

Laboratorio 4 MA-33A 2007-1:

Integración y Ecuaciones Diferenciales No Lineales.

Gonzalo Hernández - Gonzalo Rios

UCHile - Departamento de Ingeniería Matemática

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

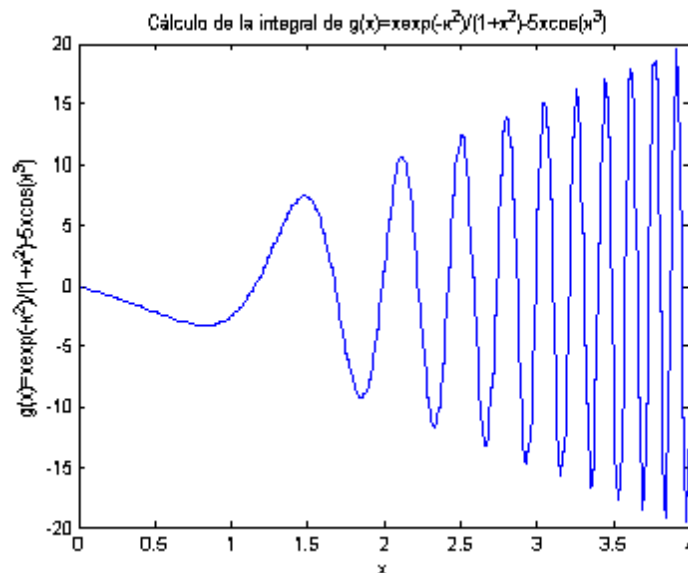
- 1) Calcular integrales en una y varias variables.
- 2) Resolver ecuaciones y sistemas de ecuaciones diferenciales no-lineales

1 Integración simple (25 min)

En esta sección aprenderemos a calcular integrales de funciones. Para ejemplificar los cálculos, trabajaremos con la siguiente función:

$$g(x) = \frac{xe^{-x^2}}{1+x^2} - 5x \cos(x^3)$$

Su gráfico esta dado por:



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 calcula integrales de funciones a variable real es:

$$[q, fcnt] = quad('funcion', a, b, tol)$$

Obs: El algoritmo implementado por el comando *quad* es el de Simpson Recursivo Adaptivo. Veremos este método en clases de cátedra.

Las salidas de *quad* son:

1°) *q*: Valor de la integral:

$$q = \int_a^b f(x)dx$$

2°) *fcnt*: Número de evaluaciones de funciones realizadas por el método.

Los argumentos de *quad* son:

1) '*funcion*' es la función en una variable que se desea integrar. Hay 3 formas de ingresar una función a *quad*. Tomemos como ejemplo la función:

$$f(x) = \cos(x^2)$$

(a) Como string:

$$[q, fcnt] = quad('cos(x.^2)', 0, 1)$$

(b) Como función anónima:

$$\begin{aligned} f &= @(x)(\cos(x.^2)); \\ [q, fcnt] &= quad(f, 0, 1) \end{aligned}$$

(c) Como función .m: $f(x) = \cos(x^2)$:

```
function y=f(x)
y=cos(x.^2);
end
```

En este caso, el comando *quad* se utiliza de la siguiente forma:

$$[q, fcnt] = quad(@f, 0, 1)$$

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

2) Los parámetros *a, b* definen el intervalo de integración $[a, b]$. Por ejemplo, si $[a, b] = [0, 1]$ y la función se pasa por referencia:

$$[q, fcnt] = quad(@f, 0, 1)$$

3) El parámetro *tol* define el máximo error tolerado en valor absoluto. A valores más grandes en *tol* se hacen menos evaluaciones de la función *f* (y por lo tanto es cálculo es más rápido), pero la precisión de la integral es menor. El valor por default es $tol = 1.0e - 6$.

