Universidad Nacional Autónoma de Honduras

Facultad de Ciencias

Centro de Innovación en Cómputo Científico (CICC)

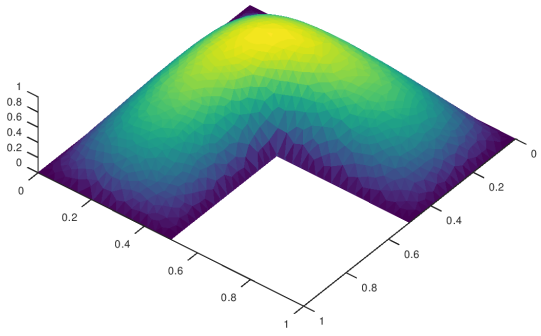
Lecturas del Curso:

Introducción al Cómputo Científico con Octave/MATLAB

Fredy Vides
Centro de Innovación en Cómputo Científico
Departamento de Matemática Aplicada
Escuela de Matemática y Ciencias de la Computación
Universidad Nacional Autónoma de Honduras

E-mail: fredy.vides@unah.edu.hn

October 8, 2018



Contents

Presentación

Este manual ha sido diseñado para guiar al lector en el aprendizaje de las técnicas fundamentales más modermas del cómputo científico, utilizando como herramientas computacionales los programas GNU Octave en su versión 4.4 y MATLAB en su versión R2017b.
A la vez, este documento ha sido pensado para servir como guía para el curso de **Introducción al Cómputo Científico**, que forma parte de las actividades de capacitación continua del **Centro de Innovación en Cómputo Científico (CICC)** de la Facultad de Ciencias de la Universidad Nacional Autónoma de Honduras.

1 Preliminares de Cómputo Matricial Básico

1.1 Operaciones y funciones matriciales

Consideremos las siguientes matrices:

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \tag{1}$$

$$B = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{2}$$

$$C = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{3}$$

$$D = \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4}$$

$$v = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{5}$$

Los comandos necesarios para ingresar las matrices anteriores en Octave/MATLAB son los siguientes:

```
>> A=[2,-1,0;-1,2,-1;0,-1,2]
A =

   2  -1   0
  -1   2  -1
   0  -1   2

>> B=[0,-1,1;1,0,0;0,1,0];
>> C=[0,0,1;1,0,0;0,1,0];
>> D=[0,-1;1,0;0,1];
>> v=[0;1;0];
```

1.1.1 Operaciones con matrices

Mostraremos a continuación los procedimientos computacionales correspondientes a algunas operaciones algebraicas fundamentales.

- $R_1 = A + B$:

```
>> R1=A+B
R1 =

   2  -2   1
   0   2  -1
   0   0   2
```

- $R_2 = 7A$:

```
>> R2=7*A
R2 =

   14   -7    0
   -7   14   -7
    0   -7   14
```

- $R_3 = B^\top$

```
>> R3=B.'
R3 =

   0   1   0
  -1   0   1
   1   0   0
```

- $R_4 = (B + iC)^*$

```
>> R4=(B+i*C)'
R4 =

   0 - 0i   1 - 1i   0 - 0i
  -1 - 0i   0 - 0i   1 - 1i
   1 - 1i   0 - 0i   0 - 0i
```

- $R_5 = BC$

```
>> R5=B*C
R5=B*C
R5 =

  -1   1   0
   0   0   1
   1   0   0
```

- $R_6 = \mathbf{1}_3$

```
>> R6=eye(3)
R6 =

Diagonal Matrix

   1   0   0
   0   1   0
   0   0   1
```

- $R_7 = a_{12}$

```
>> R7=A(1,2)
R7 = -1
```

- $R_8 = [a_{12}\ a_{13}]$

```
>> R8=A(1,2:3)
R8 =

  -1   0
```

- $R_9 = A_2 = [a_{21}\ a_{22}\ a_{33}]$

```
>> R9=A(2,:)
R9 =

  -1   2  -1
```

- $R_{10} = \det(A)$

```
>> R10=det(A)
R10 =  4.0000
```

- $R_{11} = A^{-1}$

```
>> R11=inv(A)
R11 =

   0.75000   0.50000   0.25000
   0.50000   1.00000   0.50000
   0.25000   0.50000   0.75000
```

- $R_{12} = \mathrm{tr}(-2D^*D)$

```
>> R12=trace(-2*D'*D)
R12 = -6
```

- $R_{13} = \langle A^3, v \rangle$

```
>> R13=A(:,3)'*v
R13 = -1
```

- $R_{14} = A \circ B = [a_{ij} b_{ij}]$

```
>> R14=A.*B
R14 =

   0   1   0
  -1   0  -0
   0  -1   0
```

- $R_{15} = A \otimes B = [a_{ij} b_{kl}]$

```
>> R15=kron(A,B)
R15 =

   0  -2   2  -0   1  -1   0  -0   0
   2   0   0  -1  -0  -0   0   0   0
   0   2   0  -0  -1  -0   0   0   0
  -0   1  -1   0  -2   2  -0   1  -1
  -1  -0  -0   2   0   0  -1  -0  -0
  -0  -1  -0   0   2   0  -0  -1  -0
   0  -0   0  -0   1  -1   0  -2   2
   0   0   0  -1  -0  -0   2   0   0
   0   0   0  -0  -1  -0   0   2   0
```

- $R_{16} = AB^{-1}$

```
>> R16=A/B
R16 =

  -0   2  -1
  -1  -1   1
   2   0   1
```

- $R_{17} = B^{-1}A$

```
>> R17=B\A
R17 =

  -1   2  -1
  -0  -1   2
   2  -2   2
```

- Descomposición espectral (en autovalores) $A = P\Lambda P^{-1}$:

```
>> [P,Lambda]=eig(A);
>> A-P*Lambda/P
ans =

   6.6613e-16   2.2204e-16  -1.1102e-16
  -4.4409e-16  -4.4409e-16   0.0000e+00
  -1.8645e-16   4.4409e-16  -4.4409e-16
```

- Descomposición DVS en valores singulares $A = U\Sigma V^*$:

```
>> [U,Sigma,V]=svd(A);
>> A-U*Sigma*V'
ans =

   4.4409e-16   2.2204e-16   6.9797e-17
   2.2204e-16   6.6613e-16   8.8818e-16
  -6.7008e-17  -6.6613e-16  -4.4409e-16
```

- Definir una matriz cero de 4×3:

```
>> Z=zeros(4,3)
Z =

   0   0   0
   0   0   0
   0   0   0
   0   0   0
```

- Definir una matriz de unos de 3×5:

```
>> O=ones(3,5)
O =

   1   1   1   1   1
   1   1   1   1   1
   1   1   1   1   1
```

- Consideremos nuevamente la matriz A anteriormente ingresada:
  - Extraer la diagonal principal (diagonal 0) de A:

```
>> D0=diag(A)
D0 =

   2
   2
   2
```

– Definir una matriz diagonal $D_A \in \mathbb{C}^{3\times 3}$ con la misma diagonal 0 que A:

```
>> DA=diag(diag(A))
DA =

Diagonal Matrix

   2   0   0
   0   2   0
   0   0   2
```

– Extraer las diagonales $1, -1, -2$ de A:

```
>> D1=diag(A,1)
D1 =

  -1
  -1

>> D_1=diag(A,-1)
D_1 =

  -1
  -1

>> D_2=diag(A,-2)
D_2 = 0
```

– Definir una matriz $K \in \mathbb{C}^{4\times 4}$ cuya diagonal 2 es igual a la diagonal -1 de A y todas sus demás entradas son cero:

```
>> K=diag(diag(A,-1),2)
K =

   0   0  -1   0
   0   0   0  -1
   0   0   0   0
   0   0   0   0
```

Recomendación: Para obtener más información sobre el funcionamiento de cualquier comando estándar de Octave/MATLAB basta escribir:

```
>> help nombre_del_comando
```

en la ventana de comandos.

1.1.2 Descomposiciones y funciones matriciales

Consideremos la representación vectorial del polinomio $p(z) = z^7 - 1$ en términos de sus coeficientes $\mathbf{p} = [p_7\ p_6\ \cdots\ p_1\ p_0] = [1\ 0\ 0\ 0\ 0\ 0\ 0\ -1]$, podemos ingresarlo en el sistema como:

```
>> p=[1,zeros(1,6),-1]
p =

   1   0   0   0   0   0   0  -1
```

Para calcular las raíces de $p(z)$ podemos utlizar la secuencia de comandos:

```
>> r=roots (p)
r =

  -0.90097 + 0.43388i
  -0.90097 - 0.43388i
  -0.22252 + 0.97493i
  -0.22252 - 0.97493i
   1.00000 + 0.00000i
   0.62349 + 0.78183i
   0.62349 - 0.78183i
```

Es posible evaluar p en un elemento cualquiera de su dominio utilizando el comando `polyval`, en particular la expresión:

```
>> polyval(p,-1)
ans = -2
```

permite calcular el valor $p(-1)$. La evaluación de un polinomio puede llevarse a cabo también en un rango de valores, por ejemplo, la expresión:

```
>> t=-1:1/4:1;
>> polyval(p,t)
ans =

 Columns 1 through 8:

  -2.00000  -1.13348  -1.00781  -1.00006  -1.00000  -0.99994  -0.99219  -0.86652

 Column 9:

   0.00000
```

calcula los valores $p(x_k)$ correspondientes a la partición $1 = t_0 < t_1 < \cdots < t_9 = 1$ del intervalo $[0, 1]$, donde $t_k = -1 + k * h$, $h = 1/4$.
Podemos ahora calcular la matriz compañera $C(p) \in \mathbb{C}^{7\times 7}$ de $p(z)$.

```
>> Cp=fliplr(flipud(compan(p)))
Cp =

   0   1   0   0   0   0   0
   0   0   1   0   0   0   0
   0   0   0   1   0   0   0
   0   0   0   0   1   0   0
   0   0   0   0   0   1   0
   0   0   0   0   0   0   1
   1  -0  -0  -0  -0  -0  -0
```

La teoría de matrices compañeras de polinomios nos asegura que dada una raíz r de $p(z)$, $r \in \sigma(C(p)^\top)$ (r es un autovalor de $C(p)^\top$) con autovector correspondiente $v_r = [1\ r\ r^2\ \cdots\ r^{n-1}]^\top$, es posible verificar esto en particular para r_1, con las siguiente secuencia de comandos:

```
>> Cpt=Cp.';
>> v1=r(1).^(0:6)';
>> norm(Cpt*v1-r(1)*v1)
ans =    4.7018e-15
```

donde `norm(v)` es el comando utilizado para calcular el valor $\|v\|$ para $v \in \mathbb{C}^{n\times m}$. Podemos ahora calcular las matrices de diagonalización P y Q de $C(p)^\top$ y $C(p)$, respectivamente, utlizando matrices de Vandermonde con la siguiente secuencia de comandos:

```
>> P=fliplr(vander (r).');
>> Q=P.';
>> norm(diag(diag(P'*Cpt*P))-P'*Cpt*P)
ans =    1.4378e-14
>> norm(diag(diag(Q*Cp*Q'))-Q*Cp*Q')
ans =    1.4414e-14
```

De forma alternativa podemos calcular las matrices de diagonalización P y Q, los valores propios de $C(p)^\top$ y $C(p)$ utilizando las siguientes secuencias de comandos basadas en el comando `eig(A)` de Octave/MATLAB que calcula la descomposición espectral de $A \in \mathbb{C}^{n\times n}$:

```
>> [Q,l]=eig(Cp);
>> [P,lt]=eig(Cpt);
>> [Q,l]=eig(Cp);
>> norm(diag(diag(P'*Cpt*P))-P'*Cpt*P)
ans =    2.6413e-15
>> norm(diag(diag(Q'*Cp*Q))-Q'*Cp*Q)
ans =    2.6672e-15
```

1.1.3 Lemniscatas y Pseudo-espectros

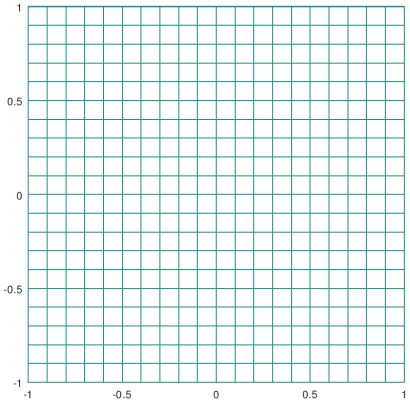
Consideremos nuevamente el polinomio de la sección anterior, una vez ingresado en el sistema con los comandos previos, podemos calcular las Lemniscatas de

$p(z)$ para localizar sus raíces computacionalmente utilizando el siguiente procedimiento.

