Experiencia docente con Octave UPM

Carlos Castro

carlos.castro@upm.es
Universidad Politécnica de Madrid

Idea

Introducir el cálculo científico como una herramienta más en la docencia y aprendizaje aprovechando

1. Las nuevas tecnologías que han hecho del ordenador una herramienta común (moodle)
2. El software de cálculo científico (tipo MATLAB) que es cada vez más fácil de usar y accesible. OCTAVE UPM es similar a MATLAB y libre.

POLITÉCNICA
"Ingeniamos el futuro"

¿Por qué es beneficioso el cálculo científico en la docencia?

- Informática: programación
- Cálculo: visualización de funciones, aproximación funcional, optimización, etc.
- Álgebra: cálculo matricial, sistemas lineales, geometría lineal
- Estadística: análisis de datos, test estadísticos
- Mecánica: simulación y visualización de complejos sistemás mecánicos.
- etc.

POLITÉCNICA
"Ingeniamos el futuro"

Principales inconvenientes del cálculo científico en la enseñanza

1. Requiere salas de ordenadores que son costosas
2. Organización de los cursos en pequeños grupos para acceder a las aulas de ordenadores
3. Software de cálculo avanzado, que es en general costoso
4. Aulas de estudio para trabajo personal
5. Dificultad para acceder a medios materiales que permitan un aprendizaje práctico
6. Adaptar la enseñanza al uso del cálculo científico como herramienta
7. Curso de iniciación al uso de ordenadores y software específico que no suele estar recogido en los actuales planes de estudio
8. Coordinación entre diferentes asignaturas para que lo aprendido en unas pueda servir en otras

POLITÉCNICA

"Ingeniamos el futuro"

OCTAVE UPM: ¿Qué mejoramos y qué no mejoramos?

- **Ventajas**
  1. Fácil instalación y acceso en el ordenador propio
  2. Hace más accesible el cálculo científico como herramienta de enseñanza y estudio
  3. Puede usarse en aulas normales. En general se requieren mínimos cambios: más tomas de corriente y WIFI
  4. El trabajo en casa es más sencillo puesto que los alumnos usan su propio ordenador

POLITÉCNICA
"Ingeniamos el futuro"

- **Inconvenientes**
  1. Los alumnos tienen que traer sus propios ordenadores al aula
  2. Requiere más atención por parte de los profesores (1 profesor cada 20 alumnos)
  3. Adaptar la enseñanza al uso del cálculo científico como herramienta
  4. Curso de iniciación que no suele estar recogido en los actuales planes de estudio
  5. Coordinación entre diferentes asignaturas para que lo aprendido en unas pueda servir a otras

POLITÉCNICA
"Ingeniamos el futuro"

Una propuesta

- **Enseñanza global:**
  1. Incluir un curso introductorio de MATLAB/OCTAVE en el primer semestre que incluya nociones de programación
  2. Introducir en cada curso donde el cálculo científico sea útil una serie de prácticas guiadas que puedan hacerse en clase y en casa.

POLITÉCNICA
"Ingeniamos el futuro"

Ejemplo: programa del GICyT

- Primer semestre
  - Cálculo I
  - Álgebra
  - Informática
  - Empresa
  - Expresión gráfica
- Segundo semestre
  - Cálculo II
  - Estadística y optimización
  - Diseño gráfico
  - Química de materiales
  - Física
- Tercer semestre
  - Teoría de Campos
  - Geología
  - Inglés
  - Física de sólidos y fluidos
  - Materiales de construcción I
  - Topografía y cartografía
- cuarto semestre
  - Ecuaciones diferenciales
  - Electrotecnia
  - Mecánica
  - Geología aplicada
  - Materiales de construcción II
  - Resistencia de materiales

POLITÉCNICA
"Ingeniamos el futuro"

- **En cada asignatura:**
  1. **Tiempo semanal** de clases para la programación y visualización a través de prácticas guiadas.
  2. Estas clases prácticas se pueden hacer en aulas normales. **Los alumnos traen sus propios ordenadores** (por grupos)
  3. Requiere el **apoyo de otros profesores** para resolver las numerosas dudas.
  4. Requiere unas **prácticas guiadas** adaptadas a los contenidos teóricos de las asignaturas.