Veamos algunos ejemplos:

```
>> f=@(x)(cos(x.^2));
>> [q,fcnt]=quad(f,0,1)
```

q =

0.9045

fcnt =

17

```
>> f=@(x)(cos(x.^2));
>> [q,fcnt]=quad(f,0,4,1e-8)
```

q =

0.59446032869952

fcnt =

305

```
>> [q,fcnt]=quad(f,0,4,1e-10)
```

q =

0.59446032762362

fcnt =

745

Los mensajes de warning del comando *quad* son:

Mensaje	Descripción
'Minimum step size reached'	Indica que la subdivisión recursiva de la integral produjo un subintervalo de largo menor que el error de redondeo. Si esto ocurre, posiblemente se está integrando una función que tiene alguna singularidad en $[a, b]$.
'Maximum function count exceeded'	Indica que se ha excedido el número máximo de evaluaciones de la función: 10000. Si esto ocurre, posiblemente se está integrando una función que tiene alguna singularidad en $[a, b]$.
'Infinite or Not-a-Number function value encountered'	Indica un overflow de operaciones de punto flotante o división por cero durante el calculo de la integral en $[a, b]$

ACT2: Calcule $q = \int_a^b g(x)dx$ variando el intervalo $[a, b]$ de la siguiente forma:

- Considere $[a, b] = [0, 2]$. Ejecute el comando *quad* con 4 diferentes valores para *tol*. Anote en el informe como utilizó el comando y el detalle del resultado obtenido: Valor de la integral, número de evaluaciones de la función y tiempo de ejecución. Es posible calcular además el error del comando ?
- Considere $[a, b] = [2, 4]$. Ejecute el comando *quad* con 4 diferentes valores para *tol*. Anote en el informe como utilizó el comando y el detalle del resultado obtenido: Valor de la integral, número de evaluaciones de la función y tiempo de ejecución. Es posible calcular además el error del comando ?
- Compare los cálculos que realizó en (a) y (b) con respecto a: Tolerancia, error, número de evaluaciones de la función y tiempo de ejecución. Qué puede concluir ?

2 Integración múltiple (20 min)

En *Matlab* también es posible integrar funciones de 2 o 3 variables. En el caso bidimensional, si se quiere calcular:

$$q = \iint_R f(x, y) dx dy$$

Podemos ejecutar el comando *dblquad* :

$$q = \text{dblquad}('funcion', x_{\min}, x_{\max}, y_{\min}, y_{\max}, tol)$$

Los argumentos de *dblquad* son:

- '*funcion*' es la función que se desea integrar. Hay 2 formas de ingresar una función a *dblquad*. Tomemos como ejemplo la función:

$$f(x, y) = e^{-x^2 y^2}$$

- Como función anónima:

$$\begin{aligned} f &= @(x, y)(\exp(-(x.^2).*(y.^2))); \\ q &= \text{dblquad}(f, 0, 1, 0, 1) \end{aligned}$$

- Como función *.m*:

```
function z=f(x,y)
z=exp(-(x.^2).*(y.^2));
end
```

En este caso, el comando *quad* se utiliza de la siguiente forma. Si $R = [-1, 1] \times [0, 3]$:

$$q = \text{dblquad}(@f, -1, 1, 0, 3)$$

La función *.m* de 2 variables se pasa por referencia: *@f*, y no por valor.

- Los parámetros $R = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ definen la región rectangular de integración. Por ejemplo si: $R = [-\pi/2, \pi/2] \times [1, \sqrt{2}]$:

$$q = \text{dblquad}(@f, -\pi/2, \pi/2, 1, \sqrt{2})$$

- El parámetro *tol* define el máximo error tolerado en valor absoluto. El valor por default es $tol = 1.0e-6$.

ACT3: Calcule $q = \iint_R h(x, y) dx dy$ para una función $h(x, y)$ y una región de integración R de la siguiente forma:

- Defina $h : \mathbb{R}^2 \rightarrow \mathbb{R}$ y R a su gusto. Programe h como función `.m` y ejecute el comando `dblquad` con 3 diferentes valores para tol . Anote en el informe como utilizó el comando y el detalle del resultado obtenido. Es posible calcular además el error del comando ?
- Investigue en el **Help** de **Matlab** como integrar funciones de 2 variables en regiones R no rectangulares (Consulte por `dblquad`). Construya un ejemplo para su función h . Anote en el informe como utilizó el comando en este caso y el detalle del resultado obtenido. Incluya el gráfico de la función que integró.

