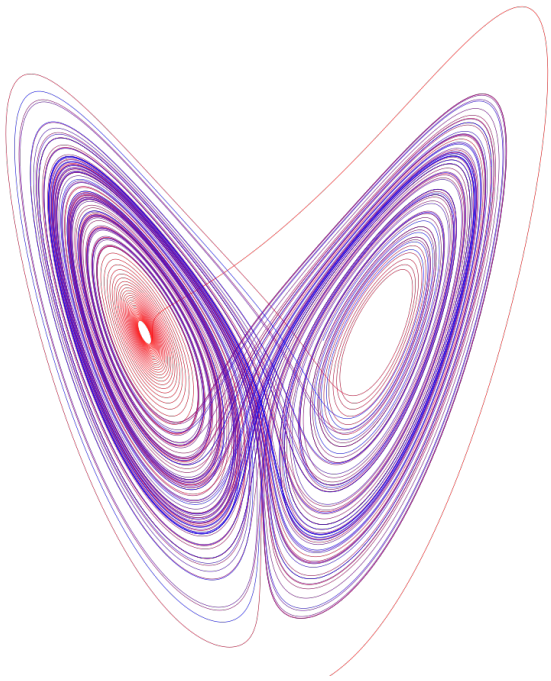
1 Calcolo matematico simbolico

In molte occasioni è necessario effettuare delle operazioni matematiche con espressioni che contengono **variabili simboliche**, ossia variabili senza un valore numerico preassegnato quando l'operazione viene eseguita. Il risultato di queste operazioni è di nuovo un'**espressione simbolica** che contiene variabili simboliche.



Esempi molto tipici e frequenti sono la risoluzione di un'**equazione algebrica** che contiene diverse variabili, la **differenziazione** e l'**integrazione** di espressioni matematiche, la risoluzione di **equazioni differenziali**, ecc. MATLAB ha la capacità di eseguire operazioni simboliche grazie al pacchetto Symbolic Math Toolbox.

1.1 Symbolic Math Toolbox

Il pacchetto Symbolic Math Toolbox è uno dei paccheti fondamentali che estendono le funzionalità di MATLAB. Generalmente, esso è compreso nella versione di MATLAB per studenti. Per verificare se è istallato nella versione in uso, si faccia riferimento al comando `ver`, che mostra informazioni sulla versione corrente di MATLAB, Simulink, e dei toolboxes istallati.

Questo pacchetto consente di definire un nuovo *data type*: un **oggetto simbolico**, tramite i comandi `sym` e `syms` secondo la sintassi

```
syms a b c x  %definiamo le variabili simboliche a b c x
d = sym('d')  %definizione alternativa meno consigliata
g = sym('gamma')
s = sym('xh12')
c = sym(5)
```

Il terzo e il quarto esempio mostrano che il nome dell'oggetto simbolico (in questo caso `g` ed `s`) può essere diverso dall'oggetto simbolico stesso (in questo caso `gamma` e `xh12`). L'ultimo esempio mostra che gli oggetti simbolici possono essere creati anche a partire dai numeri. In questo caso non si mettono gli apici.

In generale un oggetto simbolico può essere una **variabile** (senza un valore numerico preassegnato), un **numero**, oppure una **espressione** fatta di variabili simboliche e numeri. Notare la differenza con il comando `syms` che crea oggetti simbolici dallo stesso nome delle variabili stesse.

Le **espressioni simboliche** sono espressioni matematiche scritte in termini di variabili simboliche, che possono contenere operazioni matematiche eseguibili (addizione, sottrazione, moltiplicazione, e divisione)

```
syms a b c
f = 3*a^3-2*b+4.3*c  %espressione simbolica
```

La parte numerica delle operazioni simboliche viene sempre effettuata in maniera **esatta**, senza approssimare i valori, contrariamente a quello che succede quando si effettuano operazioni con epressioni numeriche.

Per vedere la differenza, si eseguino su MATLAB i comandi

```
as = sym(2); bs = sym(7); cs = bs/as+sqrt(3)  %calcolo simbolico
ae = 2; be = 7; ce = ae/be+sqrt(3)            %calcolo numerico
```

Nel primo caso MATLAB restituisce il valore *esatto* `3(1/2)+7/2`, nel secondo invece il valore numerico *approssimato* `2.0178`. Nel caso in cui le espressioni simboliche includono variabili numeriche, ottenute in precedenza dall'esecuzione di espressioni numeriche, il loro valore esatto verrà usato per l'espressione simbolica, anche quando in precedenza la variabile è stata mostrata con un valore approssimato.

