Universidad de Puerto Rico
Departamento de Matemáticas
Humacao, Puerto Rico 00791

MATE 4061 Análisis Numérico Prof. Pablo Negrón

Laboratorio I: Polinomios y Gráficas

Un polinomio como $p(x) = x^4+2x^3-3x^2+4x+5$ se especifica en MATLAB por su vector de coeficientes:

```
>> p=[1 2 -3 4 5]
```

Note que los coeficientes del polinomio se especifican a partir del coeficiente del termino de grado mayor hasta el del termino de grado menor. Los coeficientes se pueden separar con comas también. Las *raíces* de p, i.e., las soluciones de $p(x) = 0$, se pueden calcular mediante:

```
>> roots(p)
```

Esta instrucción calcula tanto raíces reales como complejas. Para aumentar el numero de dígitos en la salida se usa:

```
>> format long
```

NOTA: MATLAB lleva a cabo todos los cómputos numéricos usando presición doble. En el procesador Pentium esto es aproximadamente 16 cifras decimales. La instrucción `format` no afecta el numero de cifras en los cómputos, solo los que se muestran en la pantalla.

Para dibujar el polinomio p procedemos como sigue:

```
>> x=-4:.05:2;
>> y=polyval(p,x);
>> plot(x,y)
```

La primera instrucción crea un vector `x` con entradas $-4, -3.95, \ldots, 1.95, 2$, i.e., los números entre -4 y 2 en intervalos de 0.05. Recuerde que `;` suprime la salida en la pantalla. La segunda instrucción usa la función `polyval` de MATLAB para evaluar el polinomio p en las entradas del vector `x`, i.e., `y` es un vector del mismo tamaño que `x` pero con los valores $p(-4), p(-3.95), \ldots, p(1.95), p(2)$. Finalmente la tercera linea traza la gráfica de `y` versus `x`. Si añadimos las instrucciones

```
>> hold on
>> plot([-4,2],[0,0])
```

tenemos un trazado del eje de x. (Ver Figura (1)). De aquí podemos ver claramente que p tiene dos raíces las cuales pueden compararse con las producidas por la instrucción `roots`.

La instrucción `plot` ajusta automáticamente las escalas de los ejes para los datos a trazarse. En ocasiones queremos enfocar en ciertas regiones de la gráfica y los ejes que `plot` selecciona pueden no ser lo mejores. Esto se puede controlar usando la instrucción `axis`. Veamos un ejemplo. Suponga que trazamos la función $y = \sin(x)$ para $0 \leq x \leq 2\pi$. Esto lo hacemos con las instrucciones:

```
>> x=0:0.05:2*pi;
>> y=sin(x);
>> plot(x,y)
```

Si queremos ver solamente la región dada por $2 \leq x \leq 4$, $-05 \leq y \leq 0.5$, entonces con la instrucción

```
>> axis([2 4 -0.5 0.5])
```

logramos esto. (Otra forma de hacer acercamientos en una gráfica es mediante el comando `zoom`).

Suponga ahora que queremos trazar la función $y = x\sin(x)$. Esto lo podríamos hacer como sigue:

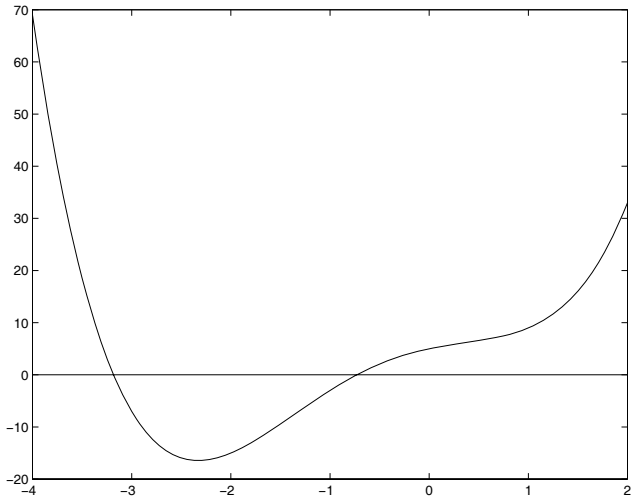

Figure 1: Gráfica del polinomio $p(x) = x^4 + 2x^3 - 3x^2 + 4x + 5$ en el intervalo $[-4, 2]$.

```
>> x=0:0.05:2*pi;
>> y=x.*sin(x);
>> plot(x,y)
```

Note el uso de la operación `.*` la cual multiplica los vectores `x`, `sin(x)` componente a componente. Si intentase usar `x*sin(x)`, MATLAB anuncia un error ya que trata de multiplicar `x`, `sin(x)` como matrices y las dimensiones de estas no son conformes para la multiplicación. (Trate las instrucciones `size(x)` y `size(sin(x))`). Otro ejemplo del uso de operaciones componente a componente es al trazar la gráfica de la función $y = 1/x$. Podemos hacer esto mediante:

```
>> x=0.01:0.01:1;
>> y=1./x;
>> plot(x,y)
```

Note el uso de `./` para realizar la división componente a componente.

