Fórmula de Newton del polinomio de interpolación clásico

Dados los nodos $\{x_0, x_1, ..., x_n\}$ y los valores $\{y_0, y_1, ..., y_n\}$, **el polinomio de interpolación utilizando la fórmula de Newton tiene la expresión**

$$p(x) = d_0 + d_1(x - x_0) + d_2(x - x_0)(x - x_1) + ... + d_n(x - x_0)(x - x_1)...(x - x_{n-1})$$

donde $d_0 = f[x_0], ..., d_n = f[x_0, ..., x_n]$ son los elementos diagonales de la tabla de diferencias divididas.

Ejercicio 1. Tabla de diferencias divididas.

- Crear la función *function dd=difdiv(xi,fi)* (ver código en anexo o en diapositivas clase). La función difdiv calcula la tabla de diferencias divididas de los datos (xi,fi) y devuelve el vector dd de coeficientes de la Fórmula de Newton del polinomio de interpolación en (xi,fi).
- Calcular la tabla de diferencias divididas de los datos xi=[0 1 2]' e yi=[1 0 1]' con la función *difdiv*.
- Sea el vector xx=0:0.01:2. Evaluar el polinomio de Newton que interpola los datos (xi,yi) en el vector xx:

$$p(x) = d_0 + d_1(x - x_0) + d_2(x - x_0)(x - x_1)$$

  donde:
  - Los coeficientes d_0, d_1, d_2 son los 3 valores del vector dd de la tabla de diferencias divididas.
  - Los nodos x_0, x_1, x_2 son los valores del vector xi.
- Dibujar la gráfica del polinomio ('b') en el intervalo [0,2] y de los puntos dónde interpola ('ro').

Objetivo: Entender función difdiv (código en anexo y en diapositivas clase), saber programar y aplicar para calcular la tabla de diferencias divididas y extraer los coeficientes del polinomio de Newton.

Ejercicio 2. Polinomio de Newton usando función polNewton

Crear la función *function px=polNewton(xi,fi,xx)* (ver código en anexo). La función polNewton devuelve el vector px de los valores del polinomio de Newton (que interpola los datos (xi,fi)) en el vector xx.

Ejecutar la función px=polNewton(xi,fi,xx) con los argumentos de entrada xi, fi, xx del ejercicio anterior. Dibujar la gráfica del polinomio y los puntos donde se interpola.

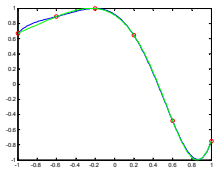
Objetivo: Saber programar una función.m que calcule la Fórmula de Newton del polinomio de interpolación y entender el código anexo al final de este documento.

Ejercicio 3. Interpolación de la función f(x)=sin(exp(x)) en un conjunto de puntos equiespaciados en el intervalo [-1,1].
a) Interpolar f(x) por el polinomio de grado mínimo en 6 puntos equiespaciados (x_{i+1}-x_i=h) en el intervalo [-1,1] ¿Cuál es el grado del polinomio? Para ello hay que construir los vectores xi e yi y resolver el correspondiente problema de interpolación. Hacer una gráfica que muestre simultáneamente el polinomio obtenido ('b'), la función f(x) ('g') y los datos del problema de interpolación ('ro') en el intervalo [-1:0.001:1]. Calcular cúal es el valor máximo del error de interpolación en los puntos xx=-1:0.01:1.
NOTA: 1. En este caso: nº de puntos=6 -> h=(1-(-1))/(n-1) distancia entre puntos consecutivos. También se puede usar comando linspace (linspace(x1,x2,n) generates n points. The spacing between the points is (x2-x1)/(n-1)).
2. Se puede utilizar la funcion polNewton(xi,fi,xx) del Anexo.

