Consideremos el problema de hallar la deflexión, en 100 puntos equi−espaciados, de una placa cuadrada, de 2m x 2m y con un espesor de z = 10−2, que se encuentra simplemente apoyada en sus orillas.

La misma esta sometida a una carga q = 33,60 [KN/m2], tiene un módulo de elasticidad de = 2 * 1011 y la relación de Poisson es = 0,3.

La deflexión en la dimensión Z es regida por la E.D.P. elíptica:

z + 2 z + z = q

x4 x2 y2 y4 D

Donde D = z3 es la rigidez flexional.

12(1−)

El problema puede descomponerse en

1) U = z + z con z = 0 sobre el borde

x2 y2

2) u + u = q con u = 0 sobre el borde

x2 y2 D

Utilizando el método de diferencias finitas, aproximamos primero u = u(x,y) a partir de 2, y luego con el resultado, obtendremos una aproximación z para la deflexión z = z(x,y).

Tomamos 2 y aproximamos las segundas derivadas parciales en el punto de la malla (x, y); de esta forma obtenemos:

Uxx (x,y) " U(xi+h,yj) − 2U(xi,yj) + U(xi−h,yj)

h2

Uyy (x,y) " U(xi,yj+h) − 2U(xi,yj) + U(xi,yj−h)

h2

con xi = ih, i = 1, ..., 11

yj = jh, j = 1, ..., 11

h = 0,2

Por lo que a partir de la ecuación de Poisson: U(x,y) = f(x,y), se obtiene

Uxx (xi,yj) + Uyy (xi,yj) = f(x,y) " i,j, de donde se logra obtener el algoritmo de diferencias finitas:

Ui+1,j − 2Ui,j + Ui−1,j + Ui,j+1 − 2Uij + Ui,j−1 = q lo que es igual a:

h2 h2 D

Ui−1,j + Ui,j−1 − 4Ui,j +Ui,j+1 + Ui+1,j = qh2

D

De aquí se puede obtener un sistema lineal del tipo AX = b, donde el vector de las incógnitas es de dimensión 121. A es una matriz 121 x 121 compuesto por los factores de U, y b = [qh2/D], también es una matriz 1 x 121.

Por lo que A será la siguiente Matriz:

−4 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 .....0 u11 0

1 −4 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 .....0 u12 0

0 1 −4 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 .....0 u13 0

0 0 1 −4 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 .....0 u14 0

0 0 0 0 −4 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 .....0 u15 0

0 0 0 0 1 −4 1 0 0 0 0 0 0 0 0 1 0 0 0 0 .....0 u16 0

....

1 0 0 0 0 0 0 −4 1 0 0 0 0 0 0 0 0 0 1 0 ..... 0 uij qh2/D

0 1 0 0 0 0 0 1 −4 1 0 0 0 0 0 0 0 0 0 1 ..... 0 uij+1 qh2/D

....

....

....

0..... 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 −4 1 unn−1 0

0..... 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 −4 unn 0

Como se puede apreciar la condición de borde u = 0, hace que existan puntos donde el vector incógnita toma valores iguales a 0, los cuales pueden ser eliminados del sistema, para hacerla más pequeño y por tanto se necesitarán menos operaciones para resolverlo. Estos puntos son eliminados de la matriz A, ya que en estos mismos puntos z va a valer cero, por condición de borde antedicha, es decir la deflexión de la placa en sus bordes es igual a 0.

Así que resumiendo, nuestro problema se reduce a encontrar la solución del sistema:

A81x81 U81x1 = b81x1.

Cargando la matriz A y el vector B en Matlab, de la siguiente manera:

```
A=zeros(81);
for i=1:81
A(i,i)=−4;
resto = mod(i,9);
if resto == 0
posicion=i−1;
if posicion>0
A(i,posicion)=1;
end
posicion=i+9;
if posicion<82
A(i,posicion)=1;
end
posicion=i−9;
if posicion>0
A(i,posicion)=1;
end
end
if resto == 1
posicion=i−9;
if posicion>0
A(i,posicion)=1;
end
posicion=i+1;
if posicion<82
```

```
A(i,posicion)=1;
end
posicion=i+9;
if posicion<82
A(i,posicion)=1;
end
end
if resto > 1
posicion=i+9;
if posicion<82
A(i,posicion)=1;
end
posicion=i+1;
if posicion<82
A(i,posicion)=1;
end
posicion=i−1;
if posicion>0
A(i,posicion)=1;
end
posicion=i−9;
if posicion>0
A(i,posicion)=1;
end
end
end
```

```
B=[];

q=33.6;

sigma=0.3;

deltaz=0.01;

e=2*10^11;

h=0.2;

D=(e*deltaz^3)/(12*(1−sigma^2));

res=(q*h^2)/D;

for i=1:81

B(i)=res;
```

El Sistema Lineal podrá ser resuelto mediante 2 métodos distintos:

a) *Métodos Directos*, que son aquellos que con un número finito de pasos resuelven un S.L.. Y aquí se podrá utilizar el método de Gauss o el de Descomposición LU. Quedando el Algoritmo de Tomas excluido para la resolución de este problema, ya que la Matriz A no es tridiagonal.