POLITÉCNICA

"Ingeniamos el futuro"

Ejemplo: Programación

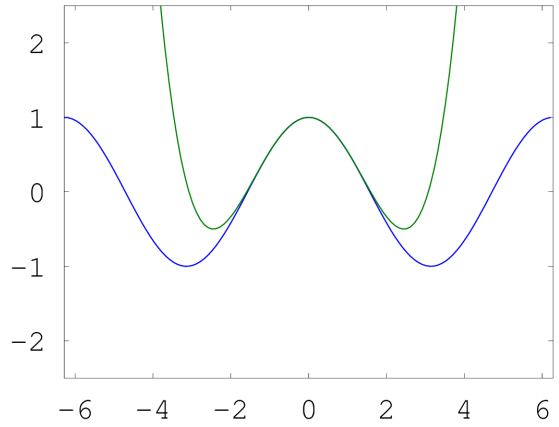
Programa en MATLAB para conocer si un número es primo

```
1 n=input('introduce un numero natural: ');
2 t='el numero es primo';
3 for i=2:floor(sqrt(n))
4   if rem(n,i)~=0
5       continue
6   end
7   t='el numero no es primo';
8 end
9 disp(t)
```

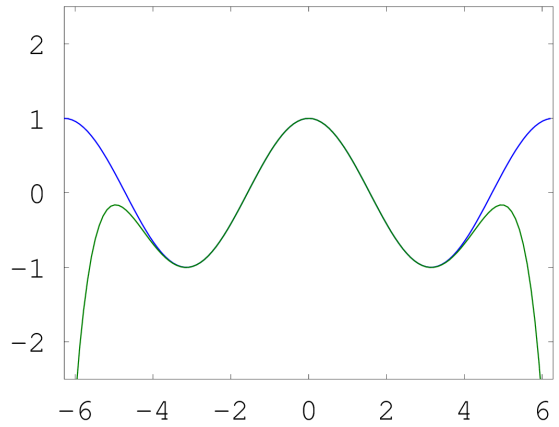
Ejemplo: Aplicaciones al Cálculo

Aproximar una función por su polinomio de Taylor

```
1  % programa para aproximar la funcion cos(x)
     con el polin de Taylor en x=0
2  x=-2*pi:0.1:2*pi;
3  fx=cos(x);
4  fy=1;
5  n=0;
6  while n<=2
7     n=n+1;
8     fy=fy+(-1)^n*x.^(2*n)/(factorial(2*n));
9  end
10 plot(x,[fx;fy],'LineWidth',2)
11 axis([-2*pi,2*pi,-2.5,2.5])
12 set(gca,'FontSize',16)
```



Polin. grado 4 en $x = 0$



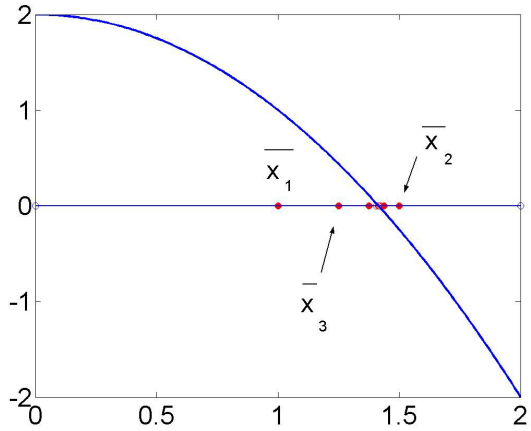
Polin. grado 8 en $x = 0$

POLITÉCNICA
"Ingeniamos el futuro"

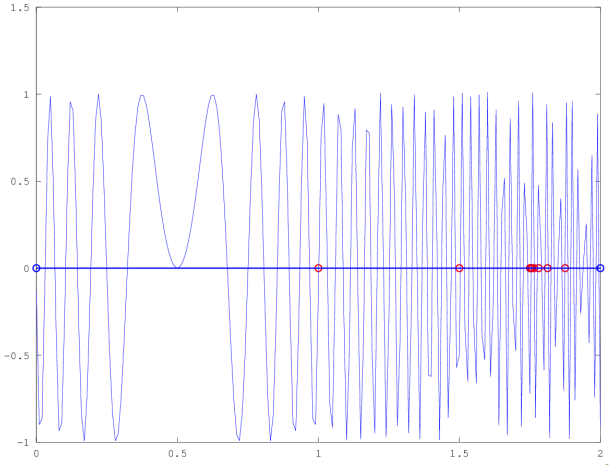
Ejemplo: Aplicaciones al Cálculo

Método de bisección para aproximar raíces

```
1  % programa para aproximar raices
2  a=0;
3  b=2;
4  ep=1.e-3;
5  J=floor(log2((b-a)/ep));
6  for n=1:J
7    if func((a+b)/2) < 0
8       a=(a+b)/2;
9    elseif func((a+b)/2) > 0
10      b=(a+b)/2;
11   else
12      resultado=(a+b)/2
13      break
14   end
15 end
16 x=0:.01:2;
17 plot(x,func(x),'LineWidth',1)
18 resultado = (a+b)/2
```