b) Vamos a repetir el apartado anterior considerando ahora n puntos equiespaciados con n=2:40, esto es calcular los correspondientes polinomios de interpolación en 2 puntos, en 3 puntos y así sucesivamente hasta 40 puntos equiespaciados en el intervalo [-1,1]. En cada caso, para cada n, calcular el valor máximo del error de interpolación y guardarlo en el vector maxerror (maxerror(n)=max(abs(f(xx)-p(xx))) sobre una muestra xx de puntos del intervalo [-1,1] (por ejemplo xx=-1:0.01:1). Representar gráficamente el máximo del error de interpolación en función del nº de puntos empleados, utilizar escala logarítmica en el eje y. Comentar el comportamiento del error al aumentar el nº de puntos de interpolación utilizados.

SOLUCIÓN:
a)
```
clear all
% Datos:
n=6;    h=2/(n-1);
xi=-1:h:1;  % xi=linspce(-1,1,6);
yi=sin(2.*exp(xi));
```

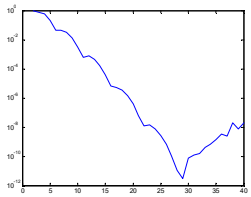
```
% Evaluación del  polinomio (usando F. Newton) y de la función sobre una muestra xx de puntos en [-1,1]. Gráfica de la función y el polinomio:
xx=-1:0.01:1;
fx=sin(2.*exp(xx));
px= polNewton(xi',yi',xx);
plot(xi,yi,'ro',xx,px,'b',xx,fx,'g')
```



```
% Error interpolación en xx. Máximo:
error_interp=(abs(fx-px));
maxerror=max(error_interp)
```

b). Gráfica. Comportamiento del error:

```
clear
for n=2:40
   h=2/(n-1);
   xi=-1:h:1;
   yi=sin(2.*exp(xi));
   px= polNewton(xi',yi',xx);
   maxerror(n)=max(abs(fx-px));
end
semilogy(2:40,maxerror(2:40))
```



Observación: El error va disminuyendo de forma exponencial (el comportamiento que se aprecia en la gráfica es lineal pero tenemos que tener en cuenta que se está usando escala logarítmica) hasta n=27, a partir de entonces, al utilizar más puntos de interpolación, el valor máximo del error crece de forma exponencial.

Ejercicio 4. Regla de Horner para evaluar el polinomio de Newton.

Una tarea computacional muy común es evaluar un polinomio p(x) en un punto x. La Fórmula de Newton del polinomio de interpolación $p(x) = d_0 + d_1(x - x_0) + ... + d_n(x - x_0)...(x - x_{n-1})$ se puede reescribir como

$$p(x) = (..((d_n(x - x_{n-1}) + d_{n-1})(x - x_{n-2}) + d_{n-2})(x - x_{n-3}) + ... + d_1)(x - x_0) + d_0$$

que sugiere la regla de Horner aplicada en este caso para evaluar el polinomio *p(x)* de forma recursiva a la Fórmula de Newton:

$$\begin{aligned} &p(x) = d_n \\ &p(x) = d_{n-1} + p(x)(x - x_{n-1}) \\ &... \\ &p(x) = d_0 + p(x)(x - x_0) \end{aligned}$$

Usando las diferencias divididas, modificar la función polNewton incluida en anexo (llamar *pol_newton_horner)* para evaluar el polinomio de Newton en los puntos del vector xx, utilizando la regla de Horner.
Comprobar la función *pol_newton_horner* interpolando los datos de la tabla del Ejercicio 1.

Ejercicio 5. Error interpolación polinomial clásica. Se considera la función $f(x) = \dfrac{1}{1+25x^2}$.

1. Calcular el polinomio interpolador p(x) de la función $f(x)$ en 3 nodos equidistantes en el intervalo [-1, 1] (incluyendo los extremos), esto es en -1:1:1. Dibujar las gráficas del polinomio p(x) (rojo) y de f(x) (verde) en el intervalo [-1,1] y los valores interpolados (puntos verdes), tomando como vector auxiliar para crear las gráficas xx= [-1:0.01:1] (subplot(2,1,1)). Dibujar la gráfica del error $E(x) = |f(x) - p(x)|$ en xx, superponiendo los puntos interpolados (subplot(2,1,2)). Calcular el valor máximo del error en xx.

