

Cálculo numérico e simbólico

- Resolución **numérica** de ecuación non linear dunha variábel:
 $f(x)=0$:

$f=@(x)$ expresión; $x = fzero(f, x0)$

Función anónima

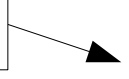
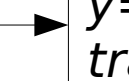
- $x0$ é un punto de inicio
 - Ex: $xe^{-x}=0.2$: $f=@(x) x*\exp(-x)-0.2$; $fzero(f, 0.7)$
- Mínimo dunha función:

$f=@(x)$ expresión
 $xmin=fminbnd(f,a,b)$;
 $[xmin \ vmin] = fminbnd(f, a,b)$

- Busca o valor mínimo de f no intervalo $[a,b]$
 - Retorna punto e valor mínimo $xmin$ e $vmin$
 - Exemplo: $f=@(x) x^3-12*x^2+3*x-1$;
 $[x \ v]=fminbnd(f,0,10)$

Integración numérica

$$x = quad('función', a, b)$$

- A función pode ser unha expresión, función predefinida de Matlab ou función definida por nós. Igual que en *fzero()* e *fminbnd()*
- A función debe escribirse considerando que *x* é un vector (operandos compoñente a compoñente)
- Ex: `quad(@(x) x.*exp(-0.8*x)+0.2, 0, 8)` 
$$\int_0^8 (xe^{-0.8x} + 0.2) dx$$
- Para integral de función definida polos puntos $\{(x_i, y_i), i=1 \dots n\}$ mediante o método dos trapezios: `trapz(x, y)` 

```
x=0:0.01:8;  
y=x.*exp(-0.8*x)+0.2;  
trapz(x, y)
```
- Simbólicamente: `syms x; eval(int(x*exp(-0.8*x)+0.2,x,0,8))`
- En octave: `pkg load symbolic; syms x; f=x*exp(-sym(0.8,'f')*x)+sym(0.2,'f'); eval(int(f,x,0,8))`

Cálculo simbólico (I)

- Definición de variábeis simbólicas: `syms v1 ... vn`
- En **Octave**: antes de definilas, ejecuta `pkg load symbolic`.
- Límites: `limit(expresión, variábel, valor, lado)`
 - `var`, `valor` e `lado` son opcionais (`var=x`, `valor=0` por defecto; `lado = 'left'` ou `'right'`, por defecto calcúlase o límite por ambos lados)
 - Ex: `syms x; limit(1/x); limit(1/x,inf); limit(1/x, x, 0, 'left');`
- Derivación: `diff(expresión, var, orde)`
 - `var` e `orde` son opcionais (`var=x` e `orde=1`)
 - Ex: `syms x; diff(x^2); diff(x^2, x); diff(x^2,x,2)`
`syms x y; diff(x^2+y^2,x,y)`

Cálculo simbólico (II)

- Integración indefinida: *int(expresión, var)*
- Integración definida: *int(expresión, var, ini, fin)*
 - Ex: *int(cos(x), x); int(exp(-x^2), 0, inf); eval(ans)*
- Series numéricas: *symsum(expresión, var, ini, fin)*
 - Ex: *syms n; symsum(1/(n^2-1), n, 2, inf)* $\longrightarrow \sum_{n=2}^{\infty} \frac{1}{n^2-1}$
- Substitución de variábeis simbólicas por valores numéricos: *res = subs(expresión, var, valor(es))*
 - Ex: *subs(x^2+exp(-x), x, pi/2)*
 - En Octave: *subs(sin(x*y), [x,y], [sym(pi)/4, sym(pi)/4])*
- *Avaliación de expresión simbólica en punto flotante: eval(expr) ou double(expr).*

Conversión entre cadeas de caracteres, expresións simbólicas e referencias a función

- Conversión de **cadea de caracteres** (cunha expresión matemática) a **expresión simbólica**:

