1)^(1/2);

fori=2:k

suma2=0;

for t=1:i-1

suma2=suma2+L(k+1,t)\*L(i,t);

end

if L(i,i)~=0

L(k+1,i)=(A(k+1,i)-suma2)/L(i,i);

else

disp('La matriz no posee factorizacion')

return

end

end

end

suma3=0;

for t=1:N-1

suma3=suma3+L(N,t)^2;

end

L(N,N)=(A(N,N)-suma3)^(1/2);

display(L)

LT=L';

display(LT)

disp('su comprobacion es')

B=L\*LT

* **EJEMPLO**

*A=*

*veamos*

*L=*

*L=*

*L=*

*L=*

*L=*

*L=*

*L=*

*L=*

*L=*

*Lt=*

*Comprobando:*

*L\*Lt = A=*

* **FACTORIZACION LDU**
* **ALGORITMO**
* **CODIGO**

function LDU(A)

A=input('ingresa tu matriz A=')

[N,M]=size(A);

if M~=N

disp('La matriz no es cuadrada')

return

else

disp('La matriz es cuadrada')

end

L=eye(N)

U=eye(N)

D=zeros(N)

fori=1:N

suma1=0;

for m=1:i-1

suma1=suma1+L(i,m)\*D(m,m)\*U(m,i);

end

D(i,i)=A(i,i)-suma1

for j=i+1:N

suma2=0;

suma3=0;

for m=1:i-1

suma2=suma2+L(i,m)\*D(m,m)\*U(m,j);

suma3=suma3+L(j,m)\*D(m,m)\*U(m,i);

end

if D(i,i)~=0

U(i,j)=(A(i,j)-suma2)/D(i,i)

L(j,i)=(A(j,i)-suma3)/D(i,i)

else

disp('La matriz no posee factorizacion')

return

end

end

end

display(L)

display(U)

display(D)

B=L\*D\*U

* **EJEMPLO**

Aplicar la factorizacion LDU para la siguiente matriz:

* **FACTORIZACION UDL**
* **ALGORITMO**

|  |
| --- |
|  |

* **CODIGO**

function UDL(A)

A=input('ingresa tu matriz A=')

[N,M]=size(A);

if M~=N

disp('La matriz no es cuadrada')

return

else

disp('La matriz es cuadrada')

end

L=eye(N)

U=eye(N)

D=zeros(N)

D(N,N)=A(N,N)

for k=N-1:-1:1

if D(N,N)~=0

L(N,k)=A(N,k)/D(N,N)

U(k,N)=A(k,N)/D(N,N)

else

disp('La matriz no se puede factorizar')

return

end

end

D(N-1,N-1)=A(N-1,N-1)-U(N-1,N)\*D(N,N)\*L(N,N-1);

for k=N-2:-1:1

for j=k:-1:1

suma3=0

for t=k+2:N

suma3=suma3+U(k+1,t)\*D(t,t)\*L(t,j);

end

L(k+1,j)=(A(k+1,j)-suma3)/D(k+1,k+1);

end

for j=N-1:-1:k+1

suma1=0;

for t=j+1:N

suma1=suma1+U(k,t)\*D(t,t)\*L(t,j);

end

U(k,j)=(A(k,j)-suma1)/D(j,j);

end

suma2=0;

for t=k+1:N

suma2=suma2+U(k,t)\*D(t,t)\*L(t,k);

end

D(k,k)=A(k,k)-suma2;

end

display(U)

display(D)

display(L)

B=U\*D\*L

* **EJEMPLO**
* **FACTORIZACION LDLT**
* **ALGORITMO**
* **CODIGO**

function LDLT(A)

A=input('ingresa tu matriz simetrica A=')

[N,M]=size(A);

fori=1:N

deter=det(A(1:i,1:i));

if deter<=0

disp('la matriz A no esta definida positiva')

disp('la matriz no factorizable A=U\*UT')

return

end

end

disp('la matriz A espositiva')

N=length(A);

L=eye(N)

D=zeros(N)

fori=1:N

suma1=0;

for m=1:i-1

suma1=suma1+D(m,m)\*(L(i,m)^2);

end

D(i,i)=A(i,i)-suma1

for j=i+1:N

suma2=0;

for m=1:i-1

suma2=suma2+L(j,m)\*D(m,m)\*L(i,m)

end

if D(i,i)~=0

L(j,i)=(A(i,j)-suma2)/D(i,i)

else

disp('La matriz no posee factorizacion')

return

end

end

end

display(L)