Los métodos de descomposición LU y Gauss, también quedan excluido para resolver el S.L., ya que no son buenos para trabajar con matrices dispersas, como la de este problema, pues provocan que se pierdan muchos de los ceros que tiene la matriz A, por lo que el sistema se torna más complejo.

b) *Métodos Indirectos.* Estos no dan una solución exacta, pero dada una tolerancia, resuelven el S.L. con un error menor a la misma.

Aquí encontramos el Método de Jacobi y el de Gauss−Seidel

La idea de ambos es partir de un punto inicial e ir construyendo una sucesión de puntos que converja en la solución del sistema.

Los métodos indirectos son recomendados para resolver sistemas con matrices dispersas, por lo que a continuación compararemos el Método de Jacobi con el Método de Gauss−Seidel, para decidir en que método nos apoyaremos para llegar a la solución aproximada del S.L.

Para ello rescribimos la matriz A como tres matrices, tal que A = T+J+D. Donde T es la matriz triangular inferior, J la triangular superior y D la diagonal de A.

De aquí obtenemos que para Jacobi, X(m+1) = D−1 (b−(L+T)X(m), y para Gauss−Seidel X(m+1) = (D+T)−1 (b−JX(m)). La diferencia básica entre ambos es que Gauss−Seidel utiliza los valores más recientes apenas se obtienen, mientras que Jacobi no. Por lo que se puede esperar que Gauss converja más rápido que Jacobi, aunque esto no siempre es así.

La condición necesaria y suficiente para la convergencia de un método iterativo es que el radio espectral de la matriz de iteración Q sea menor que 1. Siendo QJ = D−1 (T+J) y QG−S = −(D+T)−1J.

Por otra parte, calcular el radio espectral de Q es sumamente costoso, por lo que no aplicaremos esta propiedad directamente.

El radio espectral no es otra cosa que el ínfimo de todas las normas de la matriz Q, por lo que será condición suficiente para la convergencia que la norma sea menor que 1.

normag = norm(Q); Esta es la norma de QGS y su resultado es: 0.5971.

normaj = norm(Qj); Esta es la norma de QJ y su resultado es: 0.95106.

La norma de la matriz de iteración de Gauss−Seidel es menor que la de Jacobi, por lo tanto, es de esperar que el primero converja más rápidamente.

Por lo que utilizaremos este método para resolver el S.L

Primero debemos resolver el primer S.L. con vector de incógnitas U, luego de hallado el mismo, este se volverá el vector solución y volvemos a correr Gauss Seidel para hallar el Vector de Incógnitas S, que es la solución final de este sistema.