1.2 Risoluzione di equazioni algebriche

Consideriamo un esempio standard e supponiamo di volere risolvere l'equazione di secondo grado $ax^2+bx+c=0$. Come primo passo, dobbiamo dichiarare le variabili simboliche `a,b,c` e `x`. Successivamente, possiamo richiamare il comando `solve` come segue

```
syms a b c x;
y = solve(a*x^2+b*x+c);
y = solve(a*x^2+b*x+c==0); %equivalentemente
```

Nel caso in cui non viene specificato a cosa uguagliare l'espressione, MATLAB assume che l'espressione sia uguagliata a zero. In entrambi i casi, il risultato sarà un oggetto simbolico 2×1 che contiene le espressioni delle due soluzioni.

Una volta visualizzato il risultato, può sorprendere come MATLAB abbia saputo come risolvere assumendo che l'incognita fosse `x`. In effetti, siccome non abbiamo specificato un secondo argomento per la funzione `solve`, che specifichi l'incognita, MATLAB decide seguendo un *ordine lessicografico*, selezionando la variabile più vicina alfabeticamente alla variabile `x`, opppure `X` (dando precedenza alle minuscole), e risolve per quella variabile. Se vogliamo specificare a piacere la variabile si procede secondo la sintassi

```
syms a b c x
y = solve(a*x^2+b*x+c,b);  %risolviamo la stessa equazione
                           %nella variabile b
```

Esempio 1.1. *Si risolvino le seguenti equazioni tramite l'utilizzo del comando* `solve`

1. $cos(\alpha)y + 14d + \pi = 0$
2. $ax^2 + 14d + 2y = 0$
3. $(4 + 3a)z + 6b - k = 0$
4. $g + 2G = 0$

dopo avere appropriatamente introdotto le rispettive variabili simboliche tramite l'utilizzo del comando `syms`*, senza specificare l'incognita. Si commenti sull'icognita in cui MATLAB decide di risolvere in ciascun caso.*

```
syms alpha d y
s1 = solve(cos(alpha)*y + 14*d + pi)  %soluzione: risolve in y

syms a d x y
s2 = solve(a*x^2 + 14*d + 2*y)        %soluzione: risolve in x

syms a b k z
s3= solve((4+3*a)*z + 6*b - k)        %soluzione: risolve in z

syms g G
s4= solve(g + 2*G)                    %soluzione: risolve in g
```

Una volta risolta un'equazione, per verificare che il risultato sia corretto, è sufficiente sostituire il risultato al posto della variabile, facendo attenzione alle operazioni per array. Ad esempio

```
syms a b c x
y = solve(a*x^2+b*x+c);
a*y.^2+b*y+c
```

Può capitare che il risultato non venga mostrato come zero, bensì in una forma **non semplificata**. Di fatto, è tipico di molti pacchetti simbolici richiedere un *postprocessing* dei risultati, in modo da semplifichicare una data espressione simbolica. In generale, per cambiare la forma di un'espressione simbolica esistente si possono utilizzare i comandi

- ⋆ `collect` per collezionare i termini dell'espressione con la stessa potenza, in ordine decrescente della potenza
- ⋆ `expand` per espandere prodotti di sommatorie oppure espressioni trigonometriche o logaritmiche di sommatorie
- ⋆ `factor` per ottenere i fattori primi di una espressione
- ⋆ `simplify` per generare una forma più semplice usando operazioni matematiche e identità funzionali o trigonometriche
- ⋆ `simple` per generare la forma dell'espressione con meno caratteri (spesso, dando lo stesso risultato di `simplify`)

Molte volte, siccome MATLAB assume implicitamente che tutte le variabili siano complesse, non riesce a semplificare le variabili simboliche. Informazioni note a priori, che ci dicono che la variabile sia reale oppure positiva, possono essere fondamentali nel calcolo simbolico. In questo caso, la dichiarazione di una variabile simbolica può essere accompagnata dai comandi `real`, `unreal`, `positive` secondo la sintassi

```
syms x real
x = sym('x','real') %equivalentemente
syms y positive
syms a b unreal
```

Supponiamo di avere una variabile simbolica `y`, che dipende da altri parametri. Per ottenere il valore di `y` per determinati valori numerici dei parametri si utilizza il comando `subs`, come mostrato nel seguente esempio

```
syms a b c x
y = solve(a*x^2+b*x+c);
a = 1; b = 3, c = -5;
subs(y)

%alternativamente:
syms a b c x
y = solve(a*x^2+b*x+c);
subs(y, {a,b,c}, {1,3,-5})
```

In genere, il comando `subs(y)` ritorna la variabile `y` con i parametri sostituiti dai loro valori numerici (se ci sono) come memorizzati nel Workspace.