`syms x;f=str2sym('x^2')`

Non podes chamala (p.ex. $f(5)$), pero si podes derivala: `diff(f,x)`, e calcular o seu valor con `subs(f,x,5)`

Non funciona `diff('x^2')`: `diff(str2sym('x^2'))`

En Octave, `str2sym` non existe, debes facer `sym('x^2')`

- Podes facer operacións de cálculo simbólico:

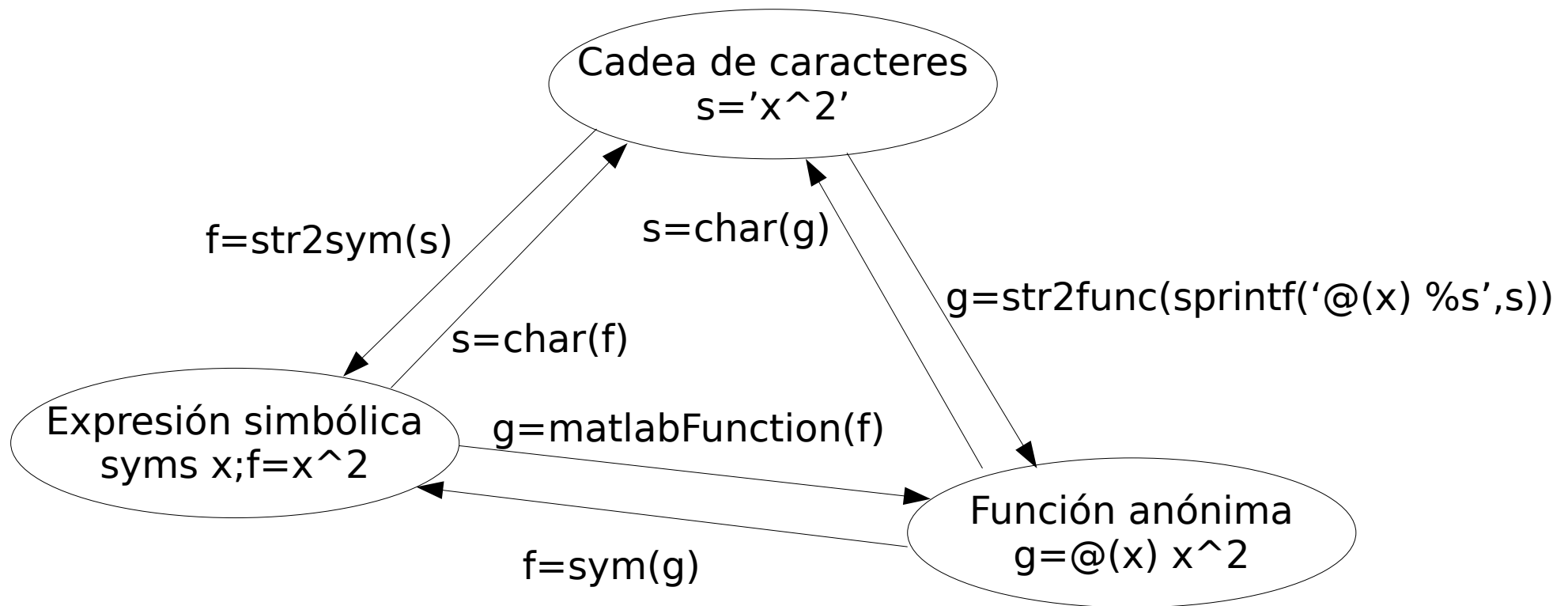
con funcións `inline`: `f=inline('x^2');` `diff(f(x))`

con funcións anónimas: `f=str2func('@(x) x^2');` `diff(f(x))`

- Conversión de **expresión simbólica** a **función anónima**:

`syms x;f=x^2;g=matlabFunction(f)`

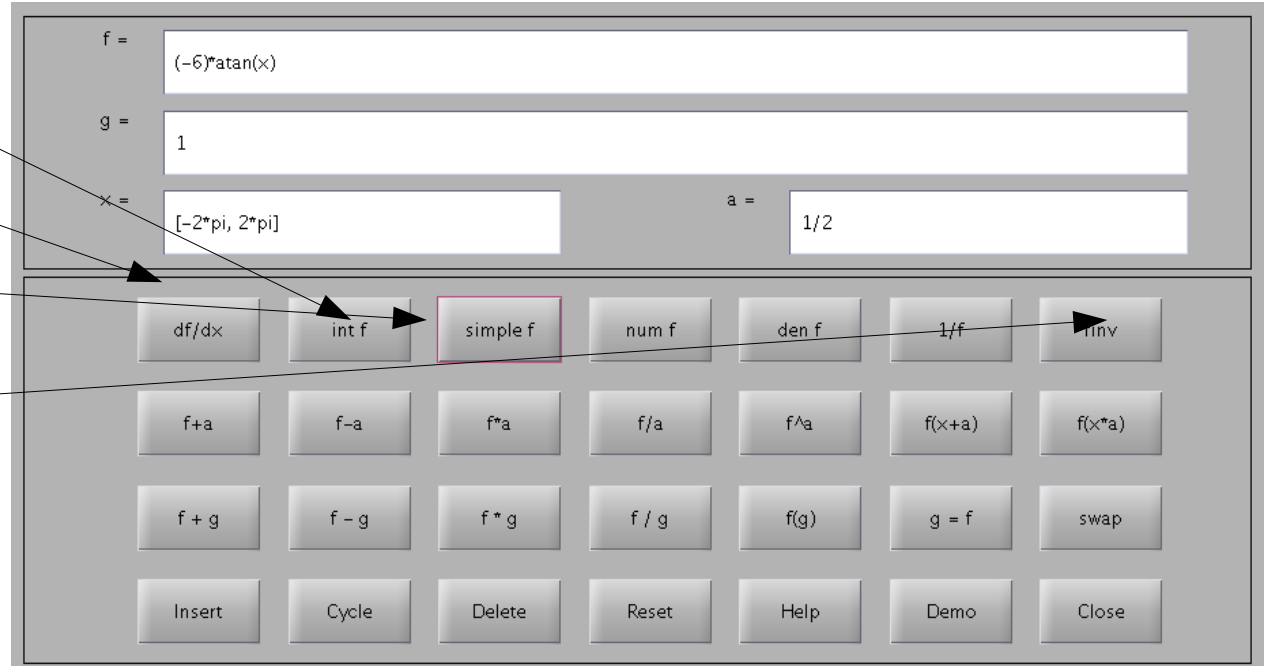
Esquema de conversión: cadea de caracteres ↔ función anónima ↔ expresión simbólica



Cálculo simbólico (III)

- *funtool*: calculadora de funciones

- Integrar
- Derivar
- Simplificar
- Invertir
- *help funtool*



Cálculo simbólico (IV)

Polinomio de Taylor orde $n-1$ dunha función f en $x=a$:

taylor(f,x,'ExpansionPoint',a,'Order',n)

taylor(f): en $x=0$, orde 5

taylor(f,x): en $x=0$, orde 5

taylor(f,x,'ExpansionPoint',1);

en $x=1$, orde 5

taylor(f,x,'Order',7);