Ejemplo 1



Ejemplo 2

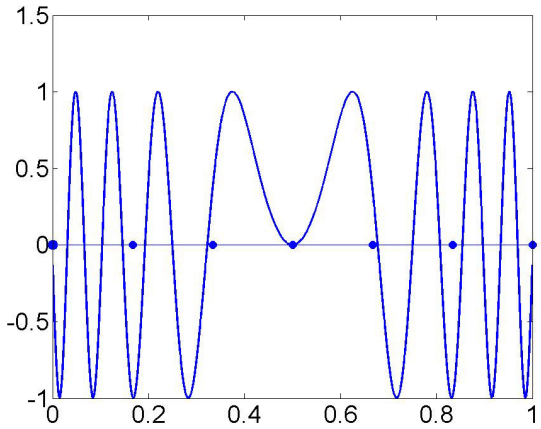
POLITÉCNICA

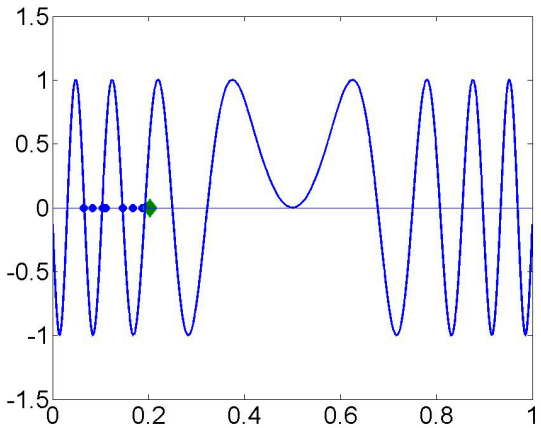
"Ingeniamos el futuro"

Ejemplo: Aplicaciones al Cálculo

Algoritmo genético para aproximar raíces

```
1  % Algoritmo genetico
2  % Seleccion inicial
3  n=10;
4  h=1/(n+1);
5  x=0:h:1;
6  z=0:0.0001:1;
7  % Iteraciones
8  for j=1:10
9      % Seleccion
10     fun=funcion(x);
11     [funi i]=min(fun);
12     [funM k]=max(fun);
13     % Reproduccion
14     x=.5*(x(i)+x);
15     y=rand;
16     x(k)=y;
17 end
18 Solucion=x(i)
```



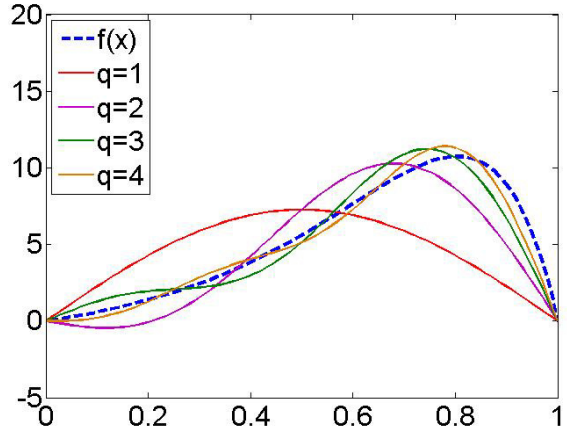


POLITÉCNICA
"Ingeniamos el futuro"

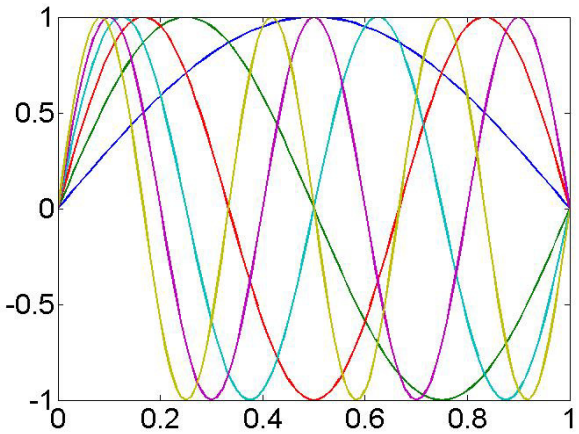
Ejemplo: Aplicaciones al Álgebra

Aproximación por mínimos cuadrados

```
1  % Trigonometric approximation
2  q=4; h=0.01;
3  x=0:h:1;
4  f=(exp(5*x)-1).*(1-x);
5  ap=0*x;
6  for n=1:q
7     cn=h*sum(f.*sin(n*pi*x))*2;
8     ap=ap+cn*sin(n*pi*x);
9  end
10 plot(x,ap,'r')
```