La integración de funciones sobre \mathbb{R}^3 se realiza de manera similar mediante el comando `triplequad`.

3 Solución de ecuaciones diferenciales (30 min)

En esta sección aprenderemos a resolver ecuaciones diferenciales no-lineales con los comandos que están disponibles en **Matlab**. Primero, resolveremos una ecuación diferencial no lineal de primer orden. Utilizaremos como ejemplo la ecuación de crecimiento logístico:

$$\begin{aligned}\frac{dp}{dt} &= \alpha p(t) - \beta p(t)^2 \\ p(0) &= p_0\end{aligned}$$

que modela el crecimiento de la población de Estados Unidos en el siglo pasado. Los datos reales del problema son:

Año	Tiempo t_k	$p(t)$ real
1900	0.0	76.1
1910	10.0	92.4
1920	20.0	106.5
1930	30.0	123.1
1940	40.0	132.6
1950	50.0	152.3
1960	60.0	180.7
1970	70.0	204.9
1980	80.0	226.5

Los modeladores determinaron que:

$$\begin{aligned}\alpha &= 0.02 \\ \beta &= 0.00004\end{aligned}$$

y de acuerdo a estos datos, la solución exacta de la edo de crecimiento logístico es:

$$p(t) = \frac{500}{\left(1 + \frac{4239}{761} e^{-\frac{1}{50}t}\right)}$$

Existe en **Matlab** una familia de comandos que permite resolver ecuaciones y sistemas de ecuaciones diferenciales. Los de mejor comportamiento son:

`ode45`, `ode23`, `ode15s`, `ode23s`

Los comandos *ode45* y *ode23* implementan los metodos de Runge-Kutta de orden 4 y 2, siendo el método de orden 4 el de mayor precisión. Por otra parte, los comandos *ode15s* y *ode23s* implementan métodos para edo rígidas (stiff) o no suaves de difícil solución numérica en la práctica. La forma de utilizar estos comandos es:

$$[t, y] = \text{solver}('funcion', [t_0 \ T], y_0, options)$$

donde *solver* es alguno de los comandos *ode45*, *ode23*, *ode15s*, *ode23s*. Este comando resuelve numericamente el problema de Cauchy de primer orden:

$$\begin{aligned} \frac{dy}{dt} &= f(t, y) \quad \forall t \in [t_0, T] \\ y(t_0) &= y_0 \end{aligned}$$

Las salidas de *solver* son:

- 1º) *t*: vector de instantes de tiempo, donde $t_0 \leq t(k) \leq T \ \forall k = 1, \dots, n$
- 2º) *y*: Solución de la ecuación diferencial evaluada en los instantes de tiempo *t*

Los argumentos de *solver* son:

- 1) '*funcion*' es la función que define la edo. Debe ser ingresada como función *.m*.
- 2) [*t0 T*] define el intervalo donde se resolverá la edo.
- 3) *y0* es el valor inicial
- 4) *options* son las opciones del comando *solver*. Utilizaremos las opciones por default.

Resolvamos por ejemplo la edo no lineal:

$$\begin{aligned} \frac{dy}{dt} &= \frac{\sin(y^2 + t)}{1 + t^2} \quad \forall t \in [0, 10] \\ y(t_0) &= 0 \end{aligned}$$