1.3 Calcolo differenziale ed integrale

Il pacchetto Symbolic Math Toolbox consente di calcolare analiticamente la **derivata**, l'**integrale** e il **polinomio di Taylor** di semplici funzioni tramite i comandi `diff`, `int`, e `taylor`. Ci sono altre funzioni di calcolo fondamentali incluse nel pacchetto, quali

`diff`	differenziazione
`int`	integrazione
`limit`	limite
`taylor`	serie di Taylor
`jacobian`	matrice Jacobiana
`mysum`	serie somma

Per informazioni sulla loro sintassi si faccia riferimento al comando `help` di MATLAB.

Supponiamo che ci venga data l'espressione di una funzione `f` nella variabile x. Anzitutto abbiamo bisogno di manipolare algebricamente la funzione `f` senza doverla necessariamente valutare. Per questo si rincorre al calcolo simbolico e si dichiara la variabile simbolica `x` che compare nella definizione della funzione `f` con il comando `syms x`. Dopodichè, si definisce l'espressione della funzione `f` sulla quale si intende operare, e si utilizzano i seguenti comandi secondo la sintassi

- ⋆ `diff(f)` per calcolare la derivata prima
- ⋆ `diff(f,n)` per calcolare la derivata di ordine n
- ⋆ `int(f)` per calcolare l'integrale in senso indefinito
- ⋆ `taylor(f,'expansionPoint',x0,'Order',n+1)` per cacolare il polinomio di Taylor di grado n in un intorno del punto $x0$.

È molto consigliato l'utilizzo del comando `simplify` da applicare alle espressioni generate da `diff`, `int` e `taylor` per semplificare le loro espressioni, qualora possibile, in modo da renderle più semplici possibili.

Esempio 1.2. *Utilizzando le funzioni del pacchetto* Symbolic Math Toolbox*, si calcoli la derivata, l'integrale indefinito ed il polinomio di Taylor del quarto ordine della funzione*

$$f(x) = \frac{x^2 + 2x + 2}{x^2 + 1} \tag{1}$$

in un intorno del punto $x_0 = 1$.

Soluzione:

```
syms x
f=(x^2+2*x+2)/(x^2+1);
d=diff(f);
intg=int(f);
```

```
t= taylor(f,'expansionPoint',1,'Order',5);
d_simpl=simplify(d)
intg_simpl=simplify(intg)
t_simpl=simplify(t)
```

Esercizio 1.1. *Utilizzando le funzioni del pacchetto* Symbolic Math Toolbox*, si calcoli la derivata prima, seconda e l'integrale indefinito delle funzioni*

1. $f(x) = \sqrt{x^2+1}$
2. $g(x) = sin(x^3) + cosh(x)$

Soluzione:

```
syms x
f=sqrt(x^2 +1);
df=diff(f);
df2=diff(f,2);
df_simpl=simplify(df)
df2_simpl=simplify(df2)
intg_f=int(f);
intg_f_simp=simplify(intg_f)

g=sin(x^3)+cosh(x);
dg=diff(g);
dg2=diff(g,2);
dg_simpl=simplify(dg)
dg2_simpl=simplify(dg2)
intg_g=int(g)
```

L'integrazione può presentare diverse volte delle difficoltà. Spesso una forma chiusa dell'integrale non esiste, oppure a volte, MATLAB non è capace di trovarla. La funzione `g` dell'Esercizio 1.1 ne è un esempio. In questi casi MATLAB restituisce semplicemente l'espressione irrisolta di `int(g)`, senza avere effettuato alcun calcolo. In questo caso, si può calcolare un integrale indefinito approssimato, prima approssimando la funzione integranda con il suo polinomio di Taylor, e calcolando successivamente l'integrale di quest'ultimo.

Per espressioni a più variabili simboliche, i comandi `diff(Espr,var)` e `int(Espr,var)` vengono utilizzati per specificare la variabile rispetto a cui si sta derivando oppure integrando. Infine, ricordiamo che i comandi `int` e `diff` possono essere entrambi applicati a matrici. In questo caso le operazioni vengono fatte elemento per elemento.