Cálculo de Lemniscata de p(z):

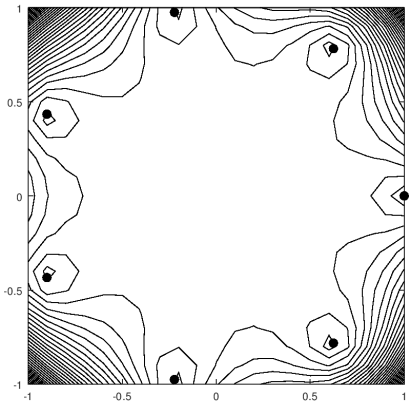
- Creación de malla inicial de la región $[-1,1]^2$ determinada por el Teorema de Gerschgorin aplicado a $C(p)^\top$:

```
>> [x,y]=meshgrid (-1:2/20:1);
>> mesh(x,y,zeros(size(x)));
>> view (2)
>> axis square
```



- Cálculo de Lemniscatas de $p(z)$ en la malla inicial:

```
>> p=[1,zeros(1,6),-1];
>> r=roots (p);
>> contour(x,y,abs(polyval(p,x+i*y)),32,'k')
>> hold on
>> plot(r,'k.','markersize',25)
>> axis square
```

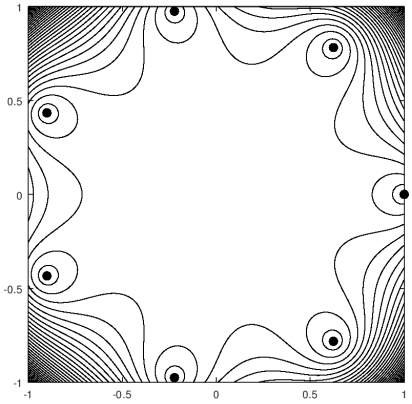


- Refinamiento de malla incial:

```
>> [x,y]=meshgrid (-1:2/100:1);
```

- Cálculo de Lemniscatas de $p(z)$ en la malla refinada:

```
>> close all
>> contour(x,y,abs(polyval(p,x+i*y)),32,'k')
>> hold on
>> plot(r,'k.','markersize',25)
>> axis square
```



El cálculo de Lemniscatas es importante para estudiar la estabilidad de sistemas dinámicos y de control discretos.
Definición: Dado $\varepsilon \geq 0$, se define el ε-Pseudo-espectro $\Lambda_\varepsilon(A)$ de $A \in \mathbb{C}^{n\times n}$ como el conjunto:

$$\Lambda_\varepsilon(A) = \left\{\lambda \in \mathbb{C} | \exists x \in \mathbb{C}^n \backslash \{0\}, E \in \mathbb{C}^{n\times n} : (A+E)x = \lambda x, \|E\| \leq \varepsilon\right\}$$

Podemos además calcular el Pseudo-espectro de $C(p)$ utilizando la siguiente secuencia de comandos basada en el comando `svd(A)` que calcula la descomposición en valores sigulares de A.

Cálculo de Pseudo-espectro de $C(p)$:

- Cálculo de malla de $[-1,1]^2$:

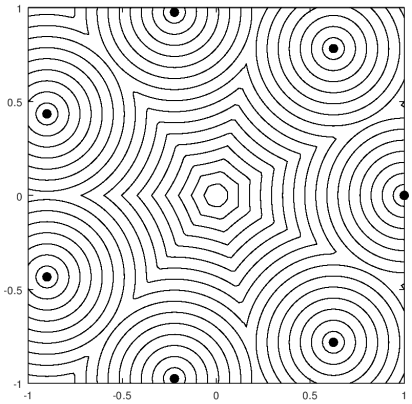
```
>> [x,y]=meshgrid (-1:2/100:1);
```

- Definición de la función `pspectra`:

```
>> pspectra=@(X)min(svd(X));
```

- Cómputo del Pseudo-espectro de $C(p)$:

```
>> Cp=fliplr(flipud(compan(p)));
>> N=size(x,1);
>> for k=1:N,for j=1:N,Z(k,j)=pspectra (Cp-(x(k,j)+i*y(k,j))*eye(7));end;end
>> contour(x,y,Z,16,'k')
>> l=eig(Cp);
>> hold on;
>> plot(l,'k.','markersize',25)
>> axis square
```



Al igual que el cómputo de Lemiscatas, el cómputo de Pseudo-espectros es importante para estudiar el compartamiento de sistemas dinámicos y de control discretos, importantes en la simulación numérica de procesos en ingeniería y ciencias.

1.2 Sistemas de Ecuaciones Lineales

En esta sección trabajaremos con algunas herramientas básicas de los programas orientados a matrices Matlab y Octave. Consideremos las siguientes matrices:

$$A = \begin{bmatrix} a & b & & e & b \\ b & a & & & \\ & & c & d & \\ e & & d & c & \\ b & & & & a \end{bmatrix} \tag{6}$$

$$f = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{bmatrix} \tag{7}$$

Donde supondremos que $a, b, c, d, e, b_1, ..., b_4 \sim N(0, 1)$. Tenemos que para ingresar las matrices anteriores en Matlab/Octave basta escribir:

```
>> a=randn;
>> b=randn;
>> c=randn;
>> d=randn;
>> e=randn;
>> A=[a b 0 e b;b a 0 0 0;0 0 c d 0;e 0 d c 0;b 0 0 0 a]
A =
-0.90411  -0.45550   0.00000  -1.63458  -0.45550
-0.45550  -0.90411   0.00000   0.00000   0.00000
 0.00000   0.00000   0.66103   0.25664   0.00000
-1.63458   0.00000   0.25664   0.66103   0.00000
-0.45550   0.00000   0.00000   0.00000  -0.90411
>> f=randn(5,1)
f =
  -1.49668
  -0.18249
  -0.44852
  -0.59657
   0.28638
>>
```

Para resolver el sistema de ecuaciones lineales

$$Ax = f$$

donde A y b estan dados por (6) y (7) respectivamente, basta con ejecutar la secuencia de comandos.

```
>> x=A\f
x =
   0.533964
  -0.067173
  -0.989978
   0.802238
  -0.585767
>>
```

2 Modelos Discretos en Diferencias Finitas y Aproximación de Funciones

2.1 Solución Numérica de Ecuaciones Diferenciales por Diferencias Finitas

2.1.1 Problemas de Valor de Frontera y Diferencias Finitas:

Estudiar la solución de las siguientes formas generales de ecuaciones diferenciales utilizando diferencias finitas, desarrollando un algoritmo computacional e implementando el mismo en Octave/SciLab.

PROBLEMA GENERAL 1D.

$$\begin{cases} -\frac{d^2u}{dx^2} = f(x), x \in [0,1] \\ u(0) = u(1) = 0 \end{cases}$$

Solución:

Algoritmo de Solución para el PROBLEMA GENERAL 1D:

1. Consideremos las fórmulas de tipo $O(h^2)$ estudiadas en clase de la forma:

$$\frac{d^2u}{dx^2}(x) \approx \frac{1}{h^2}(u(x+h) - 2u(x) + u(x-h))$$

2. Consideremos la partición de $[0,1]$ definida por:

$$0 = x_0 < x_1 < x_2 < \cdots < x_{N-1} < x_N = 1$$

donde $x_k = kh$ para $h = 1/N$ y $0 \leq k \leq N$.

3. Definamos $u_k := u(x_k) = u(kh)$, $0 \leq k \leq N$. Tenemos que $u_0 = u(0) = 0$, $u_N = u(x_N) = u(1) = 0$. Aplicando los pasos [1] y [2] podemos obtener una formulación discreta del problema general de la forma:

$$-\frac{1}{h^2}(u_{k-1} - 2u_k + u_{k+1}) = f(x_k), 1 \leq k \leq N-1.$$

4. Podemos ahora utlizar el paso [3] para obtener un sistema de ecuaciones lineales de la forma:

$$\mathbf{L}_2\mathbf{u} = \mathbf{f},$$

donde $\mathbf{u} = [u_1, u_2, \ldots, u_{N-1}]^\top$, $\mathbf{f} = [f(x_1), f(x_2), \ldots, f(x_{N-1})]^\top$ y donde $\mathbf{L}_2$ es la matriz de coeficientes dada por la ecuación:

$$\mathbf{L}_2 = -\frac{1}{h^2}\begin{bmatrix} -2 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ldots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & -2 & 1 \\ 0 & \cdots & 0 & 0 & 0 & 1 & -2 \end{bmatrix}$$

5. Resolvemos el sistema del paso [4] utilizando un procedimiento nuérico de solución de sistemas de ecuaciones.

6. Representamos las solución exacta y aproximada de la ecuacón en forma general en un gráfico combinado. Calculamos y representamos el error relativo.

PRÁCTICA 1.
Resolver la siguiente ecuación implementando el algoritmo de solución para el problema general 1D:

$$\begin{cases} -\frac{d^2u}{dx^2} = 8, x \in [0,1] \\ u(0) = u(1) = 0 \end{cases}$$

SOLUCIÓN:
Implementaremos el algoritmo de solución utilizando Octave:
Construimos la malla/partición de $[0, 1]$:

```
>> N=1000;
>> h=1/N;
>> x=0:h:1;
```

Construimos la matriz $\mathbf{L}_2$ de coeficientes del sistema:

```
>> L2=spdiags (ones(N-1,1)*[1 -2 1],-1:1,N-1,N-1);
>> L2=-1/h^2*L2;
```

Construimos el vector de forzamiento $\mathbf{f}$:

```
>> f=8*ones((N-1),1);
```

Resolvemos el sistem $\mathbf{L}_2\mathbf{u} = \mathbf{f}$:

```
>> u=L2\f;
```

Definimos y calculamos la solución exacta U_e y la solución aproximada U, incorporando las condiciones de frontera $u(0) = u(1) = 0$ del problema original.

```
>> U=zeros(N+1,1);
>> U(2:N)=u;
>> Ue=4*x.*(1-x);
>> plot(x,Ue,x,U,'g.')
>> plot(x,abs(U-Ue'))
```

Calculamos el error absoluto entre U_e y U:

```
>> norm(U-Ue',inf)
ans =    2.5380e-13
```

PROBLEMA GENERAL 2D.

$$\begin{cases} -(\partial_x^2 u + \partial_y^2 u) = f(x,y), (x,y) \in [0,1]^2 \\ u(x,y) = 0, (x,y) \in \partial[0,1]^2 \end{cases}$$

Solución:

Algoritmo de Solución para el PROBLEMA GENERAL 2D:

1. Consideremos las fórmulas de tipo $O(h^2)$ estudiadas en clase de la forma:

$$\begin{aligned} (\partial_x^2 + \partial_y^2)u(x,y) &\approx \frac{1}{h^2}(u(x+h,y) - 2u(x,y) + u(x-h,y)) \\ &+ \frac{1}{h^2}(u(x,y+h) - 2u(x,y) + u(x,y-h)) \\ &\approx \frac{1}{h^2}(u(x-h,y) + u(x,y-h) - 4u(x,y) + u(x+h,y) + u(x,y+h)) \end{aligned}$$

2. Consideremos la partición de $[0,1]^2$ definida por los pares:

$$(x_k, y_j) \in [0,1]$$

donde $x_k = kh$, $y_j = jh$ para $h = 1/N$ y $0 \leq k, j \leq N$.