2. Crear la función Matlab (function c=coef_interp(xi,yi)) que implementa el cálculo de los coeficientes del polinomio de interpolación en los datos xi, yi. Argumentos entrada: vectores xi, yi (asumimos vectores columna) y argumento salida: c, vector de coeficientes del polinomio en la base {1,x,..,x^n}.

Si l es la longitud de xi ¿qué grado tendrá el polinomio y cuál será la dimensión de la matriz H de coeficientes del sistema?
¿Qué forma tendría la matriz H si trabajásemos con la expresión del polinomio en la base {1, (x-1),,…,(x-1)^n}?

3. El objetivo ahora es sistematizar lo hecho en el Apartado 1, generando polinomios que intepolen f(x) en un nº cada vez mayor 3, 5, 9 y 17 nodos equidistantes en el intervalo [-1,1] (incluyendo los extremos) y estudiar el error de interpolación. Para ello, hacer un bucle de n=1 a n=4 (3(n=1), 5 (n=2), 9 (n=3) y 17 (n=4) nodos equidistantes en [-1,1]), de forma que para cada n:

- xi: Nodos donde se interpola: **h=1/2^(n-1)** (distancia entre nodos consecutivos), xi=[-1:h:1]'
- yi: Valores de la función f(x) en xi.
- Aplicar la función creada en Apartado 2 para resolver el sistema lineal resultante.
- Para pintar las gráficas de los polinomios resultantes: Para cada n hay que evaluar el polinomio en los puntos xx. Previamente podemos crear una matriz pp de ceros de dimensión 4xlenght(xx), de forma que en la fila n-ésima pp(n,:) guardaremos los valores del correspondiente polinomio en xx. Nota: Para evaluar el polinomio en xx utilizar comando polyval o algoritmo recursivo tipo for k=1:l, pp(n,:)=pp(n,:)+c(k)*xx.^(k-1);end o método de Horner (ver anexo1).
- Para pintar las gráficas del error de interpolación en cada caso: Previamente creamos una matriz (que llamaremos error) de ceros de dimensión 4xlenght(xx), de forma que en la fila n-ésima error(n,:) guardaremos los valores del correspondiente error de interpolación en xx.
- Para cada n pintar las gráficas de la función f(x) y los polinomios interpolantes (subplot(2,4,n)) y las de los correspondientes errores de interpolación ((subplot(2,4,n+4)).
- Calcular para cada n el valor máximo del error (max(err(n,:))) y guardar en el vector maxerr. Utilizar el comando fprintf para que en cada línea muestre el nº de nodos empleados (1+2^n %2d) y el máximo del error (%0.2e). Comentar los resultados.

1.

```
% Datos:
xi=[-1:1:1]';
yi=1./(1+25*xi.^2);
% Matriz y solución del sistema:
H=[xi.^0 xi.^1 xi.^2];
c= H\yi;;
%Gráficas y errores
xx=-1:0.01:1;
pp=polyval(c(end:-1:1), xx); (o bien pp=c(1)+c(2)*xx+c(3)*xx.^2; )
% Error en los puntos xx
error=abs(pp-(1./(1+25*xx.^2)));
subplot(1 2 1) plot(xx,pp, xi,yi,'*r')
subplot(1 2 2) plot(xx,error, xi,yi,'*r')
```

2.