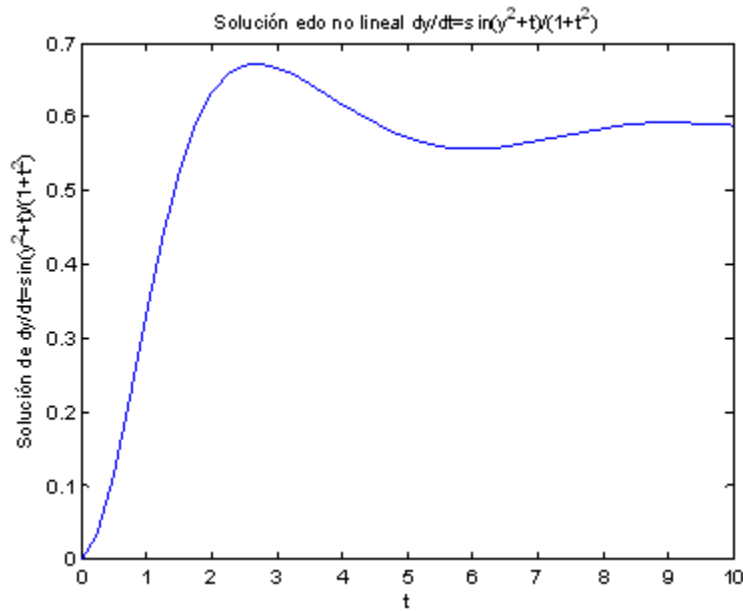
Primero se debe programar $f(t, y) = \frac{\sin(y^2 + t)}{1 + t^2}$ como función *.m*:

```
function dy= f(t,y)
dy = sin(y^2+t)/(1+t^2);
end
```

Ahora se ejecuta algún comando de *solver* pasando la función *f* por referencia. Se recomienda utilizar *ode45* por su precisión y rapidez de cálculo. Si *ode45* demora en entregar la solución, se debe ocupar *ode15s*:

$$[t, y] = \text{ode45}(@f, [0 \ 10], 0);$$

Podemos apreciar la solución graficando:



Volvamos ahora a nuestro ejemplo inicial.

ACT4: Modelo de Crecimiento Logístico:

- (a) Determine la solución numérica de la edo de crecimiento logístico utilizando los comandos *ode23* y *ode45* :

$$\begin{aligned}\frac{dp}{dt} &= 0.02p(t) - 0.00004p(t)^2 \\ p(0) &= 76.1\end{aligned}$$

Compare el tiempo de ejecución y grafique las soluciones entregada por *ode23* y *ode45*. Calcule el error del método utilizando los datos reales. Llamaremos este error E_{RK23} y E_{RK45} . Anote en el informe como utilizó el comando y los resultados.

- (b) Determine el error del modelo comparando los datos reales con la solución exacta de la ecuación:

$$p(t) = \frac{500}{\left(1 + \frac{4239}{761}e^{-\frac{1}{50}t}\right)}$$

Llamaremos este error E_M . Anote en el informe como calculó E_M y el detalle del resultado obtenido.

- (c) Analice el error del método y del modelo. De acuerdo a sus resultados:
- Puede concluir que la solución entregada por el método de Runge - Kutta es precisa ?
 - Puede concluir que el modelo de crecimiento logístico representa bien los datos de la tabla ?
- Justifique claramente sus respuestas en base a los cálculos realizados.

4 Solución de sistemas de ecuaciones diferenciales (15 minutos)

Como última actividad, se resolverá numericamente una edo no lineal rigida de segundo orden mediante el comando *ode15s*. Considere la ecuación del oscilador de Van der Pol que modela un circuito electrónico utilizado en las radios alrededor de 1920:

$$\frac{d^2y}{dt^2} + y(t) + \mu(y^2(t) - 1)\frac{dy}{dt} = 0$$

La función $y(t)$ es la corriente eléctrica y la constante μ mide la no linealidad del sistema. Se utilizará:
 $\mu = 1000$

ACT5: Oscilador de Van der Pol

- (a) Transforme la edo de Van der Pol en un sistema de ecuaciones diferenciales de primer orden aplicando la transformación standard:

$$\begin{aligned}y_1(t) &= y(t) \\ y_2(t) &= \frac{dy}{dt}\end{aligned}$$

Programe el sistema resultante como función `.m`. Llame a esta función `vdp.m`. Escriba el código en el informe.

- (b) Resuelva la edo no lineal de Van der Pol en el intervalo $[0 \ 6000]$ utilizando el comando `ode15s` y las condiciones iniciales $[1; 0]$ y $[2; 0]$. Anote en el informe como utilizó el comando en cada caso y los gráficos de $y_1(t)$ y $y_2(t)$ en el intervalo $[0 \ 6000]$. Podrá apreciar el comportamiento oscilatorio y periódico de la solución.