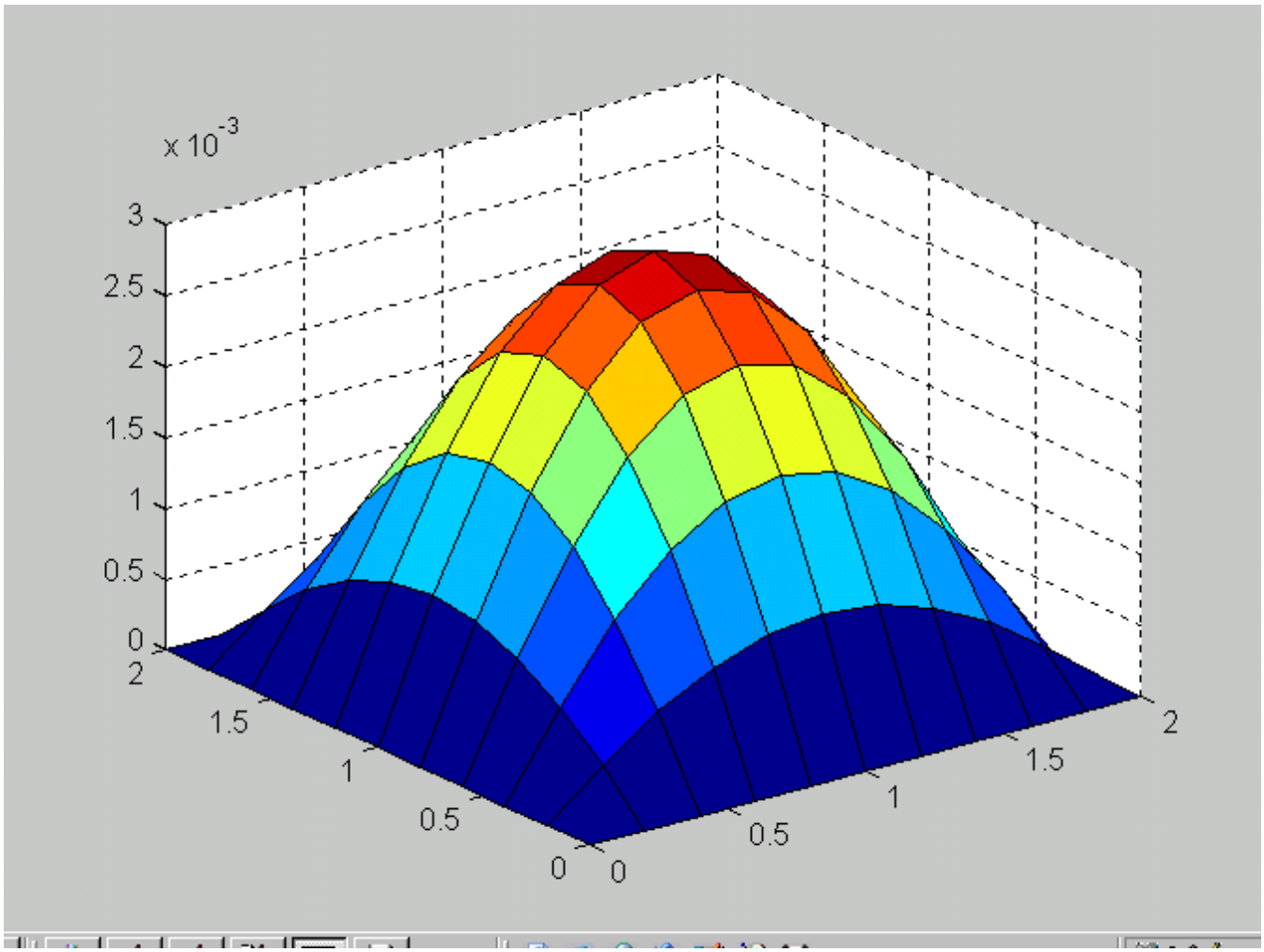
(Por comodidad nos referiremos a los efectos de explicar el trabajo realizado, de aquí en más, a los vectores incógnita como X, tanto si se trata de U o de Z.)

Los pasos realizados para llegar a las soluciones son:

```
while (error>tol)

yanterior = U;

for i=1:81;

y(i)=(L(i,1:81)*U+B(i))./d(i);

U(i)=y(i); %el nuevo valor calculado ya lo sustituyo en el anterior

end
```

Donde realizando el cambio de y(i) por z(i), y pasando el vector U a ser nuestro vector de solución, obtenemos los valores solución de z expresados en el vector S.

Si graficamos las soluciones obtenidas en el Vector S, obtenemos la siguiente gráfica:



Sabemos que Norm(XK+1 − X) " (Norm (Q)/1−Norm(Q)) (Norm(Xk+1 − XK)

Como tenemos Condición de Parada (Norm(Xk+1 − XK) < tol, entonces podemos decir que la Norm(Xk+1 − X) " (Norm(Q)/1−Norm(Q)) * tol. Es decir que la distancia a la solución real (X) está acotada por la norma de Q y la tolerancia. Como la norma de Q es un valor constante, lo que va a determinar la precisión de la solución será la tolerancia.

Nosotros consideramos que una tolerancia de 10−10, es una buena aproximación ya que como veremos más adelante, la misma permite ver que la solución converge a la solución verdadera con un tiempo de proceso aceptable .

Por otra parte también hallamos el Número de Condición de la Matriz A (K), el mismo nos permite saber si una matriz es singular o se aproxima a ser singular y como pequeñas perturbaciones relativas A o B afectan a el vector de resultados X.