3. Definamos $u_{k,j} := u(x_k, y_j) = u(kh, jh)$, $0 \leq k \leq N$. Tenemos que $u_{k,j} = 0$, para $k, j \in 0, 1$. Aplicando los pasos [1] y [2] podemos obtener una formulación discreta del Problema General 2D de la forma:

$$\frac{1}{h^2}(u_{k-1,j} + u_{k,j-1} - 4u_{k,j} + u_{k+1,j} + u_{k,j+1}) = f(x_k, y_j), 1 \leq k, j \leq N-1.$$

4. Podemos ahora utlizar el paso [3] para obtener un sistema de ecuaciones lineales de la forma:

$$\mathbf{L}_{2D}\mathbf{u} = \mathbf{f},$$

donde $\mathbf{u} = [u_{1,1}, u_{1,2}, \ldots, u_{N-1,N-1}]^\top$, $\mathbf{f} = [f(x_1, y_1), f(x_1, y_2), \ldots, f(x_{N-1}, y_{N-1})]^\top$ y donde $\mathbf{L}_{2D}$ es la matriz de coeficientes dada por la ecuación:

$$\mathbf{L}_{2D} = \mathbf{I}_{N-1} \otimes \mathbf{L}_2 + \mathbf{I}_{N-1} \otimes \mathbf{L}_2,$$

para $\mathbf{L}_2$ definida por la expresión:

$$\mathbf{L}_2 = -\frac{1}{h^2}\begin{bmatrix} -2 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \cdots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & -2 & 1 \\ 0 & \cdots & 0 & 0 & 0 & 1 & -2 \end{bmatrix}$$

5. Resolvemos el sistema del paso [4] utilizando un procedimiento nuérico de solución de sistemas de ecuaciones.

6. Representamos las solución exacta y aproximada de la ecuacón en forma general en un gráfico combinado. Calculamos y representamos el error relativo.

PRÁCTICA 1.
Resolver la siguiente ecuación implementando el algoritmo de solución para el problema general 1D:

$$\begin{cases} -(\partial_x^2 u + \partial_y^2 u) = 32x(1-x) + 32y(1-y), (x,y) \in [0,1]^2 \\ u(x,y) = 0, (x,y) \in \partial[0,1]^2 \end{cases}$$

SOLUCIÓN:
Implementaremos el algoritmo de solución utilizando Octave:
Construimos la malla/partición de $[0,1]^2$:

```
>> N=1000;
>> h=1/N;
>> x=0:h:1;
>> [X,Y]=meshgrid (x);
```

Construimos la matriz $\mathbf{L}_2$ de coeficientes del sistema:

```
>> E=speye(N-1);
>> L2=spdiags (ones(N-1,1)*[1 -2 1],-1:1,N-1,N-1);
>> L2=-1/h^2*L2;
>> L2=kron(E,L2)+kron(L2,E);
```

Construimos el vector de forzamiento $\mathbf{f}$:

```
>> Xi=X(2:N,2:N);
>> Yi=Y(2:N,2:N);
>> f=32*Xi.*(1-Xi)+32*Yi.*(1-Yi);
>> f=f(:);
```

Resolvemos el sistem $\mathbf{L}_2\mathbf{u} = \mathbf{f}$:

```
>> tic,u=L2\f;toc
Elapsed time is 112.165 seconds.
```

Definimos y calculamos la solución exacta U_e y la solución aproximada U, incorporando las condiciones de frontera $u(0) = u(1) = 0$ del problema original.

```
>> U=zeros(N+1,N+1);
>> U(2:N,2:N)=reshape(u,N-1,N-1);
>> Ue=16*X.*(1-X).*Y.*(1-Y);
>> surf(X,Y,Ue)
>> surf(X,Y,abs(Ue-U))
>> shading interp
```

Calculamos el error absoluto entre U_e y U:

```
>> norm(U(:)-Ue'(:),inf)
ans =    4.8561e-12
```

2.2 Series de Taylor e Inteporlación de Lagrange

2.3 Problemas de Aproximación de funciones:

Estudiar la solución de los siguientes problemas de aproximación de funciones.

PROBLEMA GENERAL 1. Dada una función analítica $f : \mathbb{R} \to \mathbb{R}$, escribiremos $T_a[f](x)$ para denotar el desarrollo de f en series de Taylor de la forma:

$$T_a[f](x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k$$

Escribimos $T_a^n[f](x)$ para denotar el polinomio de la forma:

$$T_a^n[f](x) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k = f(a) + f'(a)(x-a) + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n$$

La expresión $T_a^n[f](x)$ recibe el nombre de polinomio de Taylor de f cerca de a. Es posible ver que:

$$|T_a^n[f](x) - f| = O(|x-a|^{n+1})$$

cuando $x \approx a$.

PRÁCTICA 1. Sea $f(x) = 1 - x^2$. Calcular $T_0^2[f](x)$.
SOLUCIÓN:

$$T_0^2[f](x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 = 1 + 0x + \frac{-2}{2}x^2 = 1 - x^2$$

PROBLEMA GENERAL 2. Sea f una función determinada por sus valores $f(x_1), \ldots, f(x_n)$ en n puntos $x_1, \ldots, x_n$ en $\mathbb{R}$. Escribimos $L_n[f](x)$ para denotar el polinomio:

$$L_n[f](x) = \sum_{k=1}^{n} f(x_k)\ell_k(x)$$

donde cada $\ell_k(x)$ está definido por la ecuación

$$\ell_k(x) = \prod_{j=1, j\neq k}^{n} \frac{(x - x_j)}{(x_k - x_j)}$$

para cada $k = 1, 2, \cdots, n$. El polinomio $L_n[f](x)$ recibe el nombre de interpolante de Lagrange de f.

PRÁCTICA 1. Calcular $L_3[f](x)$ respecto de $x_1 = -1, x_2 = 0, x_3 = 1$, si $f(x) = 1 - x^2$. SOLUCIÓN: Tenemos que:

$$\ell_1(x) = \frac{(x - x_2)}{(x_1 - x_2)}\frac{(x - x_3)}{(x_1 - x_3)} = \frac{(x - 0)}{(-1 - 0)}\frac{(x - 1)}{(-1 - 1)} = \frac{1}{2}x(x - 1)$$

$$\ell_2(x) = \frac{(x - x_1)}{(x_2 - x_1)}\frac{(x - x_3)}{(x_2 - x_3)} = \frac{(x - (-1))}{(0 - (-1))}\frac{(x - 1)}{(0 - 1)} = 1 - x^2$$

$$\ell_3(x) = \frac{(x - x_1)}{(x_3 - x_1)}\frac{(x - x_2)}{(x_3 - x_2)} = \frac{(x - (-1))}{(1 - (-1))}\frac{(x - 0)}{(1 - 0)} = \frac{1}{2}x(x + 1)$$

Aplicando la fórmula para $L_3[f](x)$ obtenemos:

$$L_3[f](x) = f(-1)\ell_1(x) + f(0)\ell_2(x) + f(1)\ell_3(x) = \ell_2(x) = 1 - x^2$$

PRÁCTICA 2. Si sabemos que un determinado modelo toma los valores $f_1 = -1, f_2 = 0, f_3 = 1, f_4 = 0, f_5 = -1$, correspondientes a los valores $s_1 = -2, s_2 = -1, s_3 = 0, s_4 = 1, s_5 = 2$. Calcular $L_5[f](x)$ utilizando Octave.

SOLUCIÓN:
Comenzamos por ingresar los valores del parámetro s:

```
>> s=[-2 -1 0 1 2];
>> f=[-1 0 1 0 -1];
```

Luego calculamos el interpolante de Lagrange

```
>> Lf=polyfit (s,f,length(s)-1);
```

Ahora graficamos el interpolante de lagrange junto con los puntos de referencia de modelo.

```
>> ss=-2:4/100:2;
>> plot(s,f,'ro',ss,polyval(Lf,ss),'g')
```

Solución Alternativa:
Comenzamos por ingresar los valores del parámetro s:

```
>> s=[-2 -1 0 1 2];
>> f=[-1 0 1 0 -1];
```

Luego calculamos definimos el interpolante de Lagrange

```
>> Lf=@(x)interp1(s,f,x,"spline");
```

Ahora graficamos el interpolante de lagrange junto con los puntos de referencia de modelo.

```
>> ss=-2:4/100:2;
>> plot(s,f,'ro',ss,Lf(ss),'g')
```

2.4 Ejercicios de Práctica

1. Resolver las ecuaciones diferenciales implementando diferencias finitas en Octave con $N_x = N_y = 100,000$. En el caso de problemas de tipo VP, calcular los 100 automodos correspondientes a los autovalores más cercanos a 0.

   (a)
   $$\begin{cases} -\frac{d^2u}{dx^2} = 4x(1-x)sen(\pi x), x \in [0,1] \\ u(0) = u(1) = 0 \end{cases}$$

   (b)
   $$\begin{cases} -\frac{d^2u}{dx^2} = \pi^2 \lambda u, x \in [0,1] \\ u(0) = u(1) = 0 \end{cases}$$

   (c)
   $$\begin{cases} -(\partial_x^2 u + \partial_y^2 u) = sin(\pi xy), (x,y) \in [0,1]^2 \\ u(x,y) = 0, (x,y) \in \partial[0,1]^2 \end{cases}$$

   (d)
   $$\begin{cases} -(\partial_x^2 u + \partial_y^2 u) = \pi^2 \lambda u, (x,y) \in [0,1]^2 \\ u(x,y) = 0, (x,y) \in \partial[0,1]^2 \end{cases}$$

2. Sea $f(x) = sen(\pi x)$. Calcular $T^7_{1/2}[f](x)$.
3. Sea $g(x) = cos(\pi x/2)$. Calcular $L_5[g](x)$ respecto de los puntos $x_1 = -1, x_2 = -1/2, x_3 = 0, x_4 = 1/2, x_5 = 1$. Verificar que $T_0[L_5[g]](x) = L_5[g](x)$ para toda $x \in [-1,1]$.

4. Si el rango promedio de oscilación R en cm. de un puente, en el horario de 7:00 a 8:00 A.M., se ha registrado y promediado cada 5 minutos a partir de las 7:00 A.M. por un período de varios días, obteniendo los siguientes resultados:

   $R = \{0.9\,, 1.01\,, 0.7\,, 0.6\,, 1.1\,, 0.66\,, 0.77\,, 0.95\,, 0.98\,, 0.7\,, 0.6\,, 0.9\,, 1.2\}$

   (a) Calcule una función que permita estimar el rango de oscilación del puente en cualquier momento entre las 7 y las 8 A.M.

   (b) Calcule el rango de oscilación estimado correpondiente a las: 7:02, 7:31, 7:33, 7:42 y 7:57 A.M.

3 Interpolación en Dimensiones Superiores y Matrices de Diferenciación

3.1 Interpolación en 2D

Estudiar la solución de problemas de interpolación de la Forma:

PROBLEMA GENERAL DE INTERPOLACIÓN EN 2D. Dada una colección de puntos en $\mathbb{R}^2$ de la forma:

		x			
		x_1	x_2	$\cdots$	x_m
	y_1	(x_1, y_1)	(x_2, y_1)	$\cdots$	(x_m, y_1)
y	y_2	(x_1, y_2)	(x_2, y_2)	$\cdots$	(x_m, y_2)
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	y_n	(x_1, y_n)	(x_2, y_n)	$\cdots$	(x_m, y_n)

(8)

junto con una colección de valores correspondientes de la forma:

		x			
		x_1	x_2	$\cdots$	x_m
	y_1	$f(x_1, y_1)$	$f(x_2, y_1)$	$\cdots$	$f(x_m, y_1)$
y	y_2	$f(x_1, y_2)$	$f(x_2, y_2)$	$\cdots$	$f(x_m, y_2)$
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	y_n	$f(x_1, y_n)$	$f(x_2, y_n)$	$\cdots$	$f(x_m, y_n)$

Calcular el interpolante de Lagrange $L_{m,n}[f]$ en 2D, que cumple con las condiciones $L_{m,n}[f](x_j, y_k)$ para cada (x_j, y_k) en (8).

Solución:

Podemos calcular el interpolante de Lagrange utilizando la siguiente ecuación:

$$L_{m,n}[f](x, y) = \sum_{k=1}^{m} \sum_{k=1}^{m} f(x_k, y_j) \ell_{k,j}(x, y)$$

donde $\ell_{k,j}(x, y) = \ell_k(x)\ell_j(y)$ para cada $1 \leq k \leq m$ y cada $1 \leq j \leq n$.