Aproximación



Base trigonométrica

Ejemplo: Aplicaciones al Álgebra

Factorización LU

```
1  % Factorizacion LU
2  function [L,U]=factlu(A)
3  [n,m]=size(A);
4  U=A; L=eye(n);
5  for k=1:n-1
6    for j=k+1:n
7      L(j,k)=U(j,k)/U(k,k);
8      U(j,k:n)=U(j,k:n)-L(j,k)*U(k,k:n);
9    end
10 end
```

$U = A,\ L = I$

For $k = 1, ..., n-1$

  For $j = k+1, ..., n$

    $l_{jk} = u_{jk}/u_{kk}$

    $u_{j,k:m} = u_{j,k:m} - l_{jk} u_{k,k:m}$

  end

end

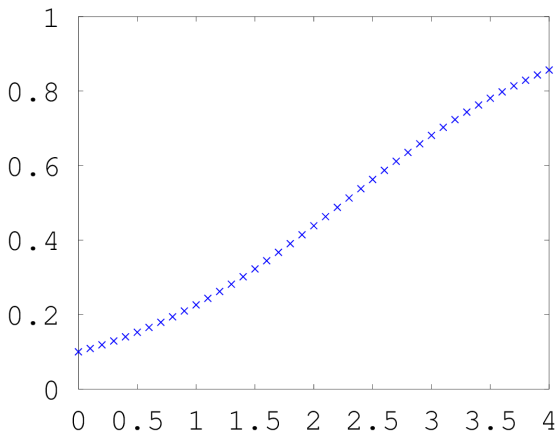
POLITÉCNICA

"Ingeniamos el futuro"

Ejemplo: Ecuaciones diferenciales

Ecuación logística

$$\begin{cases} y' = y(1-y), \\ y(0) = 1/10. \end{cases}$$




```
% Metodo de Euler
t0=0; tN=4;
y0=1/10;
N=40; h=(tN-t0)/40;
y(1)=y0;
for n=1:N
   y(n+1)=y(n)+h*y(n)*(1-y(n));
end
x=t0:h:tN;
plot(x,y,'x');
```

POLITÉCNICA
"Ingeniamos el futuro"

