

## Laboratorio 3 MA-33A 2007-1:

### Sistemas de Ecuaciones No-Lineales y Optimización

Gonzalo Hernández - Gonzalo Rios

UChile - Departamento de Ingeniería Matemática

El objetivo de este laboratorio es aprender a utilizar las funciones que están disponibles en **Matlab** para resolver:

- 1) Ecuaciones y sistemas de ecuaciones no-lineales
- 2) Problemas de optimización en una variable y en varias variables.

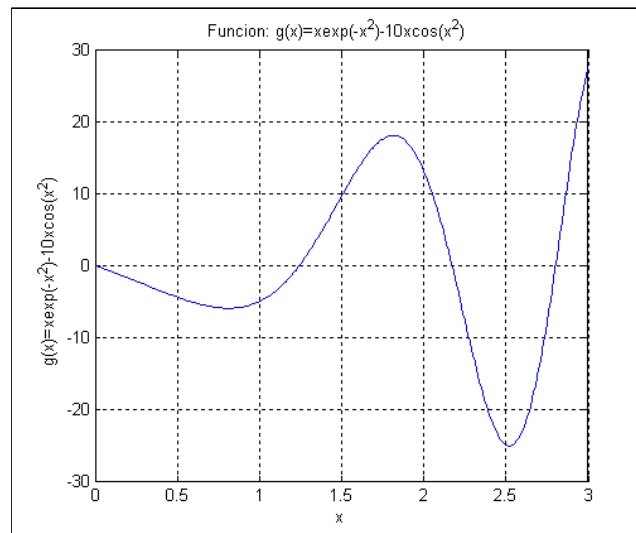
Como este es un tema extenso, nos dedicaremos a encontrar ceros de funciones a variable real y a resolver sistemas de ecuaciones no-lineales, aplicando un método de optimización.

## 1 Ceros de funciones reales: Ecuaciones no-lineales (45 min)

En esta sección aprenderemos a encontrar ceros de funciones, i.e., a resolver ecuaciones no-lineales. Para ejemplificar los cálculos, trabajaremos con la siguiente función:

$$g(x) = xe^{-x^2} - 10x \cos(x^2)$$

Su gráfico esta dado por:



Observamos que  $g(x)$  tiene 3 ceros en el intervalo  $[0, 3]$ .

Primero, necesitaremos programar  $g(x)$  como función **.m**:

**ACT1** Programe una función llamada `g.m`, que reciba como argumento un vector  $x$  de cualquier dimensión y entregue como output la función  $g(x)$  evaluada en el vector. Escriba el código en el informe.

El comando de **Matlab** que permite encontrar ceros de funciones a variable real es:

$$[x, fx] = fzero('funcion', x0, options)$$

**Obs:** El algoritmo implementado por el comando `fzero` es el de bisección con interpolación: El método encuentra un intervalo donde la función cambia de signo. Luego refina el intervalo, aproximando la función por interpolación.

Las salidas de `fzero` son:

1º)  $x$ : El cero encontrado

2º)  $fx$ : El valor de la función en el cero  $fx = f(x)$

Los argumentos de `fzero` son:

1) '`funcion`' es la función que implementa la ecuación en una variable que se desea resolver. Hay 3 formas de ingresar una función a `fzero`. Tomemos como ejemplo la función:

$$f(x) = xe^{-x} - 0.2$$

(a) Como string:

$$[x, fx] = fzero('x * \exp(-x) - 0.2', x0)$$

(b) Como función anónima:

$$\begin{aligned} f &= @(x)(x * \exp(-x) - 0.2); \\ [x, fx] &= fzero(f, x0) \end{aligned}$$

(c) Como archivo `.m`. Si está implementada como archivo `.m` la función  $f(x) = xe^{-x} - 0.2$ :

```
function y=f(x)
y=x.*exp(-x)-0.2;
end
```

En este caso, el comando `fzero` se utiliza de la siguiente forma:

$$[x, fx] = fzero(@f, x0)$$

La función `.m` se pasa por referencia: `@f`, y no por valor.