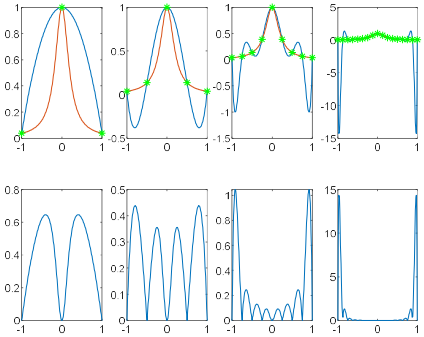
```
function c = coef_interp(xi,yi)
%Calcula los coeficientes del polinomio interpolación (en base 1,x,...,x^n-1)
%en los puntos (xi,yi), resolviendo el correspondiente sistema lineal
% Sistema lineal y resolución.
l=length(xi);
H=zeros(l,l);
for k=1:l
H(:,k)=xi.^(k-1);
end
c=H\yi;
end
```

- Si l es la longitud de xi: el polinomio tendrá grado l-1y la matriz H de coeficientes del sistema será lxl.
- En caso de usarse la base del tipo {1, (x-1),,…,(x-1)^n}? para expresar el polinomio, la matriz H sería H=[(xi-1).^0, (xi-1),,…,(xi-1).^n].

3.

```
clear
a=-1;b=1;
error_max=zeros(1,8);
for n=1:4
   h=1/2^(n-1);
   xi=[a:h:b]';
   yi=1./(1+25*xi.^2);
   %llamamos a la función que hemos creado para resolver el sitema lineal
   c= coef_interp(xi,yi);
% Gráfica de los polinomios y errores de interpolación
xx=a:0.01:b;
```

```
l=length(xx);
pp=zeros(4, l);
pp(n,:)=polyval(c(end:-1:1), xx);
error=zeros(4,l);
error(n,:)=abs(pp(n,:)-(1./(1+25*xx.^2)));
subplot(2,4,n)
plot(xx,pp(n,:),xx, 1./(1+25*xx.^2), xi,yi,'*g')
subplot(2,4,n+4)
plot(xx,error(n,:), xi,yi,'*g')
%Calculamos el máximo de la función error en los puntos xx
% y los guardamos en el vector eror_max:
maxerror=max(error(n,:));
fprintf('nº nodos: %d, maxerror: %0.2e \n',1+2^n, maxerror)
end
```



```
nº nodos: 3, maxerror: 6.46e-01
nº nodos: 5, maxerror: 4.38e-01
nº nodos: 9, maxerror: 1.05e+00
nº nodos: 17, maxerror: 1.43e+01
```

Ejercicio 6. Interpolación polinomial de Hermite. Sea el problema de hallar el polinomio p(x) de grado mínimo que verifique:

$$p(x_0) = p0, p(x_1) = p1, p'(x_0) = d0, p'(x_1) = d1$$

1. ¿Grado del polinomio p(x)? Plantear el problema de interpolación escribiendo $p(x) = c_0 + c_1 x + c_2 x^2 + \ldots$ y generando el correspondiente sistema lineal Hc=b. Escribir una función que admita como argumentos de entrada los vectores $xi = [x_0, x_1]', pi = [p0, p1]', di = [d0, d1]'$, y como argumento de salida el vector $c = [c_0, c_1, \ldots]'$ solución del sistema lineal**.**
2. Aplicación al cálculo del polinomio de Hermite que interpola a la función f(x)=sin(x) en los puntos $x_0 = 0, x_1 = \pi$:
   - Calcular los coeficientes del polinomio de Hermite de interpolación.
   - Representar gráficamente la función seno y su polinomio de interpolación de Hermite en el intervalo $[0, \pi]$.
   - Evaluar el error E(x)=abs(f(x)-p(x)) sobre los puntos xx=0:0.01:pi ¿Cuál es el valor máximo del error de interpolación en esos puntos?
   - Una cota del error de interpolación en el intervalo $[0, \pi]$ viene dada por q(x):

$$|E(x)| \le q(x) = \frac{1}{4!} x^2 (x-\pi)^2 = \quad \text{si } x \in [0, \pi].$$

Dibujar en una gráfica el error E(x)=abs(f(x)-p(x)) y la cota q(x) en el intervalo $[0, \pi]$ utilizando como base los puntos xx ¿Se confirma en este caso que q(x) es una cota del error?