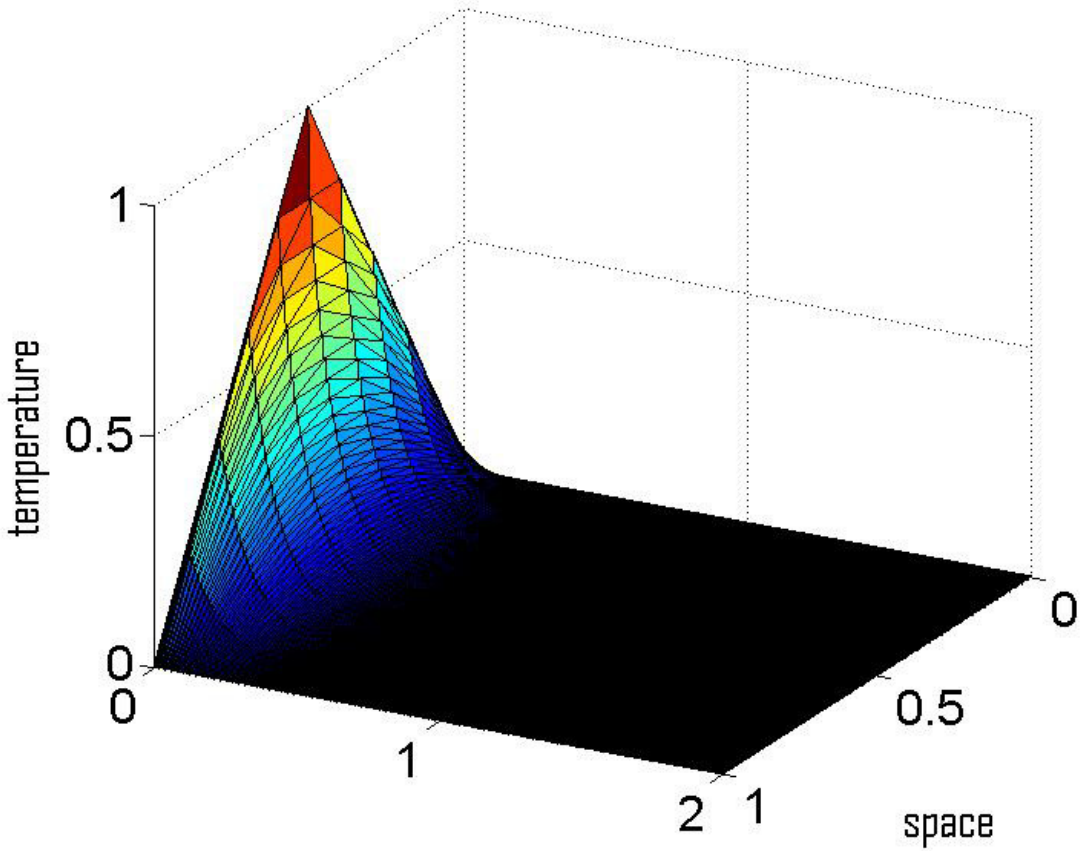
Ejemplo: Ecuaciones diferenciales

Ecuación del calor por diferencias finitas

```
1  L = 1; dx=1/10; N=L/dx;  % space mesh
2  K = 2*diag(ones(1,N-1))-diag(ones(1,N-2),1)-diag(ones(1,N-2),-1);
3  K = (1/dx^2)*K;          % matrix formulation -u_{xx}
4  x=dx:dx:L-dx; U=(1-2*abs(x-1/2))';  % initial datum
5  mu=1/2; T=2; dt=mu*dx^2; t=0:dt:T;  % time mesh
6  sol(1,:)=[0 U' 0];       % solution at time t=0
7  for j=1:length(t)-1      % time loop
8    U=U-dt*K*U;
9    sol(j+1,:)=[0 U' 0];  % save solution in 'sol' variable
10 end                      % time loop
11 x1=0:dx:L;
12 [xx,tt]=meshgrid(x1,t);  % (x,t) coordinates of the mesh
13 surf(xx,tt,sol)          % plot solution
```

POLITÉCNICA

"Ingeniamos el futuro"



Solución con el método de Euler

Courant number $\Delta t/h^2 = 1/2$.

