Calcolo Simbolico 1

- **Introduzione**

Ahhh!!! Ci voleva proprio questo bel bagnetto!!! Magari riuscirò pure a far sparire l'effetto mozzarella della mia pelle, prima che finiscano le lezioni!!!

...Ehm....

...Dicevamo... Come vi avevo già accennato, uno degli aspetti più importanti di *Mathematica* è il fatto che permette di eseguire con la stessa straordinaria facilità e velocità sia calcoli numerici che, sopratutto, calcoli simbolici. Possiamo scrivere qualcosa del tipo

```
In[1]:= 3 + 5 * (3 + 7)
```

$$\text{Out[1]= } 53$$

ma possiamo anche, se vogliamo, eseguire calcoli che includano elementi simbolici

```
In[2]:= x + 5 x + 2
```

$$\text{Out[2]= } 2 + 6\,x$$

Mathematica riordina automaticamente gli elementi:

```
In[3]:= x^2 + 2 + 4 x
```

$$\text{Out[3]= } 2 + 4\,x + x^2$$

E applica, automaticamente, alcune semplici semplificazioni:

```
In[4]:= Sqrt[1 + x]^4
```

$$\text{Out[4]= } (1 + x)^2$$

Come potete vedere, la radice quadrata è magicamente scomparsa

A volte, facciamo delle elaborazioni algebriche, ottenendo una particolare espressione, e poi ci piacerebbe sapere il valore che assumerebbe quell'espressione se andiamo a sostituire dei valori determinati alle varie incognite. Possiamo fare questo tramite l'operatore di sostituzione:

In[5]:=
```
3 + x + 24 x^4 + Sqrt[x] /. x → 4
```

Out[5]=
```
6153
```

L'operatore di sostituzione è dato dalla combinazione /. dopo l'espressione, e dalle regole di sostituzione: la regola di sostituzione è data scrivendo l'incognita da sostituire, dopo -> ed infine il valore che vogliamo al posto dell'incognita. Ovviamente, dato che conoscete le liste, a questo punto (vero?), potete intuire che possiamo andare a sostituire più incognite in una sola volta usando una lista per le regole di sostituzione, come in quest'esempio:

In[6]:=
```
x^2 + y^2 - 3 x y /. {x → 1, y → 5}
```

Out[6]=
```
11
```

Se vogliamo che la sostituzione sia invece permanente, conviene andare a definire una variabile che abbia il nome dell'incognita:

In[7]:=
```
x = 1; y = 5;
```

Notate, fra l'altro, un altra caratteristica di *Mathematica:* possiamo eseguire più comandi in una volta sola, concatenandoli tramite il punto e virgola, e premendo solo alla fine Shift+Invio. Una volta eseguita questa operazione, ogni volta che il programma incontrerà la variabile, andrà sempre a sostituire il valore corrispondente:

In[8]:=
```
x^2 + y^2 - 3 x y
```

Out[8]=
```
11
```

Per cancellare il valore di una variabile, e renderla ancora incognita, possiamo usare l'assegnamento del punto oppure il comando Clear:

In[9]:=
```
x =.
```

In[10]:=
```
Clear[y]
```

Andando a ricalcolarci l'espressione, vediamo come alle due incognite non è associato più nessun valore:

In[11]:=
```
x^2 + y^2 - 3 x y
```

Out[11]=
$$x^2 - 3\,x\,y + y^2$$

Inoltre, dato che possiamo associare alle variabili sia espressioni numeriche che simboliche, lo stesso possiamo fare per le sostituzioni, che quindi non sono limitate solo a valori numerici:

```
In[12]:= x^2 + 3 y + z /. z → 4 y
```

$$\text{Out[12]= } x^2 + 7\,y$$

- **Manipolazioni Algebriche**

Una volta ottenute e scritte le nostre espressioni, può risultare molto utile effettuare delle determinate manipolazioni per scriverle nella forma che più ci aggrada. Andiamo, per esempio, a definire la seguente equazione algebrica:

```
In[13]:= eq = (1 + x)^4
```

$$\text{Out[13]= } (1 + x)^4$$

Ci sono due funzioni basilari, che ci permettono di espandere e contrarre i termini di un'espressione algebrica:

`Expand[`*expr*`]`	moltiplica tuti i prodotti e potenze, dando il risultato come somma di prodotti
`Factor[`*expr*`]`	scrive *expr* come prodotto di fattori minimali