2)  $x0$  es el punto inicial que necesita el método numérico que está implementado en el comando `fzero`. Como valor inicial  $x0$  podemos ingresar:

(a) Un valor real. Por ejemplo  $x0 = 1$ :

$$[x, fx] = fzero('x * \exp(-x) - 0.2', 1)$$

(b) Un intervalo:

$$[x, fx] = fzero('x * \exp(-x) - 0.2', [a \ b])$$

Esta opción es de gran utilidad en el caso que no conozcamos un "buen" punto inicial . Está la restricción que:

$$f(a)f(b) \leq 0$$

es decir,  $f(a)$  debe tener signo distinto a  $f(b)$ .

Vamos algunos ejemplos:

```
>> [x,fx]=fzero('x*exp(-x)-0.2',1)
```

```
x =
```

```
0.2592
```

```
fx =
```

```
0
```

```
>> [x,fx]=fzero('x*exp(-x)-0.2',2)
```

```
x =
```

```
2.5426
```

```
fx =
```

```
0
```

```
>> [x,fx]=fzero('x*exp(-x)-0.2',[2 4])
```

```
x =
```

```
2.5426
```

```
fx =
```

```
2.7756e-017
```

```
>> [x,fx]=fzero('x*exp(-x)-0.2',[1 2])
```

```
??? Error using ==> fzero
```

```
The function values at the interval endpoints must differ in sign.
```

- 3) *options* son las posibles opciones que están disponibles en el comando *fzero*. Estas opciones si asignan de la siguiente forma:

$$options = optimset('parametro1', valor1, 'parametro2', valor2, \dots)$$

Las opciones que más se utilizan y sus valores posibles son:

Opción	Valor	Descripción
Display	'off'	Nivel de output
	'iter'	Sin output
	'final' (default)	Output de cada iteración
	'notify'	Output de la iteración final
MaxFunEvals	Entero positivo	Output si no hay convergencia
MaxIter	Entero positivo	Máximo número permitido de evaluaciones de la función
TolFun	Real positivo	Máximo número permitido de iteraciones
TolX	Real positivo	Tolerancia de término de la función: Si $ f(x)  \leq TolFun \Rightarrow x$ es cero de $f(x)$
		Tolerancia de término del cero: Si $ x_{k+1} - x_k  \leq TolX \Rightarrow x \doteq x_{k+1}$ es cero de $f(x)$

Por ejemplo, para asignar el valor 'iter' a la opción 'Display' y dar un valor a 'TolFun':

$$options = optimset('Display', 'iter', 'TolFun', 1e-12)$$

Para usar estas opciones llamamos a *fzero* de la siguiente forma:

$$[x, fx] = fzero('x * \exp(-x) - 0.2', 0.5, options)$$

```
>> options=optimset('Display','iter','TolFun',1e-12)
```

```
options =
```

```

    Display: 'iter'
  MaxFunEvals: []
    MaxIter: []
    TolFun: 1.0000e-012
    TolX: []
  FunValCheck: []
```

```
>> [x,fx]=fzero('x*exp(-x)-0.2',0.5,options)
```

Search for an interval around 0.5 containing a sign change:

Func-count	a	f(a)	b	f(b)	Procedure
1	0.5	0.103265	0.5	0.103265	initial interval
3	0.485858	0.0988848	0.514142	0.107464	search
5	0.48	0.097016	0.52	0.109151	search
7	0.471716	0.094318	0.528284	0.111485	search
9	0.46	0.0903905	0.54	0.114684	search
11	0.443431	0.0846077	0.556569	0.11901	search
13	0.42	0.0759597	0.58	0.124741	search
15	0.386863	0.0627512	0.613137	0.132105	search

17	0.34	0.0420019	0.66	0.141122	search
19	0.273726	0.0081796	0.726274	0.151305	search
20	0.18	-0.0496514	0.726274	0.151305	search

Search for a zero in the interval [0.18, 0.72627]:

Func-count	x	f(x)	Procedure
20	0.18	-0.0496514	initial
21	0.314971	0.0298691	interpolation
22	0.264274	0.00289981	interpolation
23	0.259124	-2.71326e-005	interpolation
24	0.259171	1.63199e-007	interpolation
25	0.259171	9.09947e-012	interpolation
26	0.259171	0	interpolation

Zero found in the interval [0.18, 0.726274]

x =

0.2592

fx =

0

**ACT2:** Determine los 3 ceros de la función  $g(x) = xe^{-x^2} - 10x \cos(x^2)$ , de la siguiente forma:

- Parta de 2 puntos iniciales diferentes para cada cero. Asigne algún valor a las opciones de: 'Display', 'TolFun', 'MaxIter'. Anote en el informe como utilizó el comando y el resultado obtenido.
- Parta de 2 intervalos iniciales diferentes para cada cero. Asigne algún valor a las opciones de: 'Display', 'TolFun', 'MaxIter'. Anote en el informe como utilizó el comando y el resultado obtenido.

## 2 Solución de sistemas no-lineales de ecuaciones mediante minimización (45 minutos)

En esta sección aprenderemos a resolver sistemas de ecuaciones no-lineales con los comandos que están disponibles en Matlab. Utilizaremos como ejemplo el sistema cúbico:

$$\begin{aligned} x_1^2 - 10x_1 + x_2^2 + 8 &= 0 \\ x_1x_2^2 + x_1 - 10x_2 + 8 &= 0 \end{aligned}$$

Llamaremos:

$$\begin{aligned} f_1(x_1, x_2) &= x_1^2 - 10x_1 + x_2^2 + 8 \\ f_2(x_1, x_2) &= x_1x_2^2 + x_1 - 10x_2 + 8 \end{aligned}$$

Una función vectorial de la forma:

$$F(x_1, x_2) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} x_1^2 - 10x_1 + x_2^2 + 8 \\ x_1x_2^2 + x_1 - 10x_2 + 8 \end{pmatrix}$$

se programa como archivo `.m` componente a componente. Por ejemplo, si queremos programar la función:

$$G(x_1, x_2) = \begin{pmatrix} g_1(x_1, x_2) \\ g_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} \sin(x_1^2 + x_2^2) + 1 \\ \cos(x_1^2 + x_2^2) - 1 \end{pmatrix}$$

```
function y = G(x)
y = zeros(2,1);
y(1) = sin(x(1)^2+x(2)^2) + 1;
y(2) = cos(x(1)^2+x(2)^2) - 1;
end
```

**ACT3:** Programe la función `F.m` que implementa el campo vectorial  $F(x_1, x_2)$ . Evalúe la función en 5 puntos diferentes de  $\mathbb{R}^2$ . Incluya en el informe el código y los resultados obtenidos.

Para graficar `F.m` se tiene que evaluar en una malla rectangular de puntos. Esto se hace de la siguiente forma:

1º) Crear la malla rectangular de  $41 \times 41$  puntos:

```
>> x1=(-2:0.1:2);
>> x2=(-2:0.1:2);
>> [X1,X2]=meshgrid(x1,x2);
```

Ensaye el resultado del comando `meshgrid` en una malla rectangular pequeña, por ejemplo:

```
>> x1=(-2:1:2);
>> x2=(-2:1:2);
>> [X1,X2]=meshgrid(x1,x2);
```

para ver su efecto.

2º) Evaluar las funciones  $f_1$  y  $f_2$  en esta malla:

```
>> f1=X1.^2-10*X1+X2.^2+8;
>> f2=X1.*X2.^2+X1-10*X2+8;
```

3º) Graficar cada función simultáneamente, evaluándola en puntos de la malla:

```
>> mesh(X1,X2,f1);
>> hold on
>> mesh(X1,X2,f2);
```

Rotando el gráfico (botón al lado derecho del botón mano) podremos apreciar la curva de intersección de las superficies  $f_1$  y  $f_2$ . Es posible determinar de este gráfico un punto cercano al cero ?