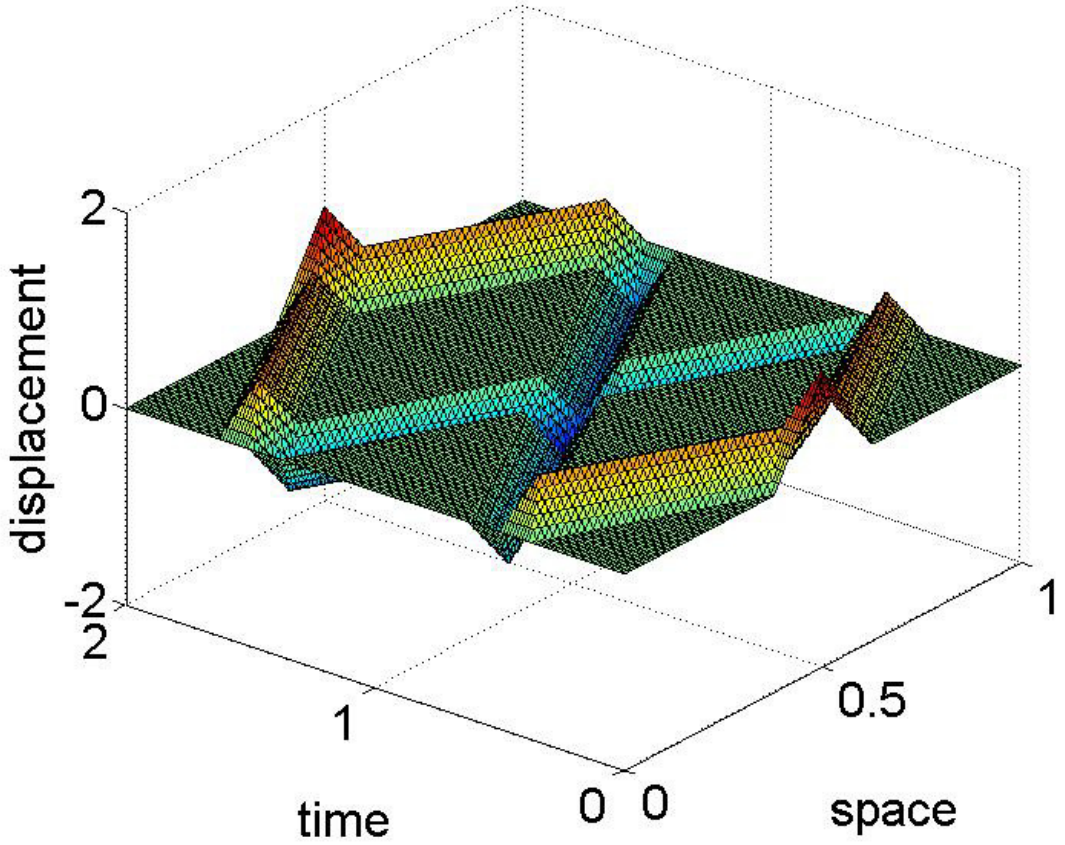
Ejemplo: Ecuaciones diferenciales

Ecuación de ondas por diferencias finitas

```
L = 1; dx=1/80; N=L/dx;
 % K matrix approximation of -u_xx
K = 2*diag(ones(1,N-1))-diag(ones(1,N-2),1)-diag(ones(1,N-2),-1);
K = (1/dx^2)*K;
 % Define the initial datum
x=dx:dx:L-dx;  % coordinates of internal space points
U=max(0,(1-8*abs(x'-1/2)));
V=0*sign(x'-1/2).*(abs(x'-1/2)<=1/8)*8;
 % Time discretization
mu=1; T=2; dt=mu*dx; t=0:dt:T;  % time mesh
sol(1,:)=[0 U' 0];  % solution at t=0
for j=1:length(t)-1
  U=U+dt*V;       % Euler explicit
  V=V-dt*K*U;
  sol(j+1,:)=[0 U' 0];
end
x1=0:dx:L;  % space mesh
[xx,tt]=meshgrid(x1,t); surf(xx,tt,sol)
```

POLITÉCNICA

"Ingeniamos el futuro"

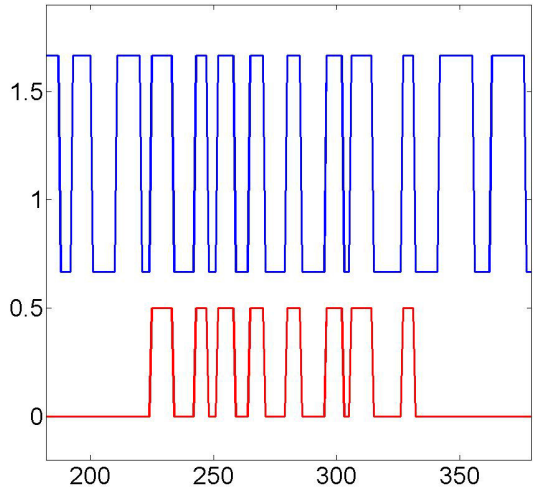


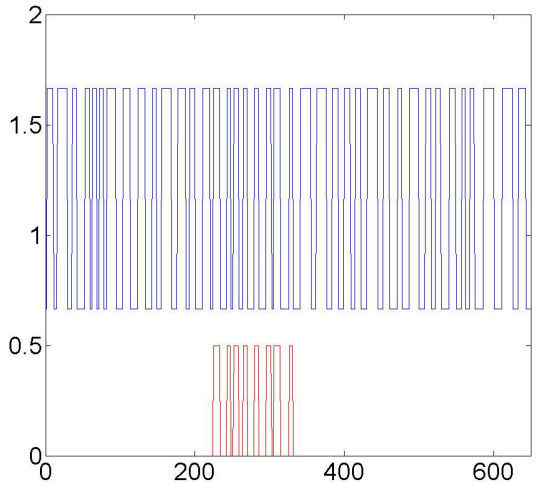
Solución con el método de Euler

Courant number $\Delta t/h = 1/2$.

Aplicación: Reconocimiento de patrones.

Problema: Encontrar la posición en la que encaja el patrón de abajo en la serie de arriba





1. encontrar una función objetivo que nos permita cuantificar el grado de encaje y
2. calcular el óptimo de la función objetivo.

POLITÉCNICA
"Ingeniamos el futuro"

Dos métodos:

1. Evaluación directa de la $fo(n)$. Convolución que requiere N^2 operaciones producto-suma.
2. Uso de DFT. Requiere $4N\log_2(N)$ operaciones suma-producto.
   1. Calculamos la FFT de $x(n)$: $X(n)$
   2. Calculamos la FFT de $y(n)$: $Y(n)$
   3. Multiplicamos $FO(n) = X(n).*\overline{Y(n)}$
   4. Calculamos la IFFT de $FO(n)$

Si $N = 10^4$, $N^2 = 10^8$ mientras que $N\log_2(N) < 14 * 10^4$.