en $x=0$, orde 6

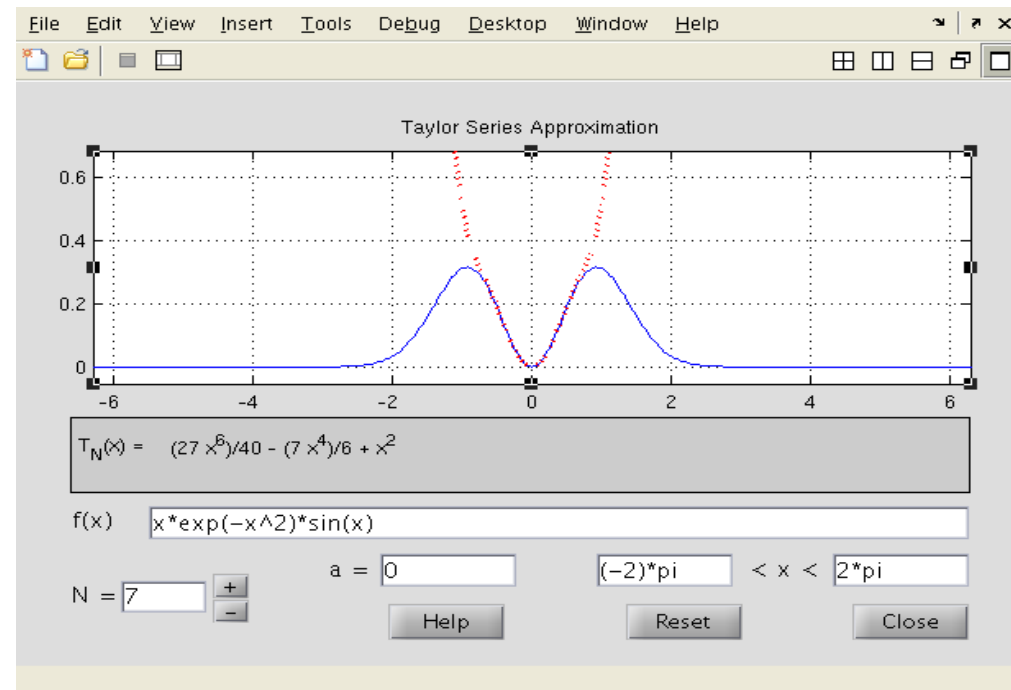
Ex: syms x; f=(x-1)/(x+1);

*taylor(f,x,'ExpansionPoint',1,
'Order',7)*

taylor() tamén en octave

taylortool: calcula e representa graficamente a función e o seu polinomio de Taylor

Programación en Octave



Cálculo numérico e simbólico

Cálculo simbólico (V)

- Resolución **simbólica** de ecuaciones e sistemas de ecuaciones (también en octave):

`syms x; h = solve(expresión, var)`

– A ecuación é da forma *`expresión=0`*

– Ex: *`syms x; h = solve(x*exp(2*x)-5)`*

- Resolución simbólica dun sistema de varias ecuaciones con varias variábeis:

`syms x1...xn; [v1 ... vn]=solve(eq1,...,eqn, x1, ..., xn)`

– Ex: *`syms x,y,z; [xz yz]= solve(10*x+12*y+16*z, 5*x-y-13*z,x,y) %x,y función de z`*

– Ex: *`syms x, y; [x y] = solve(x*exp(y)-3,y*exp(x)-2,x,y)`*

Combinatoria

- Función *nchoosek*(*n*, *k*): calcula $\frac{n!}{(n-k)!k!}$
- **Combinacións** de *n* elementos dun vector *v* tomados en grupos de *k* (*n*<15): *nchoosek*(*v*,*k*), *nchoosek*('abcde',3), *nchoosek*({'a','b','c','d','e'},3)
- Retorna unha matriz de $n!/((n - k)! k!)$ filas e *k* columnas; usa *n* < 15 para evitar explosión combinatoria
- **Permutacións** de *n* elementos: *perms*(1:*n*), *perms*('abcd'), *perms*({'a' 'b' 'c'}). Retorna unha matriz de *n*! filas e *n* columnas; *n* debe ser < 15
- Selección aleatoria dunha permutación de *n* números de 1 a *n*: *randperm*(*n*). Ex: *i=randperm(n);v(randperm(n))*: barallamento aleatorio dos elementos do vector *v* con *n* elementos.