En este caso podemos considerar que K es grande, ya que su valor es de 39.863 , y esto nos muestra que pequeñas variaciones relativas en B provocan grandes perturbaciones relativas en X. Esto está dado por la siguiente desigualdad:

Norm(x)/Norm(x) " K(A)*(Norm(B)/Norm(B))

En este caso B es un dato, conocido y sin ningún tipo de error o variación, así que lo único que lo afecta es su ingreso, a fines de resolver el sistema, a un ordenador. Esto provoca que se den errores de redondeo en sus

expresiones, ya que las mismas no cuentan con una representación exacta para la máquina. Por tal razón se puede decir que el error relativo de B es igual a epsilon machine.

Por tanto se puede inferir que el error residual de X es menor igual que K(A) por el epsilon machine, y como el epsilon machine es pequeño (2.204e−16) y K es 39.863, el error relativo estará acotado por un número pequeño (8.8515e−15). Esto junto con el hecho de que el error de la solución converge a cero, nos permite decir que el problema se encuentra bien condicionado. Y por tanto que la convergencia no se ve afectada por problemas de condicionamiento.

Antes de pasar continuar hablando de errores, vamos a determinar cual es el orden de operaciones para resolver el sistema.

Como mencionamos al inicio, los métodos iterativos no resuelven los sistemas en un número finito de pasos dados constantes, ya que la cantidad de pasos que conlleva la resolución depende de la cantidad de Iteraciones necesarias para resolverlo, y estas dependerán de la tolerancia que se establezca. De todas maneras podemos decir que el orden de operaciones por iteración es de n2, en este caso en particular sería de 812 es decir que las operaciones por iteración fueron 6561.

Ahora el orden de operaciones totales es de N2*p, siendo p en este caso igual a 166, por lo que el total de operaciones realizadas fue de 812*166 .

Ahora nos adentraremos en el tema de los errores.

Para comenzar estableceremos que desde el momento que utilizamos el método de diferencias finitas para obtener las consecutivas aproximaciones de U y Z, ya introdujimos un error de truncamiento del orden de h2, para cada aproximación. También existe un error de redondeo. Este se genera por el hecho de que el ordenador puede ingresar dos clases de números, los integrales (números enteros) y los puntos flotantes (números reales). Estos representados por un número finito de dígitos que estarán determinados por el tamaño de la mantisa, los números que excedan dicho tamaño serán desechados por el ordenador, ya sea truncando el número o redondeándolo.

Por ejemplo, si la mantisa es 2, y queremos representar un número entre 0.14 y 0.15, la cota superior del error de redondeo será 0.005 y la cota superior del error por truncamiento será de 0.01.

Como vimos arriba, el error cometido por el método dependerá, aparte de los errores recién explicados, de la tolerancia que le explicitemos al programa como condición de parada. A menor tolerancia, mayor será la aproximación a la solución verdera.

Existen varias maneras de ver cual es el error cometido al hallar las soluciones del sistema. La más que suponemos de manera natural es el vector residual r. Donde r es igual a (b−Ax*), siendo x* el vector de soluciones computadas. Es natural suponer que si r es pequeño, entonces x es una buena solución aproximada. Pero esto no siempre es así, si el sistema esta mal condicionado o X es sensible a pequeñas perturbaciones en A o en b, r puede ser pequeño y X puede estar muy lejos de las soluciones correctas. Por tal razón es que se calculó el número de condición, y como demostramos antes, se podrán obtener buenas aproximaciones de la solución.

Volviendo a r, el cálculo del mismo es costoso pues debemos hallar el producto de una matriz por un vector y luego restarle otro vector. Por lo que no tomaremos este método para hallar los errores de U y de Z.