LT=L';

display(LT)

display(D)

B=L\*D\*L'

* **EJEMPLO**

Aplicar la factorización LDLT para la siguiente matriz:

* **FACTORIZACION UDUT**
* **ALGORITMO**

|  |
| --- |
|  |

* **CODIGO :**

function UDUT(A)

A=input('ingresa tu matriz simetrica A=')

[N,M]=size(A);

fori=1:N

deter=det(A(1:i,1:i));

if deter<=0

disp('la matriz A no esta definida positiva')

disp('la matriz no factorizable A=U\*UT')

return

end

end

disp('la matriz A es positiva')

U=eye(N)

D(N,N)=A(N,N)

if D(N,N)~=0

U(1,N)=A(N,1)/D(N,N)

else

disp('La matriz no posee factorizacion')

return

end

fori=N-1:-1:2

U(i,N)=A(N,i)/D(N,N)

for j=i-1:-1:1

suma1=0

suma2=0

for t=i+1:N

suma1=suma1+D(t,t)\*((U(i,t))^2)

suma2=suma2+U(i,t)\*D(t,t)\*U(j,t)

end

D(i,i)=A(i,i)-suma1

U(j,i)=(A(i,j)-suma2)/D(i,i)

end

end

suma3=0

for t=2:N

suma3=suma3+D(t,t)\*((U(1,t))^2)

end

D(1,1)=A(1,1)-suma3

display(U)

display(D)

UT=U';

display(UT)

disp('su comprabacion es')

B=U\*D\*UT

U

D

UT=U'

B=U\*D\*UT

* **EJEMPLO**

A=

U=

U=

U=

U=

U=

U=

U=

U=

U=

U=

D=

UT =

COMPROBANDO OBTENEMOS

B=U\*D\*UT

B=

* **FACTORIZACION UUT**
* **ALGORITMO**

|  |
| --- |
|  |

* **CODIGO**

function UUT(A)

A=input('ingresa tu matriz simetrica A=')

[N,M]=size(A);

fori=1:N

deter=det(A(1:i,1:i));

if deter<=0

disp('la matriz A no esta definida positiva')

disp('la matriz no factorizable A=U\*UT')

return

end

end

disp('la matriz A es positiva')

U(N,N)=A(N,N)^(1/2)

for k=N-1:-1:1

if U(N,N)~=0

U(k,N)=A(N,k)/U(N,N)

else

disp('La matriz no posee factorizacion')

return

end

end

for k=N-1:-1:2

for j=k-1:-1:1

suma1=0;

suma2=0;

fori=k+1:N

suma1=suma1+(U(k,i))^2;

suma2=suma2+U(k,i)\*U(j,i);

end

U(k,k)=(A(k,k)-suma1)^(1/2)

U(j,k)=(A(k,j)-suma2)/U(k,k)

end

end

U(1,N)=A(N,1)/U(N,N);

suma3=0;

fori=2:N

suma3=suma3+(U(1,i))^2;

end

U(1,1)=(A(1,1)-suma3)^(1/2)

display(U)

UT=U';

display(UT)

disp('La comprobacion es')

B=U\*UT

* **EJEMPLO**

Aplicar la factorizacion UUT para la siguiente matriz:

* **FACTORIZACION UL**
* **ALGORITMO**

|  |
| --- |
|  |

* **CODIGO**

function UL(A)

A=input('ingresa tu matriz A=')

[N,M]=size(A);

if M~=N

disp('La matriz no es cuadrada')

return

else

disp('La matriz es cuadrada')

end

L=eye(N);

for k=N:-1:1

U(k,N)=A(k,N)

if U(N,N)~=0

L(N,k)=A(N,k)/U(N,N)

else

disp('factorizacionimposible')

return

end

end

for p=N-1:-1:1

suma1=0;

for t=p+1:N

suma1=suma1+U(p,t)\*L(t,p);

end

U(p,p)=A(p,p)-suma1

for i=p-1:-1:1

suma2=0;

suma3=0;

for t=p+1:N

suma2=suma2+U(i,t)\*L(t,p);

suma3=suma3+U(p,t)\*L(t,i);

end

if U(p,p)~=0

U(i,p)=A(i,p)-suma2

L(p,i)=(A(p,i)-suma3)/U(p,p)

else

dip('Factorizacionimposible')

return

end

end

end

suma4=0;

for t=2:N

suma4=suma4+U(1,t)\*L(t,1);

end

U(1,1)=A(1,1)-suma4

U

L

disp('la comprobacion es')