ANEXO

I. Evaluación (recursiva) de un polinomio en un punto o en un vector de puntos x:

Sea p(x) un polinomio de grado n: $p(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \ldots + c_n x^n$ (1)

En MATLAB manejaremos este polinomio como un vector c de tamaño (n+1) conteniendo sus coeficientes:

$$c = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 & \ldots & c_n \end{bmatrix}$$

Dado un punto x (o vector de puntos) tenemos varias formas de evaluar p(x):

OPCION A) Usar de forma directa la expresión (1) del polinomio. Para ello haríamos:

1) p=c(1)
2) desde k=1 hasta k=n p = p + C(k+1)*x.^k

Observación: Recordad que los índices en MATLAB empiezan en 1 por lo que: $c_0 = c(1) \quad c_1 = c(2), \ldots \quad c_n = c(n+1)$

OPCION B) Utilizando la función de MATLAB polyval(c,x) que evalúa un polinomio cuyos coeficientes están en el vector c, en un punto (o vector de puntos) x.

Nota importante: La función polyval espera los coeficientes en orden contrario al aquí presentado, comenzando con el asociado a la potencia más alta (cn) y bajando hasta c0. Una forma sencilla de resolverlo es invertir el vector de coeficientes c que obtenemos con nuestros cálculos antes de llamar a polyval:

```
c_inv=c(end:-1:1);
p=polyval(c_inv,x);
```

OPCION C) Regla de Horner (Procedimiento recursivo, computacionalmente más eficiente).

$$p(x) = (..((c_n x + c_{n-1})x + c_{n-2})x + \ldots + c_1)x + c_0$$

Se empieza con el coeficiente asociado al grado más alto y a partir de ahí aplicamos la regla recursiva:

1) p=C(n+1)
2) for k=n:-1:1,
   p = p .* x + C(k);
   end

II. Diferencias divididas (nodos distintos). Formula Newton

```
function dd=difdiv(xi,fi)
%  Función que  calcula la tabla de diferencias divididas de una función f en puntos xi con f(x)=fi.
%  Argumentos Entrada:   xi nodos,  fi valores de f(x) en xi
% Argumento Salida: dd vector de coeficientes en la Fórmula de Newton del polinomio de interpolación de f(x) en xi.

% En la matriz D se van guardando los valores de la tabla de diferencias divididas.
% Los coeficientes de la F. de Newton se disponen en la primera fila de D, atendiendo a cómo se guardan las dif.
%divididas en el procedimiento siguiente.
% Alternativamente se podría haber organizado de forma que los coeficientes de Newton estuvieran en la diagonal (ver
%en diapositivas de clase).
N=length(xi);
 D=zeros(N);
 D(:,1)=fi;
 for k=2:N
    for j=1:N-k+1
```

```
        dif=D(j,k-1)-D(j+1,k-1);
        dx=xi(j)-xi(j+k-1);
        D(j,k)=dif/dx;
    end
 end
%  Vector con coeficientes F. Newton
 dd=D(1,:);
 end

function px=polNewton(xi,fi,xx)
% Calcula el polinomio de Newton que interpola la tabla (xi,fi) y lo evalúa en el vector xx
% Argumentos de Entrada:   xi nodos donde se interpola,    fi valores en los nodos, xx vector donde se evalúa el valor
%del polinomio de interpolación
% Salida: px=valor(es) del polinomio de Newton en xx
 N=length(xi);m=length(xx);
 %Llamamos a función difdiv
 dd=difdiv(xi,fi);
 pp=zeros(1,m);
 pp=xx-xi(1);
 px=zeros(1,m);
 px=dd(1)+dd(2)*pp;
 for k=3:N
    pp=pp.*(xx-xi(k-1));
    px=px+dd(k)*pp;
 end
end
```