Algunas propiedades de las funciones ℓ:

1. Tal como fue establecido en las lecturas de la primer semana dada una colección de puntos $x_1, x_2, \ldots, x_n$:

$$\ell_k(x) = \prod_{j=1, j\neq k}^{n} \frac{(x - x_j)}{(x_k - x_j)}$$

2. Tenemos que:

$$\ell_k(x_j) = \begin{cases} 1, & k = j \\ 0, & k \neq j \end{cases}$$

3. Tenemos que:

$$\ell_{k,l}(x_j, y_i) = \begin{cases} 1, & k = j \text{ y } l = i \\ 0, & \text{otro caso} \end{cases}$$

PRÁCTICA 1. Calcular el interpolante de Lagrange correspondiente a la tabla de datos utilizando Octave:

		x			
		-1	-0.5	0.5	1
y	0	-1	-1	-1	-1
	0.25	-0.5	-1.5	1.5	-0.5
	0.75	0.5	1.5	-1.5	0.5
	1	1	1	1	1

Solución:

1. Generamos la malla correspondiente a la tabla (8).

```
>> x=[-1 -0.5 0.5 1];
>> y=[0 0.25 0.75 1];
>> [X,Y]=meshgrid (x,y);
```

2. Ingresamos los valores de la tabla:

```
>> Z=[-1 -1 -1 -1;-0.5 -1.5 1.5 -0.5;0.5 1.5 -1.5 0.5;1 1 1 1];
```

3. Construimos los polinomios interpolantes $L_1[f](x), \ldots, L_4[f](x)$ que resuelven los problemas:

		x			
		-1	-0.5	0.5	1
y	0	-1	-1	-1	-1

		x			
		-1	-0.5	0.5	1
y	0.25	-0.5	-1.5	1.5	-0.5

		x			
		-1	-0.5	0.5	1
y	0.75	0.5	1.5	-1.5	0.5

y

		x			
		-1	-0.5	0.5	1
y	1	1	1	1	1

respectivamente.

```
>> for k=1:4,Lfx(k,:)=polyfit (x,Z(k,:),length(x)-1);end
>> format rat
>> Lfx
Lfx =

          0          0          0         -1
         -4       -2/3          4        1/6
          4        2/3         -4       -1/6
          0          0          0          1
```

4. Construimos los polinomios base $\ell_1(y), \ldots, \ell_4(y)$:

```
>> for k=1:4,ly(k,:)=polyfit (y,y==y(k),length(y)-1);end
>> ly
ly =

      -16/3       32/3      -19/3          1
       32/3      -56/3          8         -0
      -32/3       40/3       -8/3          0
       16/3      -16/3          1          0
```

5. Ensamblamos $L_{4,4}[f](x)$ en la forma:

$$L_{4,4}[f](x,y) = \sum_{j=1}^{4} L_j[f](x)\ell_j(y)$$

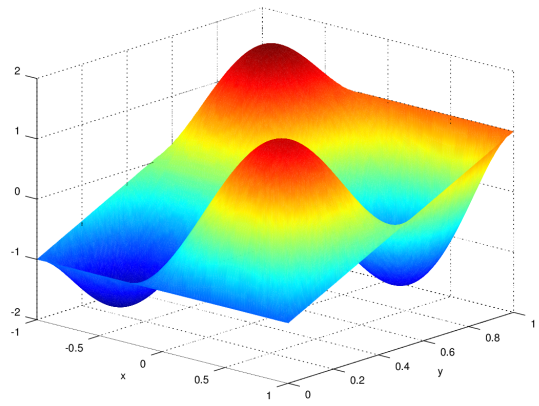
```
>> Lf=@(x,y)polyval(Lfx(1,:),x).*polyval(ly(1,:),y)+...
polyval(Lfx(2,:),x).*polyval(ly(2,:),y)+...
polyval(Lfx(3,:),x).*polyval(ly(3,:),y)+...
polyval(Lfx(4,:),x).*polyval(ly(4,:),y);
```

6. Verificamos la condición de interpolación:

```
>> Lf(X,Y)
ans =

         -1         -1         -1         -1
       -1/2       -3/2        3/2       -1/2
        1/2        3/2       -3/2        1/2
          1          1          1          1
```

7. Graficamos el interpolante de Lagrange en una malla más fina.

```
>> [xx,yy]=meshgrid (-1:2/100:1,0:1/100:1);
>> surf(xx,yy,Lf(xx,yy))
>> shading interp
```



Solución Alternativa:

1. Generamos la malla correspondiente a la tabla (8).

```
>> x=[-1 -0.5 0.5 1];
>> y=[0 0.25 0.75 1];
>> [X,Y]=meshgrid (x,y);
```

2. Ingresamos los valores de la tabla:

```
>> Z=[-1 -1 -1 -1;-0.5 -1.5 1.5 -0.5;0.5 1.5 -1.5 0.5;1 1 1 1];
```

3. Construimos $L_{4,4}[f](x)$ en la forma:

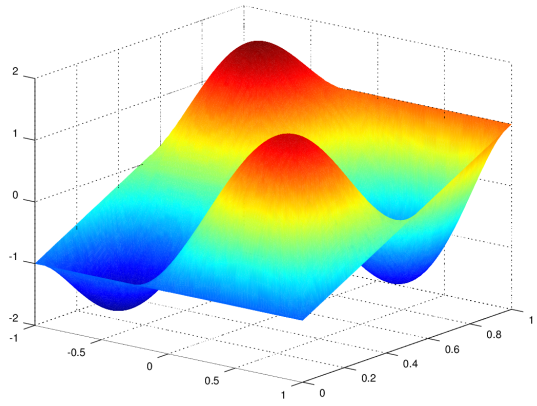
```
>> Lf=@(x,y)interp2(X,Y,Z,x,y,"spline");
```

4. Verificamos la condición de interpolación:

```
>> Lf(X,Y)
ans =

    -1    -1    -1    -1
  -1/2  -3/2   3/2  -1/2
   1/2   3/2  -3/2   1/2
     1     1     1     1
```

5. Graficamos el interpolante de Lagrange en una malla más fina.

```
>> [xx,yy]=meshgrid (-1:2/100:1,0:1/100:1);
>> surf(xx,yy,Lf(xx,yy))
>> shading interp
```



3.2 Matrices de Diferenciación

Estudiar la solución de problemas de aproximación de la derivada de una función diferenciable de la forma:

PROBLEMA GENERAL DIFERNCIACIÓN APROXIMADA POR MATRICES DE DIFERENCIACIÓN. Dada una Tabla de valores conocidos/calculables de un modelo f en $\mathbb{R}^2$ dada por la expresión:

x	x_1	x_2	$\cdots$	x_n
$f(x)$	$f(x_1)$	$f(x_2)$	$\cdots$	$f(x_n)$

Calcuar n fórmulas que permitan estimar $f'(x_j)$ para $j = 1, \ldots, n$, en la forma:

$$f'(x_k) = \sum_{j=1}^{n} d_{k,j} f(x_j)$$

y que además sean exactas cuando f es un polinomio de grado al menos $n-1$. Podemos representar las ecuaciones anteriores matricialmente como:

$$\begin{bmatrix} f'(x_1) \\ f'(x_2) \\ \vdots \\ f'(x_n) \end{bmatrix} = \begin{bmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,n} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n,1} & d_{n,2} & \cdots & d_{n,n} \end{bmatrix} \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix}$$

donde cada coeficiente $d_{k,j}$ está dado por la fórmula:

$$d_{k,j} = \ell'_j(x_k) \tag{9}$$

la matriz $\mathbf{D} = [d_{k,j}]_{n \times n}$ recibe el nombre de matriz de diferenciación.

PRÁCTICA 1. Dados los puntos $x_1 = -1, x_2 = 0, x_3 = 1$. Calcular la matriz de diferenciación correspondiente a la solución del problema general de diferenciacón aproximada.

SOLUCIÓN:

1. Calcular los polinomios base de interpolación ℓ_1, ℓ_2, ℓ_3 respecto de los puntos x_1, x_2, x_3:

$$\ell_1(x) = \frac{(x-x_2)}{(x_1-x_2)}\frac{(x-x_3)}{(x_1-x_3)} = \frac{(x-0)}{(-1-0)}\frac{(x-1)}{(-1-1)} = \frac{1}{2}x^2 - \frac{1}{2}x$$

$$\ell_2(x) = \frac{(x-x_1)}{(x_2-x_1)}\frac{(x-x_3)}{(x_2-x_3)} = \frac{(x-(-1))}{(0-(-1))}\frac{(x-1)}{(0-1)} = -x^2 + 1$$

$$\ell_3(x) = \frac{(x-x_1)}{(x_3-x_1)}\frac{(x-x_2)}{(x_3-x_2)} = \frac{(x-(-1))}{(1-(-1))}\frac{(x-0)}{(1-0)} = \frac{1}{2}x^2 + \frac{1}{2}x$$

2. Calcular las derivadas de los polinomios base:

$$\ell'_1(x) = x - \frac{1}{2}$$

$$\ell'_2(x) = -2x$$

$$\ell'_3(x) = x + \frac{1}{2}$$

3. Calcular la matrix de diferenciación $\mathbf{D}$ de acuerdo a la fórmula (9):

$$\mathbf{D} = \begin{bmatrix} \ell'_1(-1) & \ell'_2(-1) & \ell'_3(-1) \\ \ell'_1(0) & \ell'_2(0) & \ell'_3(0) \\ \ell'_1(1) & \ell'_2(1) & \ell'_3(1) \end{bmatrix} = \begin{bmatrix} -3/2 & 2 & -1/2 \\ -1/2 & 0 & 1/2 \\ 1/2 & -2 & 3/2 \end{bmatrix}$$

PRÁCTICA 2. Verificar las fórmulas de la PRÁCTICA 1 con la función $f(x) = 3x^2 - 4x + 1$.

SOLUCIÓN:

1. Tenemos que la función tiene una representación tabular respecto de x_1, x_2, x_3 de la forma:

$$\begin{array}{c|ccc} x & -1 & 0 & 1 \\ \hline f(x) & 8 & 1 & 0 \end{array}$$

2. Tenemos que $f'(x) = 6x - 4$ tiene una representación tabular:

$$\begin{array}{c|ccc} x & -1 & 0 & 1 \\ \hline f'(x) & -10 & -4 & 2 \end{array}$$

3. En forma matricial tenemos que:

$$\mathbf{Df} = \begin{bmatrix} -3/2 & 2 & -1/2 \\ -1/2 & 0 & 1/2 \\ 1/2 & -2 & 3/2 \end{bmatrix} \begin{bmatrix} 8 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -10 \\ -4 \\ 2 \end{bmatrix}$$

el resultado de la operación anterior verifica predicción acerca del nivel de aproximación de las fórmulas.

PRÁCTICA 3. Crear un archivo con nombre `dmat.m` utilizando el editor de Octave, dicho archivo generará una matrix de diferenciación a partir de cualquier colección de puntos $x_1, x_2, \ldots, x_n$ que reciba.

SOLUCIÓN:

1. El archivo `dmat.m` puede crearse implementando las siguientes líneas de código de Octave:

```
function D=dmat(x)
n=length(x);
for i=1:n
l(i,:)=polyfit(x,x==x(i),n-1);
endfor
for k=1:n
for j=1:n
D(k,j)=polyval(polyder(l(j,:)),x(k));
endfor
endfor
endfunction
```

PRÁCTICA 4. Utilizar el archivo `dmat.m` para desarrollar la práctica 3.

SOLUCIÓN:

1. Generamos los puntos $x_1 = -1, x_2 = 0, x_3 = 1$:

```
>> x=[-1 0 1];
```

2. Generamos la matrix **D** con `dmat.m`:

```
>> format rat
>> D=dmat(x)
D =

       -3/2          2       -1/2
       -1/2          0        1/2
        1/2         -2        3/2
```

3. Generamos el vector **f**:

```
>> f=3*x.^2-4*x+1;
>> f=f'
f =

          8
          1
          0
```

4. Calculamos la diferenciación exacta $\mathbf{f}'$:

```
>> fp=6*x-4;
>> fp=fp'
fp =

        -10
         -4
          2
```

5. Calculamos la diferenciación aproximada **Df**:

```
>> D*f
ans =

        -10
         -4
          2
```

6. Verificamos el error de aproximación:

```
>> norm(fp-D*f,inf)
ans = 0
```

PRÁCTICA 5. Calcular analítica y computacionalmente la sub-matriz $\mathbf{D}_2$ (el renglón 2) de la matrix $\mathbf{D}$ de la práctica 1.