Per impostazione predefinita, il comando `taylor(f)` resitutisce la serie di Taylor del quinto ordine di $f(x)$ attorno allo zero, come mostrato dal risultato dell'esempio

```
syms x
taylor(log(1+x))
```

Il comando `taylortool` fornisce un'interfaccia grafica a `taylor`, plottando sia la funzione che la serie di Taylor, come mostrato nelle Figure 1 e 2 per le funzioni $log(1+x)$ e $sin(tan(x)) - tan(sin(x))$.

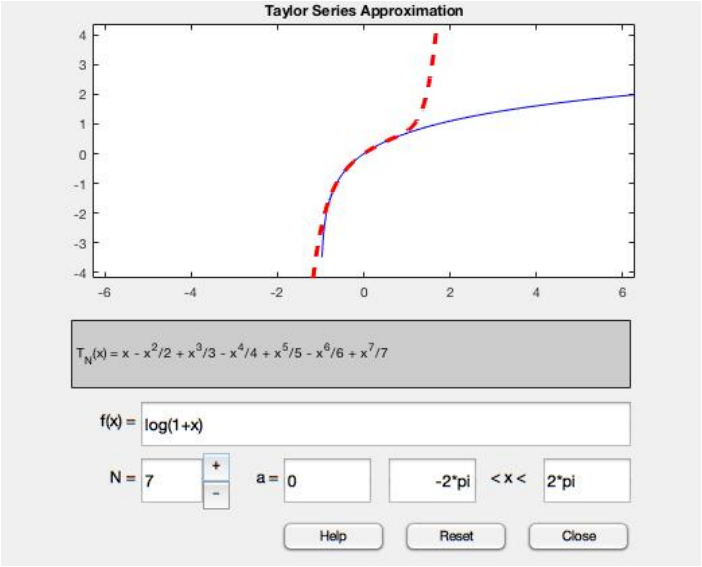


Figure 1: Finestra Taylor Tool per la funzione $f(x) = log(1+x)$.

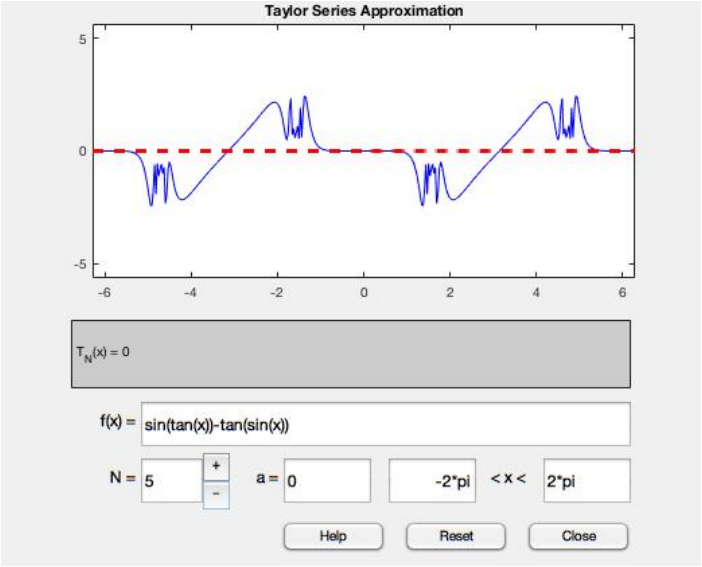


Figure 2: Finestra Taylor Tool per la funzione $f(x) = sin(tan(x)) - tan(sin(x))$.

Il comando `pretty` permette di stampare risultati complicati in formato plain-text. Esso non è consigliato da MATLAB, ma a volte può essere utile per leggere meglio risultati molto complicati, come nel seguente esempio

```
syms x
y=int(sqrt(tan(x)))
pretty(y)
```

Per calcolare integrali definiti $\int_a^b f(x)\,dx$ è sufficiente specificare gli estremi dell'intervallo al comando `int`(f,a,b). In questi casi può tornare utile il comando `double` che converte in forma numerica un oggetto simbolico che è scritto in forma esatta.

Esempio 1.3. *Si calcoli l'integrale definito* $\int_0^1 \frac{arctan(x)}{\sqrt{x^3}}\,dx$.

```
syms x
y = atan(x)/x^(3/2)
risultato=int(y,0,1)
double(risultato)
```

2 Esercizi complementari

2.1 Risoluzione di sistemi di equazioni

Il comando `solve` può essere utilizzato per risolvere un sistema di equazioni. Se il numero delle equazioni e delle variabili coincidono, la soluzione è numerica. Altrimenti, se il numero delle variabili è maggiore del numero delle equazioni, la soluzione è simbolica. Una possibile sintassi è la seguente

```
[var1, var2, ...] = solve( eq1, eq2, ... )
```

Un sistema di equazioni può avere una oppure diverse soluzioni. Se il sistema ha una soluzione, allora le variabili `var1, var2, ...` avranno un valore numerico, altrimenti ciascuna variabile avrà più valori.