Se volessimo utilizzare le funzioni per l'espressione che abbiamo appena scritto, possiamo avere:

```
In[14]:= Expand[eq]
```

$$\text{Out[14]= } 1 + 4\,x + 6\,x^2 + 4\,x^3 + x^4$$

Mentre, Factor lo riporta nella forma originale

```
In[15]:= Factor[%]
```

$$\text{Out[15]= } (1 + x)^4$$

Anche se in genere non abbiamo problemi, bisogna prestare comunque, un attimino di attenzione nell'usare queste due funzioni. Se, per esempio, volessimo usare Expand per l'espressione $(1 + x)^{100000}$, possiamo subito notare che avremo sullo schermo (sempre che la memoria del computer non ci tradisca prima), qualcosa che non avremmo mai voluto avere. Ma questo capita nel 99,9999% dei casi se effettuiamo qualche malaugurato errore di battitura. In questi tre anni che conosco *Mathematica* non mi è mai capitato di dover scrivere apposta formule così lunghe...

Se, invece, ci preme poter scrivere la forma più semplice di un espressione, non sempre usare i due

comandi appena citati è utile. Infatti non è sempre verificabile subito se la forma più semplice è data dall'una oppure dall'altra formula, e possiamo solo provare, come in questo caso:

```
In[16]:= eq = x^10 - 1
```

$$\text{Out[16]= } -1 + x^{10}$$

```
In[17]:= Expand[eq]
```

$$\text{Out[17]= } -1 + x^{10}$$

```
In[18]:= Factor[eq]
```

$$\text{Out[18]= } (-1 + x)\,(1 + x)\,(1 - x + x^2 - x^3 + x^4)\,(1 + x + x^2 + x^3 + x^4)$$

In questo caso la forma più semplice, al contrario del caso precedente, si ottiene con il comando Factor. Expand lavora in maniera diversa quando abbiamo a che fare con delle funzioni razionali fratte:

```
In[19]:= expr = ((x - 2) (x + 5) (x - 4)) / ((x - 1) (x - 1) (x - 8) (x + 1))
```

$$\text{Out[19]= } \frac{(-4 + x)\,(-2 + x)\,(5 + x)}{(-8 + x)\,(-1 + x)^2\,(1 + x)}$$

```
In[20]:= Expand[expr]
```

$$\text{Out[20]= } \frac{40}{(-8 + x)\,(-1 + x)^2\,(1 + x)} - \frac{22\,x}{(-8 + x)\,(-1 + x)^2\,(1 + x)} - \frac{x^2}{(-8 + x)\,(-1 + x)^2\,(1 + x)} + \frac{x^3}{(-8 + x)\,(-1 + x)^2\,(1 + x)}$$

Possiamo vedere che il denominatore non viene intaccato dal comando, ed anche che abbiamo in questo caso separato l'espressione in più espressioni razionali fratte. Questo a volte non è conveniente. Per espandere questo tipo di espressioni ci sono altri comandi più specifici:

`ExpandNumerator[`*expr*`]`	espande solamente il numeratore
`ExpandDenominator[`*expr*`]`	espande solamente il denominatore
`Expand[`*expr*`]`	espande il numeratore, distribuendo il denominatore su ogni termine
`ExpandAll[`*expr*`]`	espande sia il numeratore che il denominatore

Come possiamo vedere, quindi, se vogliamo espandere per esempio il numeratore, ma non vogliamo che sia espanso compretamente come prima, dobbiamo usare il comando appropriato:

In[21]:=
```
ExpandNumerator[expr]
```

Out[21]=
$$\frac{40 - 22\,x - x^2 + x^3}{(-8 + x)\,(-1 + x)^2\,(1 + x)}$$

In questa maniera, espandiamo il numeratore, ma la scrittura è più compatta. Possiamo anche espandere il denominatore, oppure entrambe le cose, tramite gli altri comandi:

In[22]:=
```
ExpandDenominator[expr]
```

Out[22]=
$$\frac{(-4 + x)\,(-2 + x)\,(5 + x)}{-8 + 9\,x + 7\,x^2 - 9\,x^3 + x^4}$$

In[23]:=
```
ExpandAll[expr]
```

Out[23]=
$$\frac{40}{-8 + 9\,x + 7\,x^2 - 9\,x^3 + x^4} - \frac{22\,x}{-8 + 9\,x + 7\,x^2 - 9\,x^3 + x^4} - \frac{x^2}{-8 + 9\,x + 7\,x^2 - 9\,x^3 + x^4} + \frac{x^3}{-8 + 9\,x + 7\,x^2 - 9\,x^3 + x^4}$$

Se vogliamo una riscrittura più compatta, possiamo combinare per esempio due dei comandi precedenti:

In[24]:=
```
ExpandNumerator[ExpandDenominator[expr]]
```

Out[24]=
$$\frac{40 - 22\,x - x^2 + x^3}{-8 + 9\,x + 7\,x^2 - 9\,x^3 + x^4}$$

In questa maniera abbiamo espanso tutto, ma avitando di scrivere un'espressione fratta per ogni termine del numeratore...