SOLUCIÓN:

Analíticamente:

$$\mathbf{D}_2 = [d_{2,1}\ \ d_{2,2}\ \ d_{2,3}] = [\ell_1'(0)\ \ \ell_2'(0)\ \ \ell_3'(0)] = \left[-\frac{1}{2}\ \ 0\ \ \frac{1}{2}\right]$$

Computacionalmente:

```
>> x=[-1 0 1];
>> D=dmat(x);
>> D(2,:)
ans =

       -1/2            0          1/2
```

3.3 Representaciones matriciales alternativas del operador de diferenciación

Consideremos nuevamente el operador $D : C^{n+1}([a,b]) \to C^n([a,b]) : f \mapsto Df$, para $n \in \mathbb{Z}_0^+$. Tenemos que la siguiente propiedad que puede ser deducida por el lector a manera de ejercicio:

Propiedad: Si $x_0, x_1, \ldots, x_N \in \mathbb{R}$ son distintos, entonces la función cardinal $c_j(x)$ definida por

$$c_j(x) = \frac{1}{p_j} \prod_{k=0, k\neq k}^{N} (x - x_k), \tag{10}$$

$$p_j = \prod_{k=0, k\neq k}^{N} (x_j - x_k) \tag{11}$$

es el único interpolante polinómico de grado N que cumple la condición.

$$c_j(x_k) = \begin{cases} 1, & j = k \\ 0, & j \neq k \end{cases}, 0 \leq j, k \leq N$$

Tenemos además que la matriz de diferenciación $D \in \mathbb{C}^{N\times N}$ correspondiente a $\{c_0(x), c_1(x), c_2(x), \ldots, c_N(x)\}$ tiene una representación alternativa determinada por las fórmulas:

$$D_{ij} = c_j'(x_i) = \frac{1}{p_j} \prod_{k=0, k\neq i,j}^{N} (x_i - x_k) = \frac{p_i}{p_j(x_i - x_j)} \quad (i \neq j) \quad (12)$$

$$D_{ii} = c_i'(x_i) = \sum_{k=0, k\neq i}^{N} (x_i - x_k)^{-1}. \quad (13)$$

El siguiente código Octava/MATLAB que puede ser creado como el archivo `Dmat.m` calcula una matriz de diferenciación D correspondiente a una partición $x_0 < x_1 < \cdots < x_{N-1} < x_N$.

```
function [D,x]=Dmat(x)
x=x(:);
N=length(x)-1;
D=repmat(x,1,N+1);
E=eye(N+1);
D=D'-D+E;
p=diag(prod(D));
s=diag(sum(1./D)-1);
D=p*(1./D')/p-E+s;
endfunction
```

El lector puede ahora repetir las prácticas anteriores correspondientes al cómputo con matrices de diferenciación, para verificar la equivalencia entre representaciones.

3.4 Ejercicios de Práctica

1. Verificar la propiedad 3 de las funciones base ℓ:

$$\ell_{k,l}(x_j, y_i) = \begin{cases} 1, & k = j \text{ y } l = i \\ 0, & \text{otro caso} \end{cases}$$

2. Probar que para cualquier $\ell_j(x)$ correspondiente a una colección de puntos $x_1, x_2, \ldots, x_n \in \mathbb{R}$. $T_{x_k}^{n-1}[\ell_j](x) = \ell_j(x)$ para cada punto x_j en la colección.

3. Resolver analíticamente el problema de interpolación:

		x	
		-1	1
y	0	-1	1
	1	1	-1

4. Se denominan puntos de Chebyshev de orden n a cada colección de puntos $x_1, \ldots, x_{n+1}$ definidos por $x_k = \cos(\pi(k-1)/n)$ para $1 \leq k \leq (n+1)$. Calcular la matrix de diferenciación $\mathbf{D}$ correspondiente a los puntos de Chebyshev de orden 6.

4 Método de Diferencias Finitas (MDF) en Dimensiones Superiores

4.1 Solución Numérica de Modelos Dinámicos

4.1.1 MDF para Problemas de Valor Inicial y de Frontera:

Estudiar la solución de las siguientes formas generales de ecuaciones diferenciales utilizando diferencias finitas, desarrollando un algoritmo computacional e implementando el mismo en Octave/SciLab.

PROBLEMA GENERAL DE MOVIMIENTO ONDULATORIO EN 1D: Utilizamos este tipo de modelos para simular la deflexión de **varillas** representativas de una estructura dada.

$$\begin{cases} \partial_t^2 u - \alpha^2 \partial_x^2 u = f(x,t), (x,t) \in [0,1] \times [0,\infty) \\ u(0,t) = u(1,t) = 0 \\ u(x,0) = u_0(x) \\ \partial_t u(x,0) = v_0(x) \end{cases}$$

Solución:

Algoritmo de Solución para el PROBLEMA GENERAL DE MOVIMIENTO ONDULATORIO EN 1D:

1. Consideremos las fórmulas estudiadas en clase de la forma:

$$\partial_t u(x,t) \approx \frac{1}{h_t}(u(x,t+h) - u(x,t))$$

$$\partial_x^2 u(x,t) \approx \frac{1}{h^2}(u(x-h,t) - 2u(x,t) + u(x+h,t))$$

2. Consideremos las particiones de $[0,1] \times [0,T]$ definidas por:

$$0 = x_0 < x_1 < x_2 < \cdots < x_{N-1} < x_N = 1$$

$$0 = t_0 < x_1 < x_2 < \cdots < t_{M-1} < t_M = T$$

   donde $x_k = kh$, $t_j = jh_t$ para $h = 1/N$, $h_t \leq h^2/\alpha^2$ y $0 \leq k \leq N$.

3. Reduzcamos el orden del modelo diferencial a la forma:

$$\begin{cases} \partial_t u = v, \\ \partial_t v = \alpha^2 \partial_x^2 u + f(x,t), (x,t) \in [0,1] \times [0,\infty) \\ u(0,t) = u(1,t) = 0 \\ u(x,0) = u_0(x) \\ v(x,0) = v_0(x) \end{cases}$$

Llamamos a las ecuaciones resultantes de esta reducción, ecuaciones dinámicas representativas de primer orden, o solo ecuaciones dinámicas por simplicidad.

4. Definamos $u_{k,j} := u(x_k, t_j) = u(kh, jh_t)$ y $v_{k,j} := v(x_k, t_j) = v(kh, jh_t)$, $0 \leq k, j \leq N$. Aplicando los pasos [1], [2] y [3] podemos obtener una formulación discreta del problema general de la forma:

$$\begin{cases} u_{k,j+1} = u_{k,j} + h_t v_{k,j} \\ v_{k,j+1} = v_{k,j} + \frac{h_t \alpha^2}{h^2}(u_{k-1,j} - 2u_{k,j} + u_{k+1,j}) + h_t f(x_k, t_j) \end{cases}, 1 \leq k \leq N-1.$$

5. Podemos ahora utlizar el paso [4] para obtener una representación matricial de la discretización de la forma:

$$\begin{cases} \mathbf{u}_{j+1} = \mathbf{u}_j + h_t \mathbf{v}_j \\ \mathbf{v}_{j+1} = \mathbf{v}_j + h_t \alpha^2 \mathbf{L}_{1d} \mathbf{u}_j + h_t \mathbf{f}_j \end{cases}, 1 \leq k \leq N-1.$$

donde $\mathbf{u}_j = [u_{1,j}, u_{2,j}, \ldots, u_{N-1,j}]^\top$, $\mathbf{f}_j = [f(x_1, t_j), f(x_2, t_j), \ldots, f(x_{N-1}, t_j)]^\top$ y donde $\mathbf{L}_{1d}$ es la matriz de coeficientes dada por la ecuación:

$$\mathbf{L}_{1d} = \frac{1}{h^2} \begin{bmatrix} -2 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ldots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & -2 & 1 \\ 0 & \cdots & 0 & 0 & 0 & 1 & -2 \end{bmatrix}$$

6. Resolvemos el sistema del paso [5] utilizando un procedimiento nuérico de integración numérica en el tiempo.
7. Representamos las solución aproximada de la ecuacón en forma general en un gráfico dinámico.

PRÁCTICA 1.

Simular la deflexión de una varilla representativa descrita por una ecuación de la forma:

$$\begin{cases} \partial_t^2 u - \alpha^2 \partial_x^2 u = 0, (x,t) \in [0,1] \times [0, \infty) \\ u(0,t) = u(1,t) = 0 \\ u(x,0) = Asen(\pi m x) \\ \partial_t u(x,0) = 0 \end{cases}$$

SOLUCIÓN: Implementaremos el algoritmo de solución utilizando Octave:

1. Creamos el archivo `*.m` cuyo código se presenta a continuación:

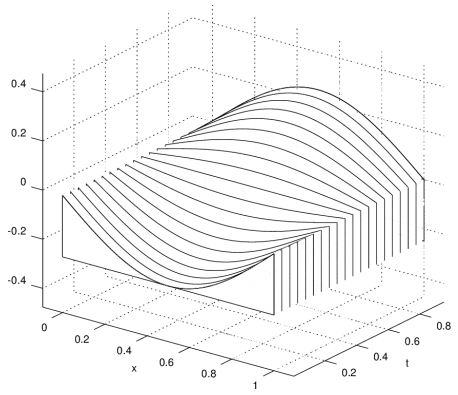
```
function [U,x,T]=varilla_simple(M,N,c2,m,A)
h=1/N;
ht=1/M;
x=0:h:1;
L2=spdiags (ones(N-1,1)*[1 -2 1],-1:1,N-1,N-1);
L2=1/h^2*L2;
u0=A*sin(pi*m*x(2:N));
v0=zeros(N-1,1);
var=@(xx,u)interp1(x,u,xx,"spline");
u=u0(:);
v=v0(:);
U=[];
T=U;
for j=1:M
u=u+ht*v;
v=v+ht*c2*L2*u;
if mod(j-1,floor(M/20))==0
U=[U [0;u;0]];
T=[T j*ht];
endif
plot(x,[0;u;0],0:1/10:1,var(0:1/10:1,[0;u;0]),'ro');
axis([0 1 -1 1]);
pause(.2);
endfor
endfunction
```

2. Realizamos un experimento computacional con los parámetros $N = 100, M = 400, A = -0.25, \alpha^2 = c^2 = 1$ y $m = 1$.

```
>> [u,x,t]=varilla_simple(400,100,1,1,-.25);
```

3. Creamos un gráfico dinámico que muestra la evolución en el tiempo de la varilla representativa.

```
>> [X,T]=meshgrid (x,t);
>> waterfall (X,T,u')
>> axis equal
>> xlabel ('x')
>> ylabel ('t')
```



Solución: Implementaremos ahora el algoritmo de solución utilizando el método de Crank-Nicolson en Octave:

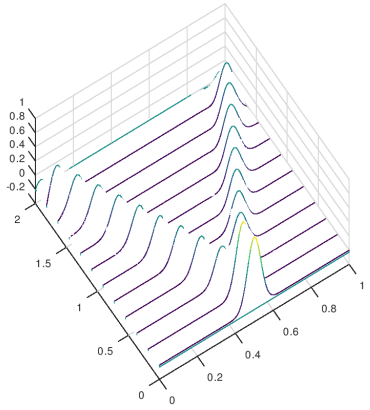
1. Creamos el archivo `*.m` cuyo código se presenta a continuación:

```
function [t,x,w]=wave_1D_CN(c,L,N,m,t)
hx=L/N;
ht=diff(t)/m;
L1D=c*spdiags(ones(N-1,1)*[1 -2 1],-1:1,N-1,N-1)/hx^2;
L1D=[sparse(N-1,N-1),speye(N-1);L1D,sparse(N-1,N-1)];
x=0:hx:L;
w0=4*exp(-(400/L)*(x-L/2).^2).*x.*(L-x)/L^2;
w=w0';
w0=w(2:N);
w0=[w0;zeros(N-1,1)];
for k=1:N
    plot(x,[0;w0(1:(N-1),1);0]);
    axis([0 L -1.5 1.5]);
    pause(.2);
    w0=(2*speye(2*(N-1))+ht*L1D)*w0;
    w0=(2*speye(2*(N-1))-ht*L1D)\w0;
    if mod(k-1,10)==0
        w=[w,[0;w0(1:(N-1),1);0]];
    end
end
w=[w,[0;w0(1:(N-1),1);0]];
Nw=size(w,2);
[x,t]=meshgrid(x,t(1):diff(t)/(Nw-1):t(2));
waterfall(x,t,w')
endfunction
```

2. Realizamos un experimento computacional con los parámetros $N = 100, M = 400, \alpha^2 = c^2 = 1$ y $L = T = 1$.

```
>> [t,x,w]=wave_1D_CN(1,1,100,200,[0 1]);
```

3. El programa desarrolla una animación de la evolución de la deflexión de la varilla representativa y calcula un gráfico dinámico que muestra la evolución en el intervalo de tiempo $[0, 1]$.