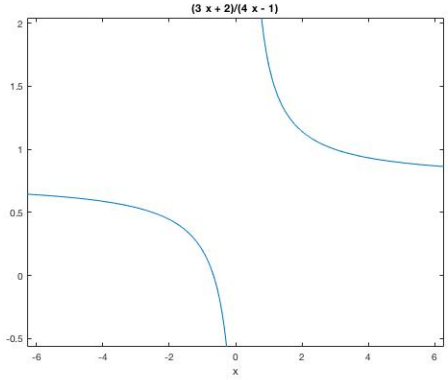
Esercizio 2.1. *L'equazione di un cerchio nel piano* $x-y$ *di raggio* R *e centro il punto (2,4) è data da* $(x-2)^2+(y-4)^2=R^2$. *Determinare le coordinate dei punti di intersezione (in funzione di* R*), del cerchio con la retta di equazione* $y=\frac{x}{2}+1$.

Soluzione:

```
syms x y R
[xs,ys]=solve((x-2)^2+(y-4)^2-R^2, y-x/2-1)
```

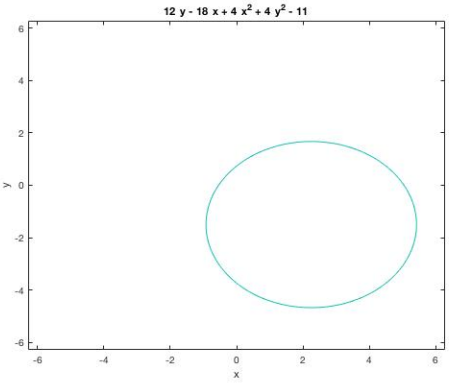
2.1.1 Visualizzazione grafica di funzioni simboliche

MATLAB consente la visualizzazione grafica di espressioni simboliche tramite il comando `ezplot`. Per le espressioni simboliche ad una sola variabile simbolica, MATLAB considera l'espressione come una funzione, e il comando `ezplot` genera il grafico della funzione in termini della variabile.




```
syms x
S=(3*x+2)/(4*x-1)
ezplot(S)
```

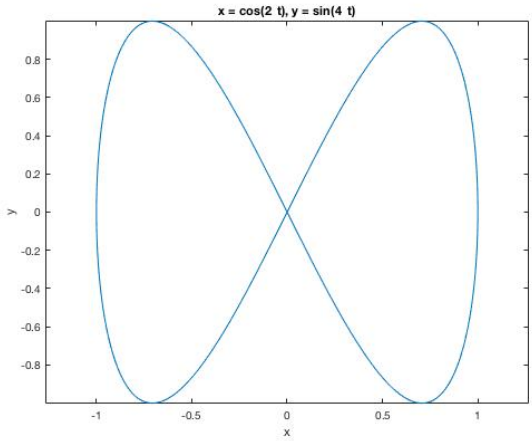
Per espressioni simboliche a due variabili, MATLAB considera l'espressione uguagliata a zero, e genera il grafico di una variabile in termini dell'altra. La variabile che è prima in ordine alfabetico è quella che verrà considerata come variabile indipendente.




```
syms x y
S=4*x^2-18*x+4*y^2+12*y-11
ezplot(S)
```

Infine, il comando `ezplot` consente la visualizzazione grafica di funzioni definite in forma parametrica.

Esercizio 2.2. *Generare il grafico della funzione parametrica* $(cos(2t), sin(4t))$ *con l'aiuto del comando* `ezplot`.



Soluzione:

```
syms t
x=cos(2*t);
y=sin(4*t);
ezplot(x,y)
```

Ultima versione aggiornata: 6 Maggio 2020

Link alle lezioni precedenti:
Lezioni 5-6
Lezioni 3-4
Lezioni 1-2

Referenze

1. MATLAB® The language of technical computing: computation, visualization, programming, The MathWorks Inc
2. MATLAB: An introduction with applications, A. Gilat, Wiley
3. Scientific Computing with MATLAB and Octave, A. Quarteroni, , F. Saleri, P. Gervasio, Springer
4. MATLAB Guide1 D. J. Higham and N. J. Higham, SIAM
5. Numerical Computing with MATLAB, C. Moler, SIAM