Exercicios

- 1) Representa gráficamente $f(x)=e^{-x^2/10}\sin(x^2)$ e atopa o punto $(x_0, f(x_0))$ que minimiza f en $[-10, 10]$
- 2) Resolve as ecuacións $4\cos 2x - e^{x/2} + 5 = 0$; $2\sin x - \sqrt{x} = -2.5$
 $\cos x = 2x^3$
- 3) Calcula θ tal que $92 = 88 / (\cos \theta + 0.45 \sin \theta)$
- 4) Calcula numericamente a integral $\longrightarrow \int_0^5 \frac{1}{0.6x^2 + 0.5x + 2} dx$
- 5) Calcula simbólicamente:

$$\lim_{x \rightarrow \infty} \frac{\ln x}{\sqrt{x}}$$

$$\lim_{x \rightarrow 1} \left(\frac{x}{x-1} - \frac{1}{\ln x} \right)$$

$$\frac{d}{dx} \left[\frac{\sqrt{x^2 + x + 2}}{x - 1} \right]$$

$$\int_0^{\pi/2} \left(1 + \frac{1}{2} \sin^2 x \right) dx$$

$$\sum_{n=1}^{\infty} \frac{2^n + 3^n}{n^2 + \log n + 5^n}$$

Soluciones aos exercicios (I)

1) Represento e atopo o mínimo en $[-10, 10]$

```
fplot('exp(-x^2/10)*sin(x^2)', [-10,10])
```

```
[xmin fmin] = fminbnd('exp(-x^2/10)*sin(x^2)', -10, 10)
```

```
x = 2.1477, fmin = -0.6274
```

2) a) $fzero('4*cos(2*x)-exp(x/2)+5', 0) \rightarrow 1.2374$

comprobación: $subs('4*cos(2*x)-exp(x/2)+5', 1.2374) \rightarrow 2.4960e-04$

```
syms x; solve(4*cos(2*x)-exp(x/2)+5, x)
```

comprobación: $subs(4*cos(2*x)-exp(x/2)+5, ans)$

b) Solución numérica: $fzero('2*sin(x)-sqrt(x)+2.5', 2) \rightarrow 3.4664$

comprobación: $eval(subs(2*sin(x)-sqrt(x)+2.5, 3.4664))$

c) Solución simbólica: $fzero('cos(x)-2*x^3', 0) \rightarrow 0.7214$

comprobación: $subs(cos(x)-2*x^3, 0.7214)$

Soluciones aos exercicios (II)

3) `fzero('92-88/(cos(x)+ 0.45*sin(x))', 0) -> -0.0881`

4) `quad('1./(0.6*x.^2+0.5*x+2)', 0, 5) -> 0.9596`

`x=0:0.01:5; y=1./(0.6*x.^2+0.5*x+2); trapz(x, y) -> 0.9596`

`syms x; eval(int(1/(0.6*x^2+0.5*x+2), x, 0, 5)) -> 0.9596`

5) `syms x; limit(x/(x-1)-1/log(x), x, 1) -> 1/2`

`syms x; diff(sqrt(x^2 + x + 2)/(x-1), x, 1)`

`syms x: int(1 + sin(x)^2/2, x, 0, pi/2) -> 5/8*pi`

`syms n; symsum((2^n + 3^n)/(n^2+log(x)+5^n), n, 1, inf)`
`-> sum((2^n+3^n)/(n^2+log(x)+5^n), n = 1 .. Inf) (non a`
`resolve, pero converxe)`

Versión vectorizada:

```
n=1:100;  
sum((2.^n + 3.^n)./(n.^2 + log(n)+5.^n))
```

Resultado: 1.8918

```
clear all  
suma = 0;sumando = inf; n = 1  
while sumando > 1e-5  
    sumando = (2^n + 3^n)/(n^2 + log(n) + 5^n);  
    suma = suma + sumando; n = n + 1;  
end  
fprintf('n= %i suma= %g\n', n, suma);
```