Podemos también trazar gráficas paramétricas en MATLAB. Por ejemplo si queremos trazar $(\cos(t), \sin(t))$, $0 \leq t \leq 2\pi$ procedemos como sigue:

```
>> t=0:0.05:2*pi;
>> x=cos(t);
>> y=sin(t);
>> plot(x,y)
```

El resultado debe ser un circulo pero la selección de escalas de la instrucción `plot` en este caso no muestra esto. Para corregir esto podemos entrar (ver Figura (2)):

```
>> axis('equal')
```

Una de las formas más comunes de aproximar a una función f es mediante sus polinomios de Taylor. Por ejemplo, el polinomio de Taylor de grado $2n-1$ alrededor de $a = 0$ para la función $f(x) = \sin(x)$ esta dado por

$$\sum_{k=1}^{n} (-1)^{k-1} \frac{x^{2k-1}}{(2k-1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots + (-1)^{n-1} \frac{x^{2n-1}}{(2n-1)!}.$$

Con el siguiente programa calculamos los primeros cuatro de estos polinomios y los trazamos junto con la función original en el mismo sistema de coordenadas en el intervalo $[-\pi, \pi]$:

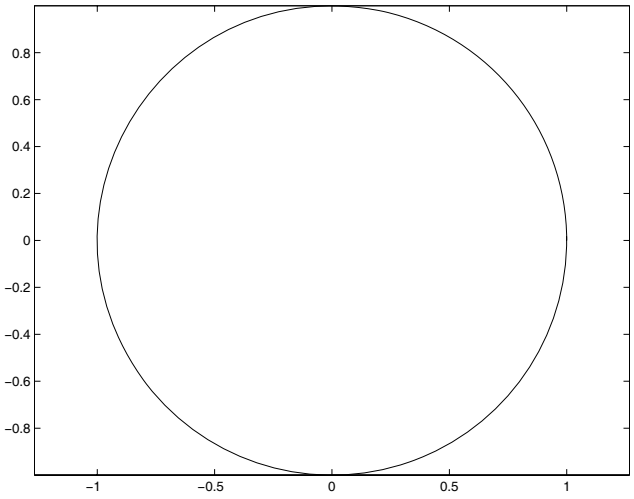

Figure 2: Gráfica del circulo unitario usando la representación paramétrica.

```
x=-pi:.01:pi;
p1=x;
p2=p1-x.^3/6;
p3=p2+x.^5/120;
p4=p3-x.^7/gamma(8);
y=sin(x);
plot(x,y,x,p1,x,p2,x,p3,x,p4)
legend('sin(x)','p1','p2','p3','p4')
```

(Vea Figura (3)). Observe la forma de calcular los polinomios lo cual reduce el numero de operaciones aritméticas. Esto es, cualquier polinomio de Taylor se obtiene añadiendo una correción al polinomio anterior. Note también el uso de la instrucción `legend` para generar una leyenda de la gráfica, y la instrucción `gamma` para calcular el factorial de un numero.

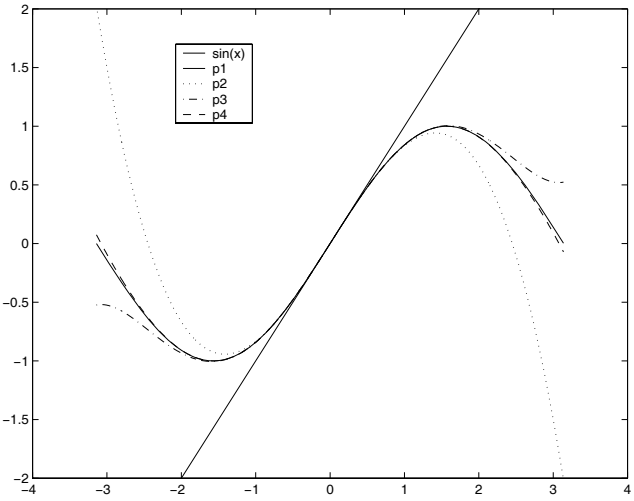

Figure 3: Gráficas de $y = \sin(x)$ y sus polinomios de Taylor alrededor de $a = 0$ de grados $1, 3, 5, 7$.