PROBLEMA GENERAL DE CÁLCULO DE MODOS DE VIBRACIÓN DE VARILLAS REPRESENTATIVAS (Representación de Helmholtz): Dado un problema de deflexión de varillas de la forma:

$$\begin{cases} \partial_t^2 u - \alpha^2 \partial_x^2 u = f(x,t), (x,t) \in [0,1] \times [0,\infty) \\ u(0,t) = u(1,t) = 0 \\ u(x,0) = u_0(x) \\ \partial_t u(x,0) = v_0(x) \end{cases}$$

Hacemos las suposiciones $u(x,t) = v(x)e^{i\omega t}$ y $f(x,t) = g(x)e^{i\omega t}$, donde $i = \sqrt{-1}$. Bajo condiciones físicamente apropiadas y realizables tenemos que las ecuaciones anteiores producen una expresión de la forma:

$$\begin{cases} e^{i\omega t}(i\omega)^2 v(x) - e^{i\omega t}\alpha^2 \partial_x^2 v(x) = g(x)e^{i\omega t}, (x,t) \in [0,1] \times [0,\infty) \\ v(0)e^{i\omega t} = v(0)e^{i\omega t} = 0 \end{cases}$$

Luego de simplificar la expresión anterior obtenemos el siguiente resultado:

$$\begin{cases} -\alpha^2 \partial_x^2 v(x) - \omega^2 v(x) = g(x), x \in [0,1] \\ v(0) = v(1) = 0 \end{cases}$$

La expresión anterior se denomina representación de Helmholtz del problema de deflexión, y corresponde a un problema de valor de frontera que podemos resolver con el MDF utilizando el siguiente algoritmo:

SOLUCIÓN:

El procedimiento de solución es el siguiente:

1. Calcular la representación de Helmholtz del problema de deflexión para obetener la expresión:

$$\begin{cases} -\alpha^2 \partial_x^2 v(x) - \omega^2 v(x) = g(x), x \in [0,1] \\ v(0) = v(1) = 0 \end{cases}$$

2. Discretizar la expresión del paso [1] utilizando las técnicas de las lecturas de la semana 1 para obtener:

$$-\frac{\alpha^2}{h^2}(v_{k-1} - 2v_k + v_{k+1}) - \omega^2 v_k = g(x_k), 1 \leq k \leq N-1,$$

donde $x_k = kh$, $h = 1/N$, para $N \in \mathbb{Z}^+$.

3. Calcular la representación matricial de la discretización del paso [2] de la forma:

$$-\alpha^2 \mathbf{L}_{1d}\mathbf{v} - \omega^2 \mathbf{v} = \mathbf{g}$$

donde $\mathbf{L}_{1d}$ está determinada por la expresión

$$\mathbf{L}_{1d} = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \cdots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & -2 & 1 \\ 0 & \cdots & 0 & 0 & 0 & 1 & -2 \end{bmatrix}$$

y donde $\mathbf{v} = [v_1, v_2, \ldots, v_{N-1}]^\top$, $\mathbf{g} = [g(x_1), g(x_2), \ldots, g(x_{N-1})]^\top$.

4. Resolver el problema de álgebra lineal del paso [3].

PRÁCTICA 2.

Utilizar el MDF implementado en Octave para resolver numéricamente el problema de Helmholtz correspondiente a la ecuación:

$$\begin{cases} \partial_t^2 u - \pi^2 \partial_x^2 u = 100 sen(\pi x) e^{i\omega t}, (x,t) \in [0,1] \times [0,\infty) \\ u(0,t) = u(1,t) = 0 \\ u(x,0) = u_0(x) \\ \partial_t u(x,0) = v_0(x) \end{cases}$$

Suponiendo que $\omega = 1$. SOLUCIÓN:

1. La representación de Helmholtz del problema esta dada por:

$$\begin{cases} -\pi^2 \partial_x^2 v(x) - \omega^2 v(x) = 100 sen(\pi x), x \in [0,1] \\ v(0) = v(0) = 0 \end{cases}$$

2. La expresión del paso [2] tiene una discretización de la forma:

$$-\frac{\pi^2}{h^2}(v_{k-1} - 2v_k + v_{k+1}) - \omega^2 v_k = 100sen(\pi x_k), 1 \leq k \leq N-1,$$

donde $x_k = kh$, $h = 1/N$, para $N \in \mathbb{Z}^+$.

3. La forma matricial de la discretización del paso [2] es:

$$-\pi^2 \mathbf{L}_{1d}\mathbf{v} - \omega^2 \mathbf{v} = \mathbf{g}$$

donde $\mathbf{L}_{1d}$ está determinada por la expresión

$$\mathbf{L}_{1d} = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \cdots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & -2 & 1 \\ 0 & \cdots & 0 & 0 & 0 & 1 & -2 \end{bmatrix}$$

y donde $\mathbf{v} = [v_1, v_2, \ldots, v_{N-1}]^\top$, $\mathbf{g} = [100sen(\pi x_1), 100sen(\pi x_2), \ldots, 100sen(\pi x_{N-1})]^\top$.

4. Supongamos que $\omega = 1$, podemos resolver el problema numéricamente utilizando los pasos anteriores y el siguiente procedimiento en Octave:

   4.1 Calculamos el mallado de $[0, 1]$:

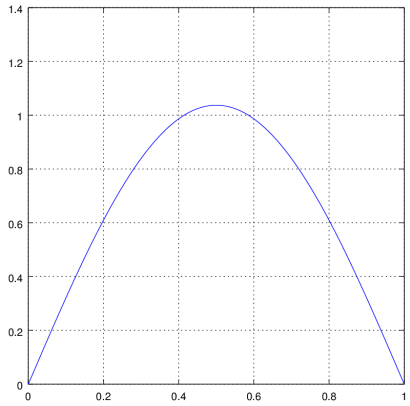
```
>> N=1000;
>> h=1/N;
>> x=0:h:1;
```

   4.2 Calculamos la representación matricial de la discretización:

```
>> alfa2=pi^2;
>> omega2=1;
>> L1d=spdiags (ones(N-1,1)*[1 -2 1],-1:1,N-1,N-1);
>> L1d=1/h^2*L1d;
>> H=-alfa2*L1d-omega2*speye(N-1);
>> g=100*sin(pi*x(2:N))';
```

   4.3 Reolvemos el problema matricial del paso [4.2] y graficamos la solución numérica:

```
>> v=H\g;
>> plot(x,[0;v;0])
```



4.4 Calculamos una malla más fina de $[0, 1]$:

```
>> M=10000;
>> he=1/M;
>> xe=0:he:1;
```

4.5 Recalculamos la discretización en la malla más fina:

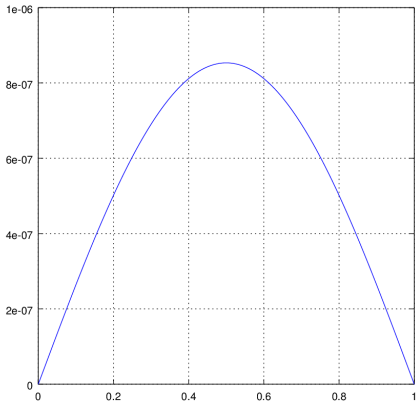
```
>> L1e=spdiags (ones(M-1,1)*[1 -2 1],-1:1,M-1,M-1);
>> L1e=1/he^2*L1e;
>> He=-alfa2*L1e-omega2*speye(M-1);
>> ge=100*sin(pi*xe(2:M))';
```

4.6 Recalculamos la solución con mayor precisión:

```
>> ve=He\ge;
```

4.7 Utilizamos este cómputo para estimar el error de aproximación:

```
Ve=@(x)interp1(xe,[0;ve;0],x,"spline");
>> norm(Ve(x)'-[0;v;0],inf)
ans =    8.5333e-07
>> plot(x,abs(Ve(x)'-[0;v;0]))
```



4.2 Ejercicios de Práctica

1. Calcular la discretización correspondiente y resolver el problema de deflexión de varillas implementando MDF en Octave.

$$\begin{cases} \partial_t^2 u - 4\partial_x^2 u = 0, (x,t) \in [0,1] \times [0,\infty) \\ u(0,t) = u(1,t) = 0 \\ u(x,0) = sen(\pi x) \\ \partial_t u(x,0) = 0 \end{cases}$$

2. Calcular la discretización correspondiente y resolver el problema de Helmholtz correspondiente al modelo de deflexión de varillas implementando MDF en Octave.

$$\begin{cases} \partial_t^2 u - 4\partial_x^2 u = sen(\pi x/4)e^{it}, (x,t) \in [0,1] \times [0,\infty) \\ u(0,t) = u(1,t) = 0 \\ u(x,0) = u_0(x) \\ \partial_t u(x,0) = v_0(x) \end{cases}$$

3. Calcular la discretización correspondiente y resolver el problema de deflexión **amortiguada/incrementada** de varillas implementando MDF en Octave.

$$\begin{cases} \partial_t^2 u + c\partial_t u - \alpha^2 \partial_x^2 u = 0, (x,t) \in [0,1] \times [0,\infty) \\ u(0,t) = u(1,t) = 0 \\ u(x,0) = Asen(\pi m x) \\ \partial_t u(x,0) = 0 \end{cases}$$

Idea: Reformular las ecuaciones dinámicas y la discretización del caso simple de la práctica 1, modificando convenientemente el archivo

`*.m` correspondiente a la varilla simple de la práctica 1. Cuál es el efecto de un coeficiente c positivo/negativo/cero?

5 Método de Diferencias Finitas (MDF) en Dimensiones Superiores

5.1 Solución Numérica de Modelos Dinámicos

5.1.1 Método de Diferencias Finitas (MDF) en Dimensiones Superiores

MDF para Problemas de Valor Inicial y de Frontera: Estudiar la solución de las siguientes formas generales de ecuaciones diferenciales utilizando diferencias finitas, desarrollando un algoritmo computacional e implementando el mismo en Octave/SciLab.

PROBLEMA GENERAL DE MOVIMIENTO ONDULATORIO EN 2D: Utilizamos este tipo de modelos para simular la deflexión de **membranas** representativas de una estructura dada.

$$\begin{cases} \partial_t^2 u - \alpha^2(\partial_x^2 u + \partial_y^2 u) = f(x,t), (x,y,t) \in [0,1]^2 \times [0,\infty) \\ u(0,y,t) = u(1,y,t) = 0 \\ u(x,0,t) = u(x,0,t) = 0 \\ u(x,y,0) = u_0(x,y) \\ \partial_t u(x,y,0) = v_0(x,y) \end{cases}$$

Solución:

Algoritmo de Solución para el PROBLEMA GENERAL DE MOVIMIENTO ONDULATORIO EN 1D:

1. Consideremos las fórmulas estudiadas en clase de la forma:

$$\partial_t u(x,y,t) \approx \frac{1}{h_t}(u(x,y,t+h) - u(x,y,t))$$

$$\partial_x^2 u(x,y,t) \approx \frac{1}{h^2}(u(x-h,y,t) - 2u(x,y,t) + u(x+h,y,t))$$

$$\partial_y^2 u(x,y,t) \approx \frac{1}{h^2}(u(x,y-h,t) - 2u(x,y,t) + u(x,y+h,t))$$

2. Consideremos las particiones de $[0,1] \times [0,T]$ definidas por:

$$0 = x_0 < x_1 < x_2 < \cdots < x_{N-1} < x_N = 1$$

$$0 = y_0 < y_1 < y_2 < \cdots < y_{N-1} < y_N = 1$$

$$0 = t_0 < x_1 < x_2 < \cdots < t_{M-1} < t_M = T$$

donde $x_k = kh$, $y_k = kh$, $t_j = jh_t$ para $h = 1/N$, $h_t \leq h^2/\alpha^2$ y $0 \leq k \leq N$.