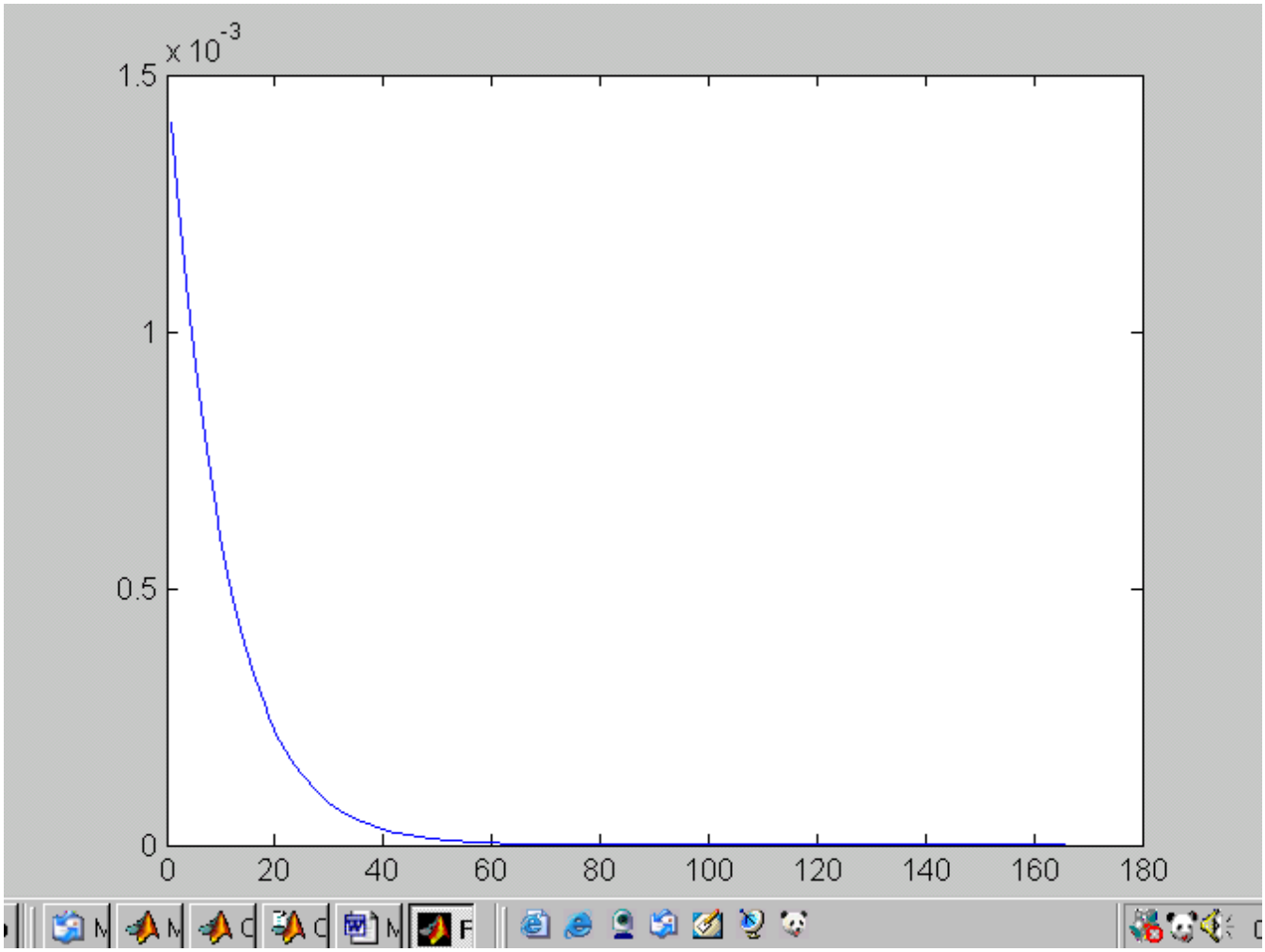
Pero podemos tomar como error la norma de (Xk+1 − Xk) , ya que como vimos la misma se encuentra acotada. Por lo que podemos establecer que el error dependerá de la tolerancia que se determine.

Para hallar este error ingresamos al programa los siguientes comandos:

```
error=norm(z−zanterior);

S=z;

p=p+1;

errorz(p) = error;

end
```

Siendo errorz(p) el vector de errores en las p iteraciones.

A efectos de este trabajo, nos alcanza con mostrar que el error va disminuyendo de iteración en iteración, como se puede apreciar en la gráfica siguiente:



Con esto podemos ver que el error tiende a cero y que luego de la iteración 60 el error es muy pequeño. De todas maneras no disminuimos la tolerancia, para que se pueda apreciar como el error no sufre variaciones y como se vuelve asintótico al eje de las p (iteraciones).

Otra forma de ver el error es mostrando el error residual, es decir, la norma de X sobre la norma de X, pero como ya vimos esto se encuentra acotado por el número de condición de A por el error residual de b (que en

este caso lo tomamos como el epsilon machine), y como demostramos que el sistema no esta mal condicionado, podemos inferir que también decrece por lo que no agrega nada nuevo el calcularlo o graficarlo.

Comentario: existe un un parámetro de relajación ( ), que puede es escogido de manera tal que el ratio de convergencia es maximizado.

En este caso la matriz de iteración Q estaría dada por (I+ L)−1((1−)I− U). , siendo únicamente de interés los valores de que cumplan 0< <2. Y cuando = 1 el método de sobrerelajación sucesiva se reduce al método de Gauss−Seidel.

Por razones de tiempo esto no es expuesto en el trabajo, pero consideramos que era importante resaltar que existe un punto (que puede ser determinado) dónde la convergencia puede ser maximizada.