B=U\*L

* **EJEMPLO**

Aplicar el programa UL(A) para la factorización de la siguiente matriz:

**NOTA:** podemos combinar los programas factorización UL y inver1 de una matriz para así hallar su inversa; todo lo ponemos en un solo pack solo agregaremos al último del código de UL; inver1(U) y inver1(L) luego llamemos AI=inversa de A; y IU=inver1(U); IL=inver1(L) concluimos con IA=IL\*IU

La nota dada es una aplicación que le podemos dar a la factorización tanto LU como UL; se vuelve más sencillo el trabajo ya que solo se sacara inversa de U y L lo cual es fácil ya que se son matrices superiores e inferiores respectivamente.

1. Determinante

* **Determinante**
* **ALGORITMO**

|  |
| --- |
|  |

* **CODIGO**

function d=deter(A)

A=input('ingresa tu matriz A=');

if n==1

x=A(1,1);

else

suma=0;

fori=1:n

ifi==1

M=A(2:n,2:n);

else

M=[A(2:n,1:i-1) A(2:n,i+1:n)];

end

suma=suma+A(1,i)\*(-1)^(i+1)\*deter(M);

end

x=suma;

end

d=x;

* **EJEMPLO**
* **ADJUNTA DE UNA MATRIZ NXN APLICANDO DETERMINANTE**
* **CODIGO**

function adjunta(A)

A=input('ingresa tu matriz A=')

J=[];

inver=inver1(A);

if deter1(A)~=0;

d=deter1(A);

c=inver/d;

disp('La adjunta es')

J=[J,c']

else

disp('no exiteadjunta')

return

end

* **EJEMPLO**

*A=*

*INVER:*

*DETER: 155*

*Luego*

*C=*

*LA ADJUNTA ES :*

*J=*

Métodos iterativos

1. Normas

* **NORMAS CON CENTRO (Circunferencias)**
* **ALGORITMO**

|  |
| --- |
|  |

* **CODIGO**

v=input ('dame tu vector v=');

p=2

x=v(1)-1:0.01:v(1)+1;

y=v(2)+(1-abs(x-v(1)).^p).^(1/p);

plot(x,y);

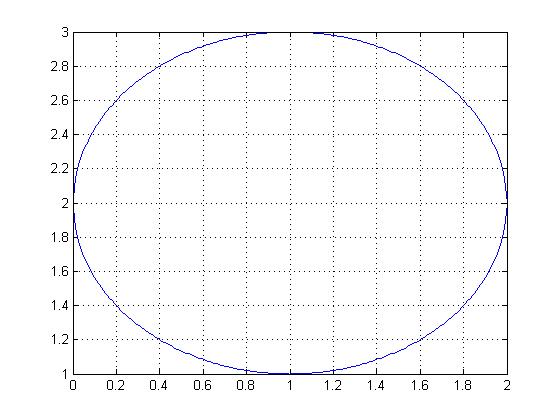
holdon

y=v(2)-(1-abs(x-v(1)).^p).^(1/p);

plot (x,y)

* **EJEMPLO**

**Circunferencia de V=[1,2]**

****

* **NORMAS CON CENTRO (Esferas)**
* **CODIGO**

v=input('dame tu vector v=')

p=2

for x=v(1)-1:0.01:v(1)+1

for y=v(2)-(1-abs(x-v(1)).^p).^(1/p):0.01:v(2)+(1-abs(x-v(1)).^p).^(1/p)

z= v(3)+(1-(abs(x-v(1)).^p)-(abs(y-v(2)).^p)).^(1/p);

plot3(x,y,z)

z= v(3)-(1-(abs(x-v(1)).^p)-(abs(y-v(2)).^p)).^(1/p)