ExpandAll, pur espandendo tutto quanto, ha un'opzione molto interessante:

`ExpandAll[`*expr*`, `*patt*`]`, etc.	espande quelle parti dell'espressione che contengono *patt*

Supponiamo di avere un'espressione con più incognite:

In[25]:=
```
expr = (x - y)^4 + (x - 3)^2
```

Out[25]=
$$(-3 + x)^2 + (x - y)^4$$

Proviamo, adesso, ad espanderlo completamente:

In[26]:=
```
ExpandAll[expr]
```

Out[26]=
$$9 - 6\,x + x^2 + x^4 - 4\,x^3\,y + 6\,x^2\,y^2 - 4\,x\,y^3 + y^4$$

Quello che ci aspettavamo, in fondo; supponiamo, adesso, di voler espandere la parte dell'espressione che contiene la y: in questo caso basta specificarla nel comando ExpandAll:

```
In[27]:= ExpandAll[expr, y]
Out[27]= (-3 + x)^2 + x^4 - 4 x^3 y + 6 x^2 y^2 - 4 x y^3 + y^4
```

In questa maniera abbiamo lasciato inalterata la parte che non contiene la y; questo è un metodo abbastanza potente per poter espandere solo le porzioni che ci interessano.

Tuttavia *Mathematica* ci viene incontro con due comandi molto potenti per la semplificazione delle espressioni:

`Simplify[`*expr*`]`	cerca la forma più semplice utilizzando un set di semplificazioni algebriche standard
`FullSimplify[`*expr*`]`	cerca la forma più semplice applicando un set di regole di semplificazione molto più ampio

Questi due comandi permettono di trovare la forma più compatta di scrivere un'equazione:

```
In[28]:= Simplify[x^2 + 4 x + 4]
Out[28]= (2 + x)^2

In[29]:= Simplify[x^10 - 1]
Out[29]= -1 + x^10
```

Si può notare che nel primo caso la forma più compatta è quella ottenuta con Factor, mentre nel secondo caso quella ottenuta con Expand, e che in entrambi i casi Simplify restituisce il risultato corretto. Tuttavia questo comando va ben oltre le semplici equazioni algebriche; permette infatti di semplificare una vasta gamma di espressioni.

```
In[30]:= Simplify[1/(4 (-1 + x)) - 1/(4 (1 + x)) - 1/(2 (1 + x^2))]
Out[30]= 1/(-1 + x^4)
```

Quando, tuttavia, le espressioni sono complicate e usano funzioni avanzate, Simplify potrebbe non farcela da solo, e qua entra in gioco il comando FullSimplify, che utilizza un numero di regole molto più vasto, includendo anche equazioni trascendentali e semplificazioni di funzioni avanzate, come la funzione Gamma:

In[31]:=
```
eq = Gamma[x] Gamma[1 - x]
```

Out[31]=
```
Gamma[1 - x] Gamma[x]
```

In[32]:=
```
Simplify[eq]
```

Out[32]=
```
Gamma[1 - x] Gamma[x]
```

In[33]:=
```
FullSimplify[eq]
```

Out[33]=
```
π Csc[π x]
```

A questo punto, si sarebbe tentati di utilizzare sempre il secondo comando, ma io lo sconsiglio: è vero che è più potente, ma è anche vero che lo è a prezzo di un costo computazionale esponenzialmente più elevato; questo potrebbe fare la differenza per espressioni grandi, per cui potrebbe impiegare un tempo esagerato per portare a termine l'operazione. Usatelo, quindi, solo se ne vale la pena, e se Simplify da solo non vi soddisfa. Anche se, per i calcoli che siamo soliti fare, non si perde in fondo tutto questo tempo...

Inoltre, ci sono anche altre funzioni che permettono di manipolare ulteriormente le espressioni, in modo da poter, magari, visualizzarle nel modo che preferiamo:

`Expand[`*expr*`]`	moltiplica tutti i prodotti e potenze
`ExpandAll[`*expr*`]`	applica `Expand` ovunque
`Factor[`*expr*`]`	scrive sottoforma di prodotto di fattori
`Together[`*expr*`]`	scrive tutti i termini con un comun denominatore
`Apart[`*expr*`]`	separa l'espressione in termini con semplici denominatri
`Cancel[`*expr*`]`	cancella i fattori comuni fra numeratore e denominatore
`Collect[`*expr*`, `*x*`]`	Scrive l'equazione raggruppando le potenze di x
`FactorTerms[`*expr*`, `*x*`]`	Scompone in due fattori, dove in uno non compare il termine x