3. Reduzcamos el orden del modelo diferencial a la forma:

$$\begin{cases} \partial_t u = v, \\ \partial_t v = \alpha^2(\partial_x^2 u + \partial_y^2 u) + f(x,t), (x,t) \in [0,1]^2 \times [0,\infty) \\ u(0,y,t) = u(1,y,t) = 0 \\ u(x,0,t) = u(x,0,t) = 0 \\ u(x,y,0) = u_0(x,y) \\ v(x,y,0) = v_0(x,y) \end{cases}$$

Llamamos a las ecuaciones resultantes de esta reducción, ecuaciones dinámicas representativas de primer orden, o solo ecuaciones dinámicas por simplicidad.

4. Definamos $u_{k,j,l} := u(x_k, y_j, t_l) = u(kh, jh, lh_t)$ y $v_{k,j,l} := v(x_k, y_j, t_j) = v(kh, jh, lh_t)$, $0 \leq k, j \leq N$, $1 \leq l \leq M$. Aplicando los pasos [1], [2] y [3] podemos obtener una formulación discreta del problema general de la forma:

$$\begin{cases} u_{k,j,l+1} = u_{k,j,l} + h_t v_{k,j,l} \\ v_{k,j,l+1} = v_{k,j,l} + \frac{h_t \alpha^2}{h^2}(u_{k-1,j,l} + u_{k,j-1,l} - 4u_{k,j,l} + u_{k+1,j,l} + u_{k,j+1,l}) + h_t f(x_k, y_j, t_l) \end{cases},$$

$1 \leq k, j \leq N-1$.

5. Podemos ahora utlizar el paso [4] para obtener una representación matricial de la discretización de la forma:

$$\begin{cases} \mathbf{u}_{j+1} = \mathbf{u}_j + h_t \mathbf{v}_j \\ \mathbf{v}_{j+1} = \mathbf{v}_j + h_t \alpha^2 \mathbf{L}_{2d} \mathbf{u}_j + h_t \mathbf{f}_j \end{cases}, 1 \leq k, j \leq N-1.$$

donde $\mathbf{u}_j = [u_{1,1,j}, u_{1,2,j}, \ldots, u_{N-1,N-1,j}]^\top$, $\mathbf{f}_j = [f(x_1, y_1, t_j), f(x_1, y_2, t_j), \ldots, f(x_{N-1}, y_{N-1}, t_j)]^\top$ y donde $\mathbf{L}_{2d}$ es la matriz de coeficientes dada por la ecuación:

$$\mathbf{L}_{2d} = \mathbf{L}_{1d} \otimes \mathbf{I}_{N-1} + \mathbf{I}_{N-1} \otimes \mathbf{L}_{1d}$$

donde $\mathbf{L}_{1d}$ es la matriz definida por la expresión:

$$\mathbf{L}_{1d} = \frac{1}{h^2} \begin{bmatrix} -2 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \cdots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & -2 & 1 \\ 0 & \cdots & 0 & 0 & 0 & 1 & -2 \end{bmatrix}$$

6. Resolvemos el sistema del paso [5] utilizando un procedimiento numérico de integración numérica en el tiempo.

7. Representamos las solución aproximada de la ecuacón en forma general en un gráfico dinámico.

PRÁCTICA 1.

Simular la deflexión de una membrana representativa descrita por una ecuación de la forma:

$$\begin{cases} \partial_t^2 u - \alpha^2(\partial_x^2 u + \partial_y^2 u) = 0, (x,y,t) \in [0,1] \times [0,\infty) \\ u(0,y,t) = u(1,y,t) = 0 \\ u(x,0,t) = u(x,1,t) = 0 \\ u(x,y,0) = Ae^{-k((x-1/2)^2+(y-1/2)^2)} sen(\pi m x) sen(\pi m y) \\ \partial_t u(x,y,0) = 0 \end{cases}$$

Solución: Implementaremos el algoritmo de solución utilizando Octave:

1. Creamos el archivo `*.m` cuyo código se presenta a continuación:

```
function [W,Wg,x,y]=membrana_simple(M,N,c2,m,k,A)
h=1/N;
ht=1/M;
x=0:h:1;
[x,y]=meshgrid(x);
L2=spdiags (ones(N-1,1)*[1 -2 1],-1:1,N-1,N-1);
L2=1/h^2*L2;
E=speye(N-1);
L2=kron(L2,E)+kron(E,L2);
u0=A*exp(-k*((x(2:N,2:N)-.5).^2+(y(2:N,2:N)-.5).^2)).*...
sin(pi*m*x(2:N,2:N)).*sin(pi*m*y(2:N,2:N));
v0=zeros(N-1,N-1);
u=u0(:);
v=v0(:);
W=zeros(N+1);
[xx,yy]=meshgrid(0:1/25:1);
Wg=@(xx,yy,u)interp2(x,y,u,xx,yy,"spline");
for j=1:M
u=u+ht*v;
v=v+ht*c2*L2*u;
W(2:N,2:N)=reshape(u,N-1,N-1);
colormap([0 0 0]);
mesh(xx,yy,Wg(xx,yy,W));
axis([0 1 0 1 -1 1]);
pause(.2);
endfor
endfunction
```

2. Realizamos un experimento computacional con los parámetros $N = 100, M = 400, A = 0.9, \alpha^2 = c^2 = 1$, $m = 1$ y $k = 20$.

```
[W,Wg,x,y]=membrana_simple(100,25,1,1,20,.9);
```

PROBLEMA GENERAL DE CÁLCULO DE MODOS DE VIBRACIÓN DE VARILLAS REPRESENTATIVAS (Representación de Helmholtz): Dado un problema de deflexión de varillas de la forma:

$$\begin{cases} \partial_t^2 u - \alpha^2(\partial_x^2 u + \partial_y^2 u) = f(x,y,t), (x,y,t) \in [0,1]^2 \times [0,\infty) \\ u(0,y,t) = u(1,y,t) = 0 \\ u(x,0,t) = u(x,0,t) = 0 \\ u(x,y,0) = u_0(x,y) \\ \partial_t u(x,y,0) = v_0(x,y) \end{cases}$$

Hacemos las suposiciones $u(x,y,t) = v(x,y)e^{i\omega t}$ y $f(x,y,t) = g(x,y)e^{i\omega t}$, donde $i = \sqrt{-1}$. Bajo condiciones físicamente apropiadas y realizables tenemos que las ecuaciones anteiores producen una expresión de la forma:

$$\begin{cases} e^{i\omega t}(i\omega)^2 v(x,y) - e^{i\omega t}\alpha^2(\partial_x^2 v(x,y) + \partial_y^2 v(x,y)) = g(x,y)e^{i\omega t}, (x,t) \in [0,1] \times [0,\infty) \\ v(0,y)e^{i\omega t} = v(1,y)e^{i\omega t} = 0 \\ v(x,0)e^{i\omega t} = v(x,1)e^{i\omega t} = 0 \end{cases}$$

Luego de simplificar la expresión anterior obtenemos el siguiente resultado:

$$\begin{cases} -\alpha^2(\partial_x^2 v(x,y) + \partial_y^2 v(x,y)) - \omega^2 v(x,y) = g(x,y), x \in [0,1] \\ v(0,y) = v(1,y) = 0 \\ v(x,0) = v(x,1) = 0 \end{cases}$$

La expresión anterior se denomina representación de Helmholtz del problema de deflexión, y corresponde a un problema de valor de frontera que podemos resolver con el MDF utilizando el siguiente algoritmo:

Solución:

El procedimiento de solución es el siguiente:

1. Calcular la representación de Helmholtz del problema de deflexión para obetener la expresión:

$$\begin{cases} -\alpha^2(\partial_x^2 v(x,y) + \partial_y^2 v(x,y)) - \omega^2 v(x,y) = g(x,y), x \in [0,1]^2 \\ v(0,y) = v(1,y) = 0 \\ v(x,0) = v(x,1) = 0 \end{cases}$$

2. Discretizar la expresión del paso [1] utilizando las técnicas de las lecturas de la semana 1 para obtener:

$$-\frac{\alpha^2}{h^2}(v_{k-1,j}+v_{k,j-1}-4v_{k,j}+v_{k+1,j}+v_{k,j+1})-\omega^2 v_{k,j} = g(x_k,y_j), 1 \leq k,j \leq N-1,$$

donde $x_k = kh$, $y_j = jh$, $h = 1/N$, para $N \in \mathbb{Z}^+$.

3. Calcular la representación matricial de la discretización del paso [2] de la forma:

$$-\alpha^2 \mathbf{L}_{2d}\mathbf{v} - \omega^2 \mathbf{v} = \mathbf{g}$$

donde $\mathbf{L}_{2d}$ está determinada por la expresión

$$\mathbf{L}_{2d} = \mathbf{L}_{1d} \otimes \mathbf{I}_{N-1} + \mathbf{I}_{N-1} \otimes \mathbf{L}_{1d}$$

con $\mathbf{L}_{1d}$ definida por:

$$\mathbf{L}_{1d} = \frac{1}{h^2} \begin{bmatrix} -2 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ldots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & -2 & 1 \\ 0 & \cdots & 0 & 0 & 0 & 1 & -2 \end{bmatrix}$$

y donde $\mathbf{v} = [v_{1,1}, v_{1,2}, \ldots, v_{N-1,N-1}]^\top$, $\mathbf{g} = [g(x_1, y_1), g(x_1, y_2), \ldots, g(x_{N-1}, y_{N-1})]^\top$.

4. Resolver el problema de álgebra lineal del paso [3].

PRÁCTICA 2.

Utilizar el MDF implementado en Octave para resolver numéricamente el problema de Helmholtz correspondiente a la ecuación:

$$\begin{cases} \partial_t^2 u - (\partial_x^2 u + \partial_y^2 u) = 0, (x,t) \in [0,1]^2 \times [0,\infty) \\ u(0,y,t) = u(1,y,t) = 0 \\ u(x,0,t) = u(x,1,t) = 0 \\ u(x,y,0) = u_0(x,y) \\ \partial_t u(x,y,0) = v_0(x,y) \end{cases}$$

Solución:

1. La representación de Helmholtz del problema esta dada por:

$$\begin{cases} -(\partial_x^2 v(x,y) + \partial_y^2 v(x,y)) - \omega^2 v(x,y) = 0, (x,y) \in [0,1]^2 \\ v(0,y) = v(1,y) = 0 \\ v(x,0) = v(x,1) = 0 \end{cases}$$

2. La expresión del paso [2] tiene una discretización de la forma:

$$-\frac{\pi^2}{h^2}(v_{k-1,j} + v_{k,j-1} - 4v_{k,j} + v_{k+1,j} + v_{k,j+1}) = \omega^2 v_{k,j}, 1 \leq k \leq N-1,$$

donde $x_k = kh$, $y_j = jh$, $h = 1/N$, para $N \in \mathbb{Z}^+$.

3. La forma matricial de la discretización del paso [2] es:

$$-\mathbf{L}_{2d}\mathbf{v} = \omega^2\mathbf{v}$$

donde $\mathbf{L}_{2d}$ está determinada por la expresión

$$\mathbf{L}_{2d} = \mathbf{L}_{1d} \otimes \mathbf{I}_{N-1} + \mathbf{I}_{N-1} \otimes \mathbf{L}_{1d}$$

con $\mathbf{L}_{1d}$ definida por:

$$\mathbf{L}_{1d} = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \cdots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & -2 & 1 \\ 0 & \cdots & 0 & 0 & 0 & 1 & -2 \end{bmatrix}$$

y donde $\mathbf{v} = [v_{1,1}, v_{1,2}, \ldots, v_{N-1,N-1}]^\top$.