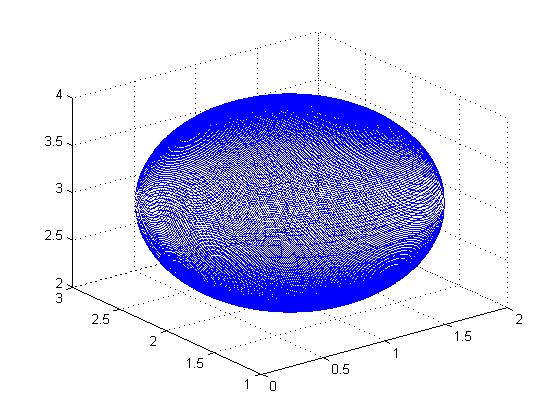
plot3(x,y,z)

holdon

gridon

end

end

* **EJEMPLO**

**GENERALIZAMOS UNA ESFERA CON CENTRO CON CUALQUIER RADIO**

* **ALGORITMO**

|  |
| --- |
|  |

* **CODIGO**

function esfera1(r,C)

r=input('ingresa la constante r=')

E=input('ingrsa el centro de la esfera E=')

for x=E(1)-r:0.1:E(1)+r

for y=E(2)-(r^2-(x-E(1))^2)^(1/2):0.1:E(2)+(r^2-(x-E(1))^2)^(1/2)

for z1=E(3)+r\*((1-((x-E(1))/r)^2-((y-E(2))/r)^2))^(1/2);

plot3(x,y,z1)

holdon

z2=E(3)-r\*((1-((x-E(1))/r)^2-((y-E(2))/r)^2))^(1/2);

plot3(x,y,z2)

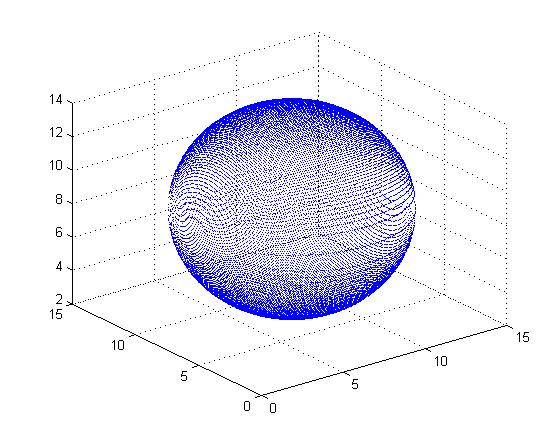
end

gridon

end

end

* **EJEMPLO**

****

**NOTA:** Aprovechando que podemos graficar circunferencias esferas hemos utilizado este código para poder graficar una elipse y elipsoides.

Para la esfera anterior con centro en el origen su algoritmo ya incluido implícitamente la generalizada bastara poner su centro C(0,0) y r=1

* **ELIPSE CON CENTRO EN EL ORIGEN**
* **CODIGO**

M=input('dame tus cordenadas M=')

p=2

for x=-((M(1))^(1/p)):0.001:(M(1))^(1/p)

for y=(abs(M(2)\*(1-(x^2/M(1)))))^(1/p)

plot(x,y)

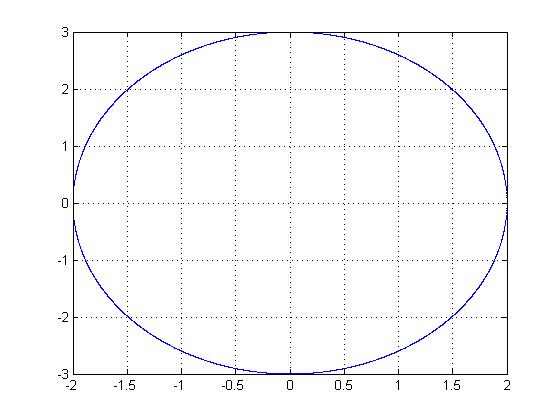
holdon

plot(x,-y)

end

end

* **EJEMPLO**



* **ELIPSE CON CUALQUIER CENTRO**
* **CODIGO**

functionelipse(a,b,C)

a=input('ingresa la constante a=')

b=input('ingresa la constante b=')

C=input('ingresa el centro de la elipse C=')

for x=C(1)-a:0.01:C(1)+a

for y1=C(2)+b\*(1-((x-C(1))/a)^2)^(1/2)

plot(x,y1)

holdon

y2=C(2)-b\*(1-((x-C(1))/a)^2)^(1/2)

