Examen de Prácticas (A). Convocatoria Julio 2011

NOMBRE Y APELLIDOS:

1. **(1.25 ptos)** Un armador de barcos tiene un barco con capacidad de hasta 700 toneladas. El carguero transporta contenedores de diferentes pesos para una determinada ruta. En la ruta actual, el carguero puede transportar algunos (no todos, pues el peso total excede las 700 toneladas permitidas) de los siguientes contenedores:

Contenedor	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
Peso	100	155	50	112	70	80	60	118	110	55

Se trata de averiguar cuáles de los anteriores 11 contenedores se han de trasnportar de modo que se maximize la carga transportada (es decir, el peso total) y de modo que ésta no exceda la capacidad máxima del barco de 700 toneladas. Si introducimos la variable

$$x_j = \begin{cases} 1 & \text{si el contenedor } c_j \text{ se carga} \\ 0 & \text{en caso contrario} \end{cases}$$

el problema se formula matemáticamente como

$$\begin{cases} \text{Maximizar} & f(x_1, \cdots, x_{10}) = 100x_1 + 155x_2 + 50x_3 + 112x_4 + 70x_5 + 80x_6 \\ & \qquad +60x_7 + 118x_8 + 110x_9 + 55x_{10} \\ \text{sujeto a} & \\ & 100x_1 + 155x_2 + 50x_3 + 112x_4 + 70x_5 + 80x_6 + \\ & +60x_7 + 118x_8 + 110x_9 + 55x_{10} \leq 700. \end{cases}$$

Se trata pues de un problema de programación lineal entera (de hecho binaria). MATLAB dispone de la función `bintprog` para resolver este tipo de problemas. En concreto, dado el problema de programación lineal binaria

$$\begin{cases} \text{Minimizar} & f(x) = f^T \cdot x \\ \text{sujeto a} & \\ & Ax \leq b \\ & A_{eq}x = b_{eq} \end{cases}$$

la sintaxis de la función `bintprog` es la siguiente:

$$[x, fval, exitflag] = \texttt{bintprog}(f, A, b, A_{eq}, b_{eq})$$

con el mismo significado que para los problemas de programación lineal. Más información sobre la función `bintprog` se puede obtener escribiendo `help bintprog` en la ventana Command Window de MATLAB.

Utiliza la función `bintprog` para resolver el problema anterior de los contenedores. Recuérdese que Maximizar $f(x)$ =Minimizar $-f(x)$.

Se han de entregar tanto los resultados como los códigos en MATLAB que son necesarios para resolver el problema.

Solución. Ejecutamos el código

```
f = [-100,-155,-50,-112,-70,-80,-60,-118,-110,-55];
A =[100,155,50,112,70,80,60,118,110,55];
b = [700];
[x,fval,exitflag] = bintprog(f,A,b)
```

y se obtiene la solución

```
x =
1
0
1
1
1
1
1
1
1
0
fval =
-700
exitflag =
1
```

2.

3. **(1.75 Ptos)** Elabora un código de elementos finitos en MatLab para resolver el siguiente problema:

$$\begin{cases} -u''(x) + (1+x^2)\, u(x) = 3 - x^4, & 0 < x < 1 \\ u(0) = 1 \\ u'(1) = -2. \end{cases}$$

La solución exacta de este problema es

$$u(x) = 1 - x^2.$$

Analiza cómo evoluciona el error entre la solución numérica obtenida con el código de elementos finitos y la solución exacta al considerar 10, 100 y 1000 elementos. Se ha de responder a esta pregunta escribiendo el código con bolígrafo (o indicando los cambios realizados sobre el código elfin.m), escribiendo los resultados del error y dibujando de manera aproximada las gráficas de las soluciones numérica y exacta.