Otra forma común de aproximar funciones es en el sentido de los *cuadrados mínimos*. La función `ployfit` de MATLAB se utiliza para calcular polinomios de cuadrados mínimos. Este polinomio minimiza en cierto sentido la diferencia entre la función a aproximar y el polinomio. La aproximación de cuadrados mínimos no requiere de derivadas de la función a aproximar (como es el caso para los polinomios de Taylor). Se requiere de un grupo de datos (x_i, y_i), $1 \leq i \leq m$ donde $y_i = f(x_i)$ para toda i. Si `x` representa el vector de las x_i's y `y` el de los y_i's, entonces la instrucción en MATLAB:

```
>> polyfit(x,y,n)
```

produce los coeficientes del polinomio de grado n que mejor aproxima a los datos en el sentido de los cuadrados mínimos. Estos coeficientes se pueden utilizar con la función `polyval` para evaluar dicho polinomio. Veamos un ejemplo:

```
x=-pi:0.5:pi;
y=sin(x);
c=polyfit(x,y,4);
x1=-pi:0.1:pi;
y1=sin(x1);
p4=polyval(c,x1);
plot(x1,y1,x1,p4,'x')
legend('sin(x)','Polinomio CM de grado 4')
```

Las primeras dos instrucciones producen una serie de datos de la función $\sin(x)$ en intervalos de 0.5. La tercera instrucción calcula los coeficientes del polinomio de cuadrados mínimos de grado cuatro para estos datos. El resto del programa evalúa en mas puntos el polinomio de cuadrados mínimos y la función $\sin(x)$ y luego los gráfica. Ver Figura (4).

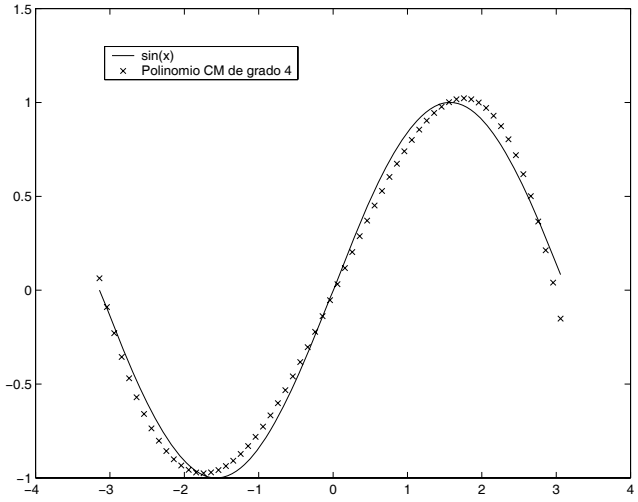

Figure 4: Gráficas de $y = \sin(x)$ y un polinomio de cuadrados mínimos de grado cuatro.

Considere la siguiente ecuación:

$$2(x^2 - 200)\cos^{-1}\left(\frac{x}{20}\right) - x\sqrt{400 - x^2} + 100\pi = 0.$$

Esta ecuación no se puede resolver en forma analítica. Podemos usar la función `fzero` de MATLAB para aproximar una raíz. Esta función necesita el nombre de un programa en MATLAB que evalúe la función del lado izquierdo de la ecuación, y una aproximación inicial para la raíz. La siguiente función en MATLAB evalúa el lado izquierdo de la ecuación para cualquier valor de x:

```
function y=goatfn(x)
   y=2.*(x.^2-200).*acos(x/20)-x.*sqrt(400-x.^2)+100*pi;
```

La función esta escrita de modo que puede recibir un vector como entrada. Esta función se debe guardar en un archivo con nombre `goatfn.m` en su directorio de trabajo. Note que el dominio de esta función es $[-20, 20]$. Podemos ahora trazar la función mediante las siguientes instrucciones:

```
>> x=-20:.01:20;
>> y=goatfn(x);
>> plot(x,y)
```

(Ver Figura (5)). Usando esta gráfica podemos obtener una aproximación `x0` de una raíz de la función y entonces calcular una mas correcta mediante:

```
>> fzero('goatfn',x0)
```

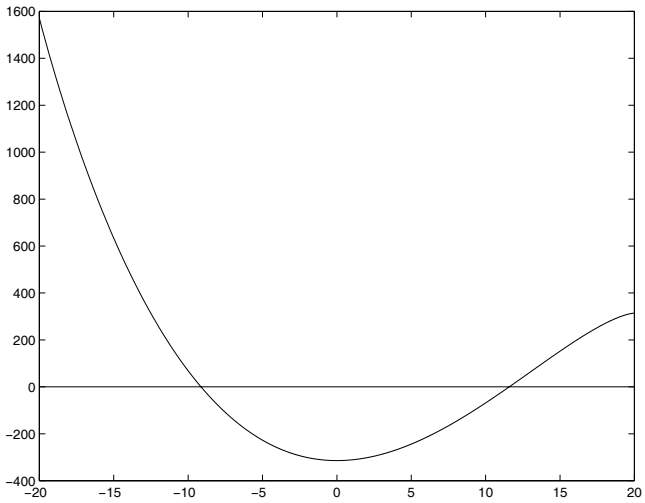

Figure 5: Gráfica de la función dada por la subrutina `goatfn`.