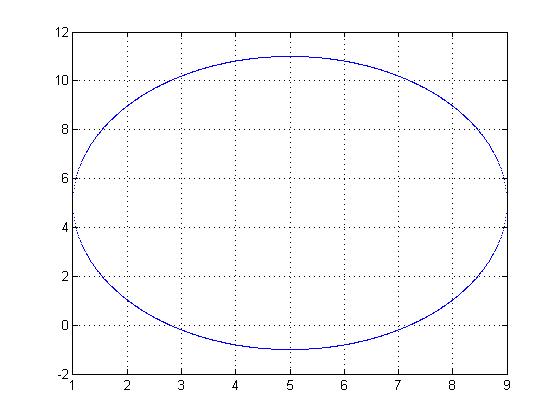
plot(x,y2)

end

end

* **EJEMPLO**

**c=[5 5]**

****

* **ELIPSOIDE CON CENTRO CUALQUIERA**
* **ALGORITMO**

|  |
| --- |
|  |

* **CODIGO**

function elipsoide1(a,b,c)

a=input('ingresa la constante a=')

b=input('ingresa la constante b=')

c=input('ingresa la constante c=')

E=input('ingrsa el centro de la esfera E=')

for x=E(1)-a:0.1:E(1)+a

for y=E(2)-(b^2-((b/a)^2)\*(x-E(1))^2)^(1/2):0.1:E(2)+(b^2-((b/a)^2)\*(x-E(1))^2)^(1/2);

for z1=E(3)+c\*((1-((x-E(1))/a)^2-((y-E(2))/b)^2))^(1/2);

plot3(x,y,z1)

holdon

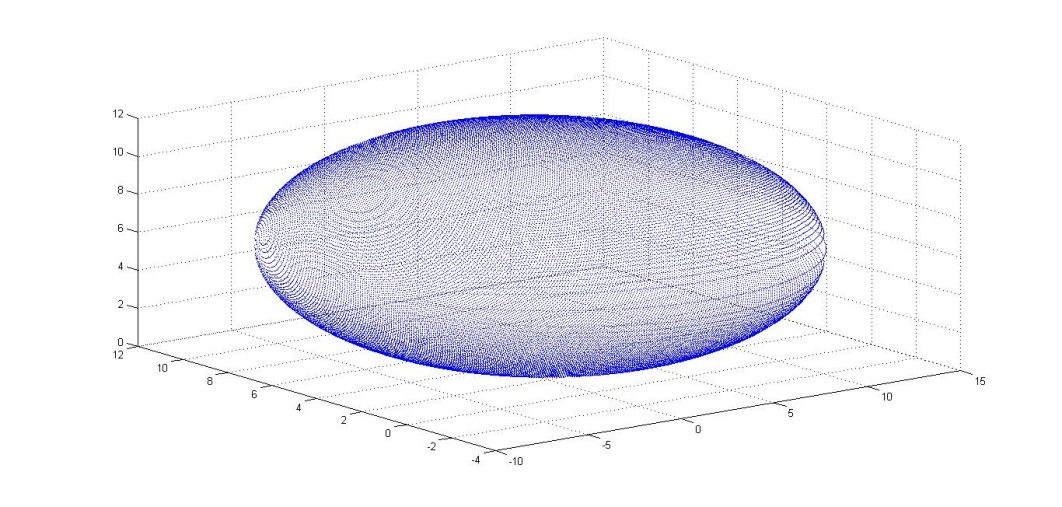
z2=E(3)-c\*((1-((x-E(1))/a)^2-((y-E(2))/b)^2))^(1/2);

plot3(x,y,z2)

end

gridon

* **EJEMPLO**

**a=5**

**b=4**

**c=8**

1. Método de Jacobi

* **ALGORITMO**
* **CODIGO**

function [x,M]=Mjac(A,b,x0,er)

A=input('ingresa tu matriz cuadrada A=')

N=length(A)

b=input('ingresa tus coordenadas b=')

b=b'

x0=input('ingresa tu punto inicial x0=')

x0=x0'

er=input('ingresa error er=')

fori=1:N

if A(i,i)~=0

R(i,:)=A(i,:)./A(i,i);

P(i)=b(i)./A(i,i);

else

disp('procesofallado')

end

end

R;

P;

B=eye(N)-R;

error=1;

M=[0 , error, x0'];

s=0;

while error>er

x=B\*x0+P';

error=norm(x-x0)/norm(x);

x0=x;

s=s+1;