`Solución.` Modificamos ligeramente el código elfin.m explicado en clase de modo que queda del siguiente modo:

```
function [vector_x,vector_u]=elfin_exa11(b,c,f,a1,b1,alfa,beta,num_pas)
% [vector_x,vector_u]=elfin(b,c,f,a1,b1,alfa,beta,iopc,num_pas)
% elfin calcula la solución discreta del problema de contorno
% lineal de segundo orden:
%******************************************************
% -u”+b(x)u’+ c(x)u=f(x), a1<x<b1
% u(a1) =alfa, u’(b1) =beta
%
%******************************************************
% utilizando elementos finitos de Lagrange de grado 1.
if(num_pas < 3)& (iopc==0)
vector_x=[a1,b1];vector_u=[alfa,beta];
return
end
%************************************
% Construcción de la malla 1-dimensional.
%************************************
nver=num_pas+1; % numero de vertices
nel=num_pas; % numero de elementos
h=(b1-a1)/num_pas; % tamaño de la discretizacion.
i=[1:num_pas+1]; vector_x=a1+(i-1)*h;
%*******************************************
% Inicializacion de la matriz Ah
%*******************************************
Ah=spalloc(nver,nver,3*nver-2);
%********************************************
% Inicializacion de bh
%********************************************
```

```
bh=zeros(nver,1);
%**********************************************
% Calculo de las longitudes caracterisiticas de cada elemento
%**********************************************
long=diff(vector_x);
pmed=(vector_x(1:nver-1)+vector_x(2:nver))/2;
v_b=feval(b,pmed);v_c=feval(c,vector_x);
v_f=feval(f,pmed);
%**********************************************
% Bucle en elementos
%**********************************************
for k=1:nel
%**********************************************
% Calculo matriz elemental
%**********************************************
Ahk=(1/long(k))*[1, -1;-1, 1]+(v_b(k)/2)*[-1, 1; -1, 1] ...
+(long(k)/2)*[v_c(k), 0; 0, v_c(k+1)];
%**********************************************
% Ensamblado de la matriz
%**********************************************
Ah(k:k+1,k:k+1)=Ah(k:k+1,k:k+1)+Ahk;
%**********************************************
% Calculo 2º miembro elemental
%**********************************************
bhk=v_f(k)*long(k)/2*[1;1];
%**********************************************
% Ensamblado del segundo miembro
%**********************************************
bh(k:k+1)=bh(k:k+1)+bhk;
end
%**********************************************
% Bloqueo de la matriz
%**********************************************
Ah(1,1)=1.e+30;%Ah(nver,nver)=1.e+30;
%**********************************************
% Bloqueo del segundo miembro
%**********************************************
bh(1)=alfa*1.e+30;%bh(nver)=beta*1.e+30;
%**********************************************
% Modificacion del segundo miembro
%**********************************************
%bh(1)=bh(1)-alfa;
bh(nver)=bh(nver)+beta;
%**********************************************
% Resolucion del sistema
%**********************************************
```

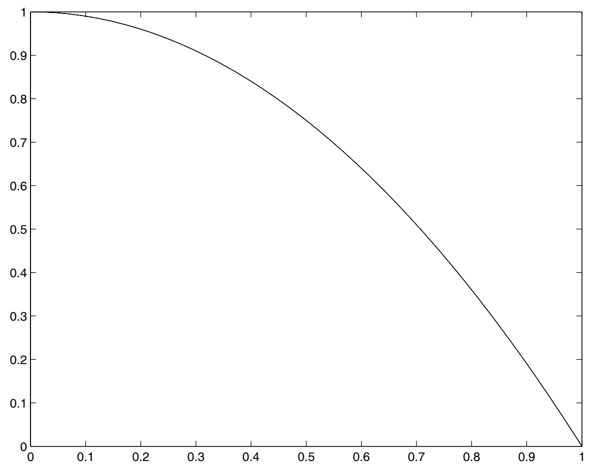

Figure 1: Solución ejercicio 2.

vector_u=(Ah\bh)';
Definimos ahora las funciones b,c y f para nuestro caso concreto:
function f = b_exa11(x)
f = zeros(size(x));
function f = c_exa11(x)
f = 1 + x.^2;
function f = f_exa11(x)
f = 3-x.^4;
Finalmente eejecutamos el código del siguiente modo:
[x,u] = elfin_exa11('b_exa11','c_exa11','f_exa11',0,1,1,-2,1000)
exacta = '1-x.^2';
plot(x,u,'b',x,eval(exacta),'r')
error = norm(abs(u-eval(exacta)),Inf)
El error que se obtiene es
error = 4.3681e-007
y la gráfica de la solución aparece en Figure 1.