Uso de la FFT

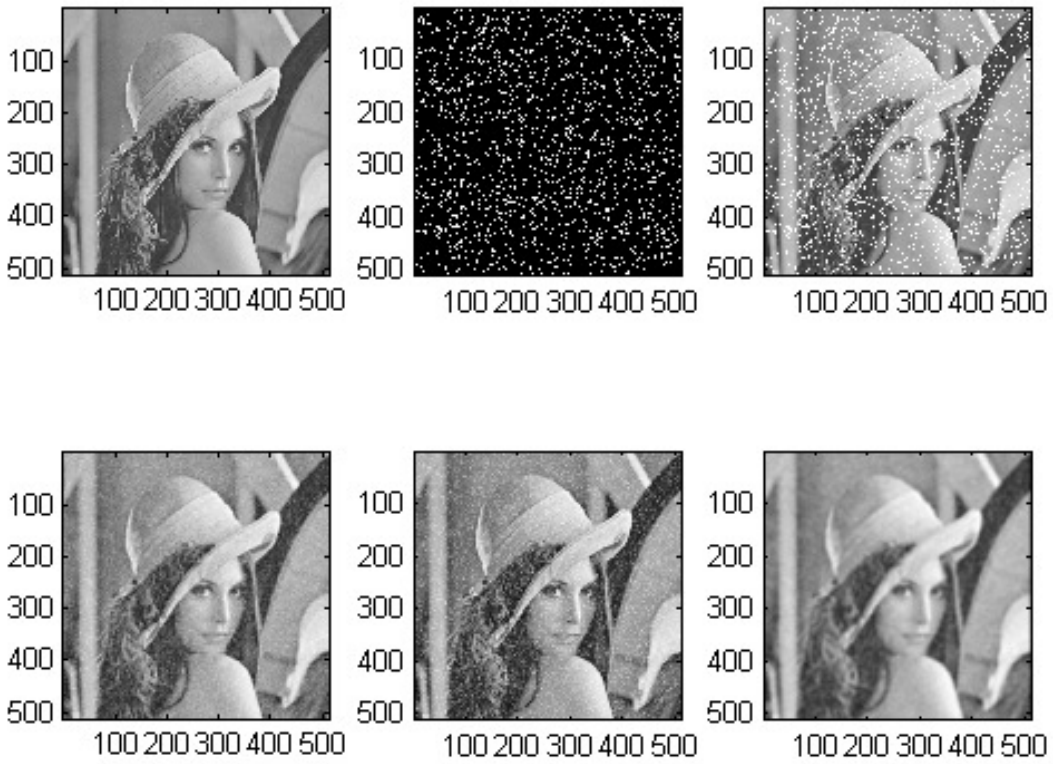
```
1  % Creacion de las muestras
2  % p tamanyo de la muestra grande
3  % q tamanyo de la submuestra
4  p=100;
5  q=4;
6  [A B]=aleatorio(p,q);
7  % Calculo de las fft
8  fta=fft(A);
9  ftb=fft(B);
10 % Producto
11 ftc=conj(ftb).*fta;
12 C=ifft(ftc);
13 % Dibujamos la funcion objetivo
14 figure(4)
15 plot(C,'x')
16 % Calculo de la posicion optima
17 [val pos]=max(C);
```

POLITÉCNICA

"Ingeniamos el futuro"

Aplicación: Filtros para tratamiento de imágenes

Filtro regularizante para eliminar ruido



```
% Importamos la imagen
lena=imread('lena.jpg');

% Dibujamos la imagen
subplot(231); imagesc(lena);
colormap('gray'); axis('image')
set(gca,'units','pixels','DataAspectRatio',[1 1 1]);

% Generamos nieve en la imagen
R=ceil(max(rand(512,512)-0.9,0));
lena=lena.*(1-cast(R,'uint8'));
lena=255-(255-lena).*(1-cast(R,'uint8'));

% Dibujamos el ruido
subplot(232); imagesc(R);
colormap('gray'); axis('image')
set(gca,'units','pixels','DataAspectRatio',[1 1 1]);

% Dibujamos la imagen con ruido
subplot(233); imagesc(lena);
colormap('gray'); axis('image')
set(gca,'units','pixels','DataAspectRatio',[1 1 1]);
```

POLITÉCNICA

"Ingeniamos el futuro"

```
1
2  % Filtro rectangular
3  h=ones(5);
4  h=h/sum(sum(h));
5
6  % Dibujo de la imagen filtrada
7  pixr=filter2(h,lena); subplot(234); imagesc(pixr);
8  set(gca,'units','pixels','DataAspectRatio',[1 1 1]);
9  axis('image')
10
11 % Filtro conico
12 h=[ 0 0 1 0 0 ;
13      0 2 2 2 0 ;
14      1 2 5 2 1 ;
15      0 2 2 2 0 ;
16      0 0 1 0 0 ];
17 h=h/sum(sum(h));
```

POLITÉCNICA

"Ingeniamos el futuro"