M=[M ;[s,error,x0'] ];

end

M

* **EJEMPLO**

Aplicar el método de jacobipara resolver el siguiente sistema

Con

Mjac(A,[-2 -1 0 1]',[0 0 0 0]',0.01)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **k** | **xk1** | **xk2** | **xk3** | **xk4** | **ERROR** |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | -0.5 | -0.25 | 0 | 0.3333 |
| 2 | 0.4450 | -0.5208 | -0.0417 | -0.2167 | 0.4167 |
| 3 | 0.2246 | -0.6479 | -0.0698 | -0.1958 | 0.5653 |
| 4 | 0.1099 | -0.6728 | 0.0043 | -0.2566 | 0.5913 |
| 5 | 0.0674 | -0.7131 | 0.0019 | -0.2520 | 0.6446 |
| 6 | 0.0345 | -0.7246 | 0.0264 | -0.2712 | 0.6556 |
| 7 | 0.0222 | -0.7383 | 0.0273 | -02708 | 0.6741 |
| 8 | 0.0117 | -0.7430 | 0.0354 | -0.2779 | 0.6788 |
| 9 | 0.0076 | -0.7478 | 0.0362 | -0.2773 | 0.6851 |

ans = -0.7478 0.0362 -0.2773 0.6851

1. Método de Gauss sidel

* **ALGORITMO**
* **CODIGO**

function [x M]=Gausseidel(A,b,x0,er)

A=input('ingresa tu matriz cuadrada A=')

N=length(A)

b=input('ingresa tus coordenadas b=')

b=b'

x0=input('ingresa tu punto inicial x0=')

x0=x0'

er=input('ingresa error er=')

G=[];

M=A

fori=1:N-1

A(i,i+1:N)=0;

end

G=A;

H=inver1(G);

R=eye(N)-H\*M;

P=H\*b;

R;

P;

error=1;

M=[0 , error, x0'];

s=0;

while error>er

x=R\*x0+P;

error=norm(x-x0)/norm(x);

x0=x;

s=s+1;

M=[M ;[s,error,x0'] ];

end

M

* **EJEMPLO**

Aplicar el método de Gaussseidelpara resolver el siguiente sistema

Con

**Gausseidel(A,[-2 -1 0 1]',[0 0 0 0]',0.01)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **k** | **xk1** | **xk2** | **xk3** | **xk4** | **ERROR** |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | -0.5 | -0.125 | -0.125 | 0.5 |
| 2 | 2.584 | -0.625 | 0 | -0.2250 | 0.6167 |
| 3 | 0.1067 | -0.7104 | 0.0255 | -0.2603 | 0.6654 |
| 4 | 0.0348 | -0.7378 | 0.0357 | -0.2735 | 0.6823 |
| 5 | 0.0125 | -0.7479 | 0.0392 | -0.2782 | 0.6884 |
| 6 | 0.0044 | -0.7515 | 0.0404 | -0.2799 | 0.6906 |

ans = -0.7515 0.0404 -0.2799 0.6906

**NOTA:** cómo podemos apreciar hemos aplicado el mismo ejemplo para los dos primeros métodos, y hemos sacado como conclusión que el método gaussidel es más efectivo ya que trabaja con menos iteraciones sin embargo existe el método del SOR, el cual combina estos dos métodos y lo hace mas efectivo veamos:

1. Método de Sorjacobi

* **ALGORITMO**
* **CODIGO**

function [y,N]=sorjacobi(A,b,x0,error,w)

A=input('ingresa tu matriz cuadrada A=')

N=length(A)

b=input('ingresa tus coordenadas b=')

b=b'

x0=input('ingresa tu punto inicial x0=')

x0=x0'

w=input('ingresa tu peso w=')

er=input('ingresa er=')

G=[];

M=A

fori=1:N-1

A(i,i+1:N)=0;

end

G=A;

H=inver1(G);

R=eye(N)-H\*M;

P=H\*b;

R;

P;

error=1;

M=[0 , error, x0'];

s=0;

while error>er

x=R\*x0+P;

error=norm(x-x0)/norm(x);

x0=x;

s=s+1;

M=[M ;[s,error,x0'] ];

end

M

er=1;

fori=1:N,

p=A(i,i);

b(i)=b(i)/p;

for j=1:N

A(i,j)=A(i,j)/p;

end

end

A;

d=eye(N)-A;

j=0;