4. Podemos resolver el problema de valores propios correspondiente numéricamente, utilizando los pasos anteriores y el siguiente procedimiento en Octave:

   4.1 Calculamos el mallado de $[0, 1]$:

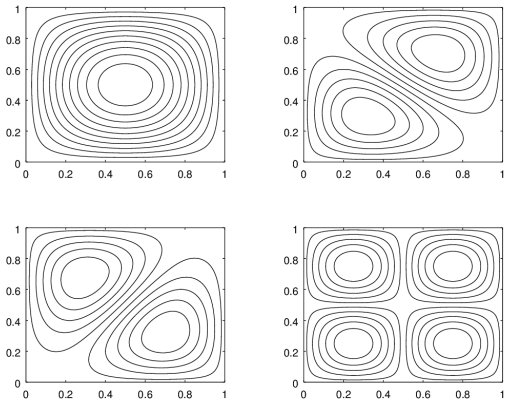
```
>> N=100;
>> h=1/N;
>> x=0:h:1;
>> [x,y]=meshgrid (x);
```

   4.2 Calculamos la representación matricial de la discretización:

```
>> alfa2=1;
>> L1d=spdiags (ones(N-1,1)*[1 -2 1],-1:1,N-1,N-1);
>> L1d=1/h^2*L1d;
>> E=speye(N-1);
>> L2d=kron(E,L1d)+kron(L1d,E);
>> L2d=-alfa2*L2d;
```

   4.3 Reolvemos el problema matricial del paso [4.2] y graficamos la solución numérica de los cuatro modos de vibración básicos:

```
>> [v,l]=eigs(L2d,4,0);
>> colormap([0 0 0]);
>> W=zeros(N+1);
>> for k=1:4, W(2:N,2:N)=reshape (v(:,k),N-1,N-1);...
subplot(2,2,k);contour(x,y,W);end
```



5.2 Ejercicios de Práctica

1. Calcular la discretización correspondiente y resolver el problema de deflexión de membranas implementando MDF en Octave.

$$\begin{cases} \partial_t^2 u - 4(\partial_x^2 u + \partial_y^2 u) = 0, (x, y, t) \in [0,1]^2 \times [0, \infty) \\ u(0, y, t) = u(1, y, t) = 0 \\ u(x, 0, t) = u(x, 1, t) = 0 \\ u(x, y, 0) = sen(\pi m x) sen(\pi m y) \\ \partial_t u(x, y, 0) = 0 \end{cases}$$

2. Calcular la discretización correspondiente y resolver el problema de Helmholtz correspondiente al modelo de deflexión de varillas implementando MDF en Octave.

$$\begin{cases} \partial_t^2 u - 4(\partial_x^2 u + \partial_y^2 u) = 0, (x, y, t) \in [0,1]^2 \times [0, \infty) \\ u(0, y, t) = u(1, y, t) = 0 \\ u(x, 0, t) = u(x, 1, t) = 0 \\ u(x, y, 0) = u_0(x, y) \\ \partial_t u(x, y, 0) = v_0(x, y) \end{cases}$$

3. Calcular la discretización correspondiente y resolver el problema de deflexión **amortiguada/incrementada** de varillas implementando MDF en Octave.

$$\begin{cases} \partial_t^2 u + c\partial_t u - 4(\partial_x^2 u + \partial_y^2 u) = 0, (x, y, t) \in [0,1]^2 \times [0, \infty) \\ u(0, y, t) = u(1, y, t) = 0 \\ u(x, 0, t) = u(x, 1, t) = 0 \\ u(x, y, 0) = sen(\pi m x) sen(\pi m y) \\ \partial_t u(x, y, 0) = 0 \end{cases}$$

Idea: Reformular las ecuaciones dinámicas y la discretización del caso simple de la práctica 1, modificando convenientemente el archivo `*.m` correspondiente a la varilla simple de la práctica 1. Cuál es el efecto de un coeficiente c positivo/negativo/cero?

6 Análisis de Modelos Estáticos por Método de Elementos Finitos

6.1 Modelos en 2D:

Consideremos un modelo descrito por la ecuación diferencial parcial:

$$\begin{cases} -\Delta u = -(\partial_x^2 u + \partial_y^2 u) = f(x,y), (x,y) \in [0,1]^2 \\ u(x,y) = 0, (x,y) \in \partial[0,1]^2 \end{cases}$$

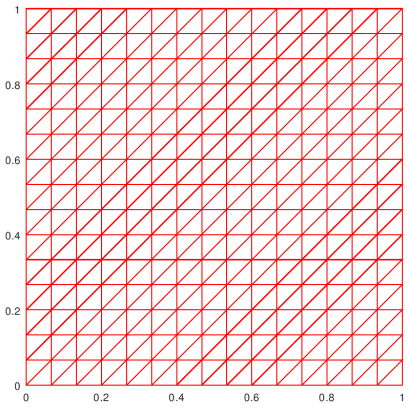
Utilizando la herramienta `bim` de GNU Octave podemos resolver varias formas particular del problema base anterior. A continuación presentamos los procedimientos correspondientes a algunos casos especiales.

Procedimiento MEF para la solución del modelo estático de bordes rígidos:

$$\begin{cases} -\Delta u = -(\partial_x^2 u + \partial_y^2 u) = 1, (x,y) \in [0,1]^2 \\ u(x,y) = 0, (x,y) \in \partial[0,1]^2 \end{cases}$$

- Creación de la malla MEF de $[0,1]^2$:

```
>> pkg load bim
>> n=m=16;
>> x=linspace (0,1,n);
>> y=linspace (0,1,m);
>> mesh = msh2m_structured_mesh(x,y,1,[1 2 3 4]);
>> mesh = bim2c_mesh_properties (mesh);
>> msh2p_mesh (mesh)
>> axis square
```



- Cálculo de la matriz estructural de coeficientes:

```
>> A = bim2a_laplacian (mesh, 1, 1);
```

- Cálculo del vector de flexión:

```
>> b = bim2a_rhs(mesh,1,1);
```

- Configuración de los grados de libertad de la estructura geométrica del problema:

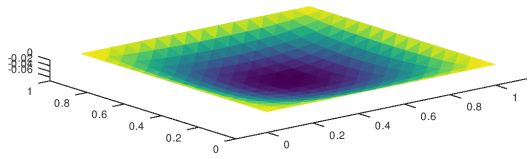
```
>> dnodes=bim2c_unknowns_on_side (mesh,1:4);
>> inodes=setdiff (1:columns(mesh.p),dnodes);
```

- Solución del sistema lineal correspondiente para calcular la configuración geométrica (deflexión) estructural (tomando el tiempo de cómputo en el proceso):

```
>> u=zeros(columns(mesh.p),1);
>> tic,u(inodes) = -A(inodes, inodes) \ (b(inodes)...
- A(inodes, dnodes) * u(dnodes));toc
Elapsed time is 0.00197601 seconds.
```

- Visualización de la configuración geométrica (deflexión) estructural:

```
>> pdesurf (mesh.p, mesh.t, u)
>> shading flat
>> axis equal
>> view(3)
```



Procedimiento MEF para la solución del modelo estático de bordes rígidos:

$$\begin{cases} -\Delta u = -(\partial_x^2 u + \partial_y^2 u) = 1, (x,y) \in L \\ u(x,y) = 0, (x,y) \in \partial L \end{cases}$$

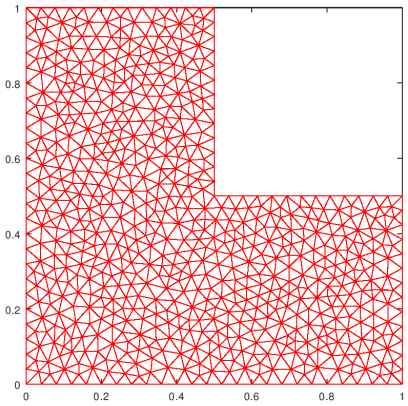
donde $L = ([0,1] \times [0,1/2]) \cup ([0,1/2] \times [0,1])$.

- Construcción de la geometría de la región L, creando el código del archivo `L.geo` (utilizando cualquier editor de texto plano conveniente):

```
h=.4;

Point (1)  = {0, 0, 0, h};
Point (2)  = {1, 0, 0, h};
Point (3)  = {1, 0.5, 0, h};
Point (4)  = {0.5, 0.5, 0, h};
Point (5) = {0.5,1,0,h};
Point (6) = {0,1,0,h};

Line (1)  = {1,2};
Line (2) = {2,3};
Line (3) = {3,4};
Line (4) = {4,5};
Line (5) = {5,6};
Line (6) = {6,1};

Line Loop(1) = {1,2,3,4,5,6};
Plane Surface(1) = {1};
```

- Creación de la malla MEF de L:

```
>> mesh = msh2m_gmsh ("L","scale",1,"clscale",.1);

Generating mesh...
Processing gmsh data...
Creating PDE-tool like mesh...
Check for hanging nodes...
Setting region number in edge structure...
Deleting temporary files...
>> mesh = bim2c_mesh_properties (mesh);
>> msh2p_mesh (mesh)
>> axis square
```



- Cálculo y visualización de la matriz estructural de coeficientes:

```
>> A = bim2a_laplacian (mesh, 1, 1);
```

- Cálculo del vector de flexión:

```
>> b = bim2a_rhs(mesh,1,1);
```

- Configuración de los grados de libertad de la estructura geométrica del problema:

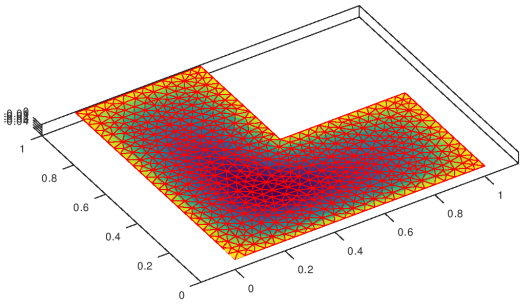
```
>> dnodes=bim2c_unknowns_on_side (mesh,1:6);
>> inodes=setdiff (1:columns(mesh.p),dnodes);
```

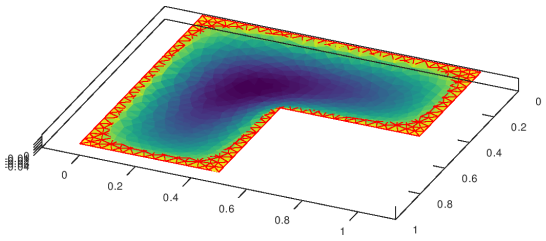
- Solución del sistema lineal correspondiente para calcular la configuración geométrica (deflexión) estructural (tomando el tiempo de cómputo en el proceso):

```
>> u=zeros(columns(mesh.p),1);
>> tic,u(inodes) = -A(inodes, inodes) \ (b(inodes)...
- A(inodes, dnodes) * u(dnodes));toc
Elapsed time is 0.00734806 seconds.
```

- Visualización de los perfiles supeior e inferior de la configuración geométrica (deflexión) estructural combinada con la configuración original (antes de la deformación estructural):

```
>> pdesurf (mesh.p, mesh.t, u)
>> shading flat
>> axis equal
>> view (3)
```





6.2 Ejercicios de Práctica

1. Aplicar el MEF con $h = 0.1$ para resolver numéricamente el problema:
$$\begin{cases} -\Delta u = -(\partial_x^2 u + \partial_y^2 u) = 1, (x, y) \in I \\ u(x, y) = 0, (x, y) \in \partial I \end{cases}$$
donde $I = ([-1, 1] \times [0, 1/4]) \cup ([-1/8, 1/8] \times [0, 1]) \cup ([-1, 1] \times [3/4, 1])$.
2. Aplicar el MEF con $h = 0.1$ para resolver numéricamente el problema:
$$\begin{cases} -\Delta u = -(\partial_x^2 u + \partial_y^2 u) = 1, (x, y) \in R \\ u(x, y) = 0, (x, y) \in \partial R \end{cases}$$
donde $R = [0, 2] \times [0, 1/2]$.

References

[1] Quarteroni A., Saleri F., Gervasio P. (2014). *Scientific Computing with MATLAB and Octave.* Texts in Computational Science and Engineering, Springer.

[2] L. N. Trefethen. (2000). *Spectral Methods in Matlab.* https://doi.org/10.1137/1.9780898719598