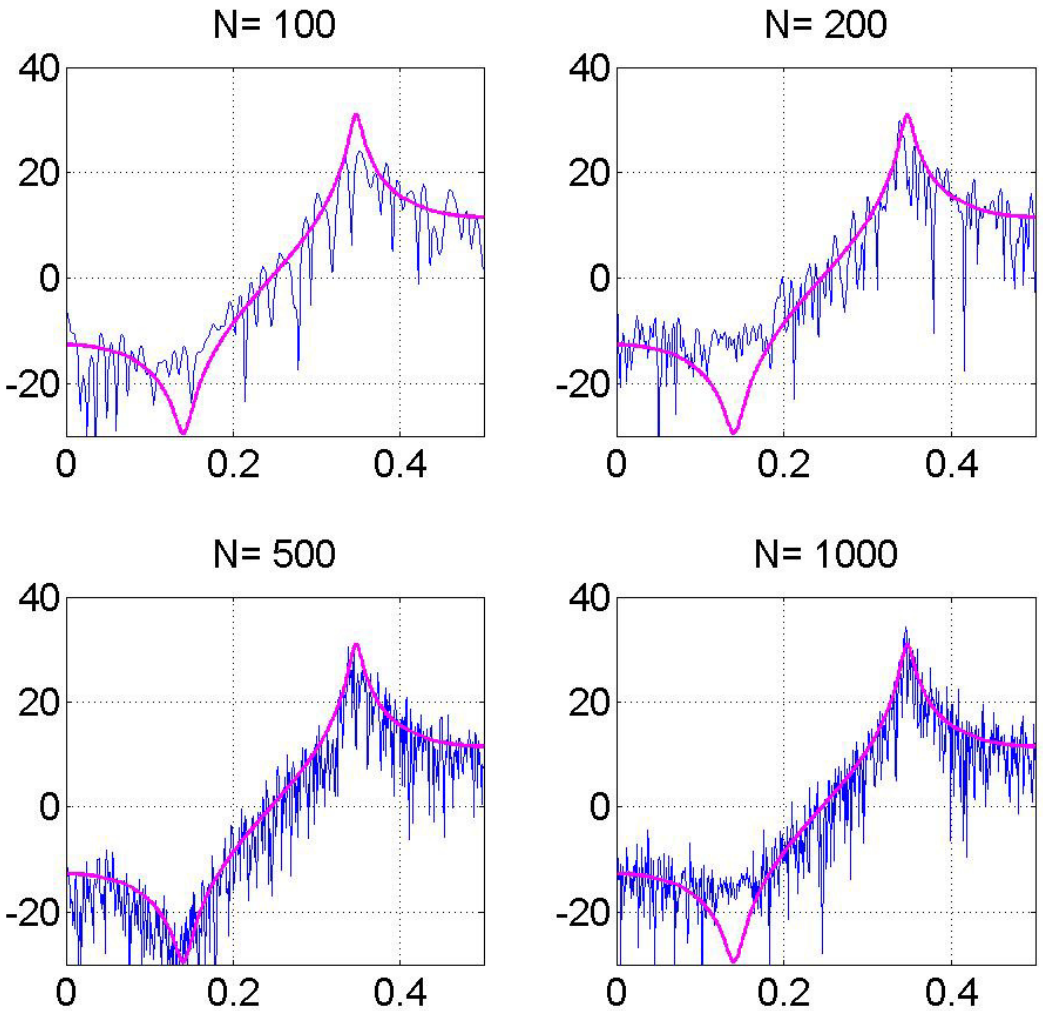
```
% Dibujo de la imagen filtrada
pixr=filter2(h,lena); subplot(235); imagesc(pixr);
set(gca,'units','pixels','DataAspectRatio',[1 1 1]);
axis('image')

% Filtro Gaussiano
h=sgauss(3);

% Dibujo de la imagen filtrada
pixr=filter2(h,lena); subplot(236); imagesc(pixr);
set(gca,'units','pixels','DataAspectRatio',[1 1 1]);
axis('image')
```

POLITÉCNICA

"Ingeniamos el futuro"

Aplicación: Análisis de series temporales

Comparación entre el periodograma y el valor real de la PSD para el proceso aleatorio ARMA(2,2). N es el número de valores de la serie temporal.



```
% Fluctperio.m
Lfft=1024;
tau=(0:Lfft-1)/Lfft-.5;
w=randn(1,1000);
b=[1 1.2 0.9]; a=[1 -1.1 0.92];
PSDth=20*log10(abs(fftshift(fft(b,Lfft))./fftshift(fft(a,Lfft))));
x=filter(b,a,w);
lxt=[100 200 500 1000];
for ii=1:length(lxt)
  xt=x(1:lxt(ii));
  per=20*log10(abs(fftshift(fft(xt,Lfft))))-10*log10(lxt(ii));
  subplot(2,2,ii); plot(tau,per,tau,PSDth,'m');grid
  axis([-0.5 0.5 -30 40]);
  title(sprintf('N= %d',lxt(ii)));
end
```