N=[0,x0',er];

whileer>error,

x=d\*x0+b;

y=w\*x+(1-w)\*x0;

er=norm(y-x0)/norm(y);

x0=y;

j=j+1;

N=[N;j,y',er];

end

N

* **EJEMPLO**

Aplicar el método de sorjacobi para resolver el siguiente sistema

Con

sorjacobi(A,[-2 -1 0 1]',[0 0 0 0]',0.01,1.1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **k** | **xk1** | **xk2** | **xk3** | **xk4** | **ERROR** |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | -0.55 | -0.2750 | 0 | 0.3667 | 1 |
| 2 | -0.5252 | 0.0046 | -0.2622 | 0.4308 | 0.5379 |
| 3 | -0.6898 | -0.0860 | -0.1820 | 0.6121 | 0.2926 |
| 4 | -0.6758 | 0.0416 | -0.2872 | 0.5936 | 0.1767 |
| 5 | -0.7361 | 0.0091 | -0.2414 | 0.6756 | 0.1193 |
| 6 | -0.7261 | 0.0477 | -0.2884 | 0.6542 | 0.0759 |
| 7 | -0.7497 | 0.0205 | -0.2643 | 0.6907 | 0.0539 |
| 8 | -0.433 | 0.0464 | -0.2860 | 0.6769 | 0.0354 |
| 9 | -0.7532 | 0.0323 | -0.2736 | 0.6934 | 0.0253 |
| 10 | -0.7495 | 0.0443 | -0.2838 | 0.6857 | 0.0170 |
| 11 | -0.7538 | 0.0372 | -0.2776 | 0.6932 | 0.121 |
| 12 | -0.7518 | 0.0429 | -0.2824 | 0.6892 | 0.0082 |

ans = -0.7518 0.0429 -0.2824 0.6892

1. Método de Sorsidel

* **ALGORITMO**
* **CODIGO**

function [x0 M]=sorGausseidel(A,b,x0,error,w)

A=input('ingresa tu matriz cuadrada A=')

N=length(A)

b=input('ingresa tus coordenadas b=')

b=b'

x0=input('ingresa tu punto inicial x0=')

x0=x0'

w=input('ingresa tu peso w=')

error=input('ingresa error error=')

er=1;

fori=1:N

p=A(i,i);

if p~=0

A(i,:)=A(i,:)/p;

b(i)=b(i)/p;

else

disp('procesofallado')

return

end

end

D=eye(N)-A;

k=0;

M=[0,x0',er];

whileer>error,

y=x0;

fori=1:N

s=0;

for j=1:N

if i~=j

s=s+D(i,j)\*x0(j);

end

end

x0(i)=s+b(i);

end

z=w\*x0+(1-w)\*y;

er=norm(y-z)/norm(z);

y=z;

k=k+1;

M=[M;k,z',er];

end

M

* **EJEMPLO**

Aplicar el método de sorjacobi para resolver el siguiente sistema

Con

sorGausseidel(A,[-2 -1 0 1]',[0 0 0 0]',0.01,1.1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **k** | **xk1** | **xk2** | **xk3** | **xk4** | **ERROR** |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | -0.55 | -0.1375 | -0.1375 | 0.55 | 1 |
| 2 | -0.6375 | 0.0125 | -0.2350 | 0.6283 | 0.2784 |
| 3 | -0.7190 | 0.0281 | -0.2638 | 0.6703 | 0.1162 |
| 4 | -0.7406 | 0.0367 | -0.2748 | 0.6840 | 0.0382 |
| 5 | -0.7489 | 0.0395 | -0.2787 | 0.6890 | 0.0137 |
| 6 | -0.7518 | 0.0405 | 0.2801 | 0.6908 | 0.0048 |

ans = -0.7515 0.0404 -0.2799 0.6906

**NOTA:**ahora apreciamos en los dos métodos que dependen de un peso al cual le hemos designado con w, y nuevamente el proceso de Sor Gaussidel fue más efectivo que Sor Jacobi, este proceso de SOR en general es más efectivo que los dos anteriores (Jacobi y Sidel ).