**ACT4:** Siguiendo los 3 pasos anteriores, grafique la función `F.m` en la región  $R = [-2, 2] \times [-2, 2]$ . Incluya el gráfico obtenido en el informe. Encuentre una buena rotación que permita determinar un punto cercano al cero.

La versión sin el toolbox de Optimización de **Matlab** (que está instalada en el Laboratorio de Cálculo Numérico), no viene con una versión del comando `fzero` en varias variables (`fsolve`). Pero si viene con comandos para encontrar mínimos de funciones de varias variables. Por esta razón, deberemos encontrar un cero utilizando un método para minimización.

En la transparencia 33 del capítulo 4 está el resultado que necesitamos:

**Teorema:** Sea  $F(x_1, x_2) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix}$  un campo vectorial formado por 2 campos escalares derivables 2 veces con continuidad. Entonces:  $\bar{x} = (\bar{x}_1, \bar{x}_2) \in \mathbb{R}^2$  es un cero de  $F(x_1, x_2)$  si y sólo si es solución del problema de optimización sin restricciones:

$$\min_{(x_1, x_2) \in \mathbb{R}^2} (f_1(x_1, x_2))^2 + (f_2(x_1, x_2))^2$$

**ACT5:** A partir de F.m, programe la función h.m que implementa  $(f_1(x_1, x_2))^2 + (f_2(x_1, x_2))^2$ . Escriba el código en el informe.

El comando de **Matlab** que permite encontrar mínimos sin restricciones de funciones de una y varias variables es:

$$[x, fx] = fminsearch('funcion', x0, options)$$

Las salidas de *fminsearch* son:

- 1º)  $x$ : El mínimo local encontrado
- 2º)  $fx$ : El valor de la función en el mínimo

Los argumentos de *fminsearch* son:

- 1) '*funcion*' implementa la función que se quiere minimizar. Hay 3 formas de ingresar una función a *fminsearch*. Tomemos como ejemplo la función:

$$g(x) = xe^{-x^2} - 10x \cos(x^2)$$

- (a) Como string:

$$[x, fx] = fminsearch('xe^{-x^2} - 10x \cos(x^2)', x0)$$

- (b) Como función anónima:

$$\begin{aligned} g &= @(x)(x * \exp(-x.^2) - 10 * x * \cos(x.^2)); \\ [x, fx] &= fminsearch(g, x0) \end{aligned}$$

- (c) Como archivo .m. Por ejemplo, si utilizamos h.m (ACT5):

$$[x, fx] = fminsearch(@h, x0)$$

En este caso  $x0$  tiene que ser un punto de  $\mathbb{R}^2$ .

- 2)  $x0$  es el punto inicial (no un intervalo) que necesita el método numérico que está implementado en el comando *fminsearch*.
- 3) *options* son las posibles opciones que están disponibles en el comando *fminsearch*. Ellas son las mismas que para el comando *fzero*

Por ejemplo, para asignar el valor 'iter' a la opción 'Display' y dar un valor a 'TolX':

$$options = optimset('Display', 'iter', 'TolX', 1e-8)$$

Para usar estas opciones llamamos a *fminsearch* de esta forma:

$$[x, fx] = fminsearch(@g, 1, options)$$

**ACT6:** Determine los mínimos locales de la función  $g(x) = xe^{-x^2} - 10x \cos(x^2)$ . Parta de 2 condiciones iniciales diferentes para cada mínimo. Asigne algún valor a las opciones de: 'Display', 'TolX', 'MaxIter'. Anote en el informe como utilizó el comando y el resultado obtenido.

Como actividad final determinaremos un mínimo local de **h.m** utilizando el comando *fminsearch*. Este mínimo local será el cero de **F.m**.

**ACT7:** Determine un mínimo local de la función **h.m**. Parta de 3 condiciones iniciales diferentes con las mismas opciones asignadas para: 'Display', 'TolX', 'MaxIter'. Anote en el informe como utilizó el comando y el resultado obtenido.