1. Métodos para obtener auto valores

* **Metido de la potencia**
* **ALGORITMO**

|  |
| --- |
|  |

* **CODIGO**

function [h,y,m]=potencias(A,error)

N=length(A);

y=ones(N,1);

er=1; h1=1; m=[];

whileer>error

x=A\*y;

[h,k]=max(x);

y=x/h;

er=abs(h-h1)/abs(h);

h1=h;

m=[m;h,y',er]

end

* **EJEMPLO**

Aplicar el método de la potencia inversa para la siguiente matriz

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **k** | **hk+1** | **yk+1** | | | **ERROR** |
| 0 | 11 | 0.3636 | 0.5455 | 1.0000 | 0.9091 |
| 1 | 7.5455 | 0.2771 | 0.5060 | 1.0000 | 0.4578 |
| 2 | 7.1205 | 0.2572 | 0.5008 | 1.0000 | 0.0597 |
| 3 | 7.0305 | 0.2520 | 0.5001 | 1.0000 | 0.0128 |
| 4 | 7.0082 | 0.2506 | 0.5000 | 1.0000 | 0.0032 |

**NOTA:**hemos omitido hallar los zk+1, eso se lo puede observar cuando ejecutamos el programa. Tomemos en cuenta que yk+1 es un vector columna en ese caso de R3. Este programa nos ayuda a encontrar el mayor autovalor real de una matriz.

* **Metodo de la potencia inversa**
* **CODIGO**

function [h,y,m]=potenciainversa(A,error)

N=length(A);

y=ones(N,1);

er=1; h1=1; m=[];

whileer>error

x=elimagauspivo1(A,y);

[h,k]=max(x);

y=x/h;

er=abs(h-h1)/abs(h);

h1=h;

m=[m;h,y',er];

end

q=1/h;

m

* **EJEMPLO**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **k** | **qk+1** | | **yk+1** | | | | | |  | |
| 0 | 0.1667 | | 1 | | 1 | | 1 | | 5 | |
| 1 | 0.1667 | | 1 | | 1 | | 1 | | 0 | |
| 2 | 0.1667 | | 1 | | 1 | | 1 | | 0 | |
| …. | | | | | | | | | | |
| 42 | | 1 | | 1 | | 1 | | 1 | | 0 |

ans = 1

**NOTA:**pusimos ese error ya que con ese converge al valor indicado, se observa hallamos el menor autovalor.

* **Metodo de la potencia inversa con desplazamiento**
* **CODIGO**

function [k,y,m]=potenciainversadespl(A,q,error)

N=length(A);

y=ones(N,1);

A=A-q\*eye(N)

er=1; h1=1; m=[];

whileer>error

x=elimagauspivo1(A,y);

[h,k]=max(x);

y=x/h;

er=abs(h-h1)/abs(h);

h1=h;

m=[m;h,y',er];

end

k=1/h+q

m

* **EJEMPLO**

Aplicar el método de la potencia inversa con desplazamiento para la siguiente matriz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3.8998 | 1 | -0.5642 | -0.8436 | 0.7436 |
| 10.3140 | 1 | -0.4862 | -1.0030 | 0.6219 |
| 9.9391 | 1 | -0.5014 | -0.9994 | 0.0377 |
| 10.0063 | 1 | -0.4998 | -1.0001 | 0.0067 |
| 9.9993 | 1 | -0.5 | -1 | 0.0007 |

1. Método de la Gradiente Conjugada

* **ALGORITMO**

|  |
| --- |
|  |

* **CODIGO**

functionGc

Q=input('la matriz Q=')

b=input('ingresa el b=')

x0=input('ingresa el punto inicial x0=')

error=input('ingresa el error=')

m=[];

er=1; r0=b-Q\*x0; d0=r0; k=0;

whileer>error

h=r0'\*d0/(d0'\*Q\*d0);

x=x0+h\*d0;

r=r0-h\*Q\*d0;

B=(d0'\*Q\*r)/(d0'\*Q\*d0);

d=r-B\*d0;

er=norm(x-x0)/norm(x);

x0=x

r0=r;

d0=d;

k=k+1;

m=[m;k,h,B,x',er];

end

m

* **EJEMPLO**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| k | h | B | Xk | | | error |
| 1 | 0.0902 | -0.0033 | 0.9922 | 0.9922 | 0.0902 | 1 |
| 2 | 0.1003 | -0.0040 | 1.0045 | 0.9954 | 0.0009 | 0.0683 |
| 3 | 0.1128 | 0.0000 | 1.0000 | 1.0000 | 0.0000 | 0.0046 |
| 4 | 0.1026 | -0.0003 | 1.0000 | 1.0000 | 0.0000 | 0.0000 |