Supponiamo, di avere per esempio

In[34]:=
```
espr = (x - 1)^2 (2 + x) / ((1 + x) (x - 3)^2)
```

Out[34]=
$$\frac{(-1+x)^2\,(2+x)}{(-3+x)^2\,(1+x)}$$

In[35]:=
```
Expand[espr]
```

Out[35]=
$$\frac{2}{(-3+x)^2\,(1+x)} - \frac{3x}{(-3+x)^2\,(1+x)} + \frac{x^3}{(-3+x)^2\,(1+x)}$$

Come possiamo vedere, Expand espande solamente i termini presenti al numeratore; se vogliamo fare la stessa cosa anche per il denominatore, dobbiamo utilizzare la funzione ExpandAll:

```
In[36]:= ExpandAll[espr]
```

$$\text{Out[36]}= \frac{2}{9+3x-5x^2+x^3}-\frac{3x}{9+3x-5x^2+x^3}+\frac{x^3}{9+3x-5x^2+x^3}$$

Riscriviamo tutto sotto comun denominatore: come facciamo? Con Together, naturalmente, dato che fa proprio questo:

```
In[37]:= Together[%]
```

$$\text{Out[37]}= \frac{2-3x+x^3}{(-3+x)^2(1+x)}$$

Aggiungiamo Factor, e riotteniamo quello che avevamo in partenza:

```
In[38]:= Factor[%]
```

$$\text{Out[38]}= \frac{(-1+x)^2(2+x)}{(-3+x)^2(1+x)}$$

Adesso, supponiamo di avere un'espressione in due variabili:

```
In[39]:= espr2 = (1 + x)^2 + (x - y)^3
```

$$\text{Out[39]}= (1+x)^2+(x-y)^3$$

Vediamo di raggruppare tuti i termini in x:

```
In[40]:= Collect[espr2, x]
```

$$\text{Out[40]}= 1+x^3+x^2(1-3y)-y^3+x(2+3y^2)$$

Come si può vedere, abbiamo ottenuto termini dove x^n moltiplica vari termini, più un eventuale termine noto... un'espressione in x, insomma... Se, invece, vogliamo ottenere la stessa espressione, ma evidenziando l'incognita y:

```
In[41]:= Collect[espr2, y]
```

$$\text{Out[41]}= x^3+(1+x)^2-3x^2y+3xy^2-y^3$$

Il comando Apart permette di scrivere un rapporto di polinomi come somma di frazioni semplici:

```
In[42]:= Apart[(x^2 - 5 x + 3)/((x - 5) (x - 3) (x + 4))]

Out[42]= 1/(6 (-5 + x)) + 3/(14 (-3 + x)) + 13/(21 (4 + x))
```

Il comando Cancel fa quello che si farebbe utilizzando una frazione numerica; riduce la frazione ai minimi termini, elidendo tutti i fattor comuni:

```
In[43]:= Expand[(x + 1) (x + 3)] / Expand[(x + 1) (x + 5) (x - 6)]

Out[43]= (3 + 4 x + x^2)/(-30 - 31 x + x^3)
```

A questo punto, si potrebbe pensare che è ridotto ai minimi termini, ma non è vero. *Mathematica* in questo caso ha prima creato le espressioni, e poi ha fatto il rapporto. In questo caso, se non lo comunichiamo con il comando Cancel, *Mathematica* non controlla automaticamente se ci sono dei fattori comuni, cosa che invece sappiamo per costruzione: per poterli cancellare, basta utilizzare il comando...

```
In[44]:= Cancel[%]

Out[44]= (3 + x)/(-30 - x + x^2)
```

Vediamo come, prima di tutto abbiamo eliminato il fattor comune, e come il risultato sia sempre dato sotto forma non di prodotto di binomi, ma come polinomi completi. Se avessimo creato l'espressione razionale fratta senza Expand, *Mathematica* non avrebbe espanso i termini, cancellando automaticamente (perchè sono presenti in maniera esplicita) i fattori comuni:

```
In[45]:= (x + 1) (x + 3) / ((x + 1) (x + 5) (x - 6))

Out[45]= (3 + x)/((-6 + x) (5 + x))
```

In questa maniera, però, non otteniamo automaticamente il denominatore in forma polinomiale, come nel caso precedente.

Il comando seguente che andiamo ad esaminare è Collect: esso permette di raccogliere i termini di un'espressione, e di scriverla sotto forma di prodotto di potenze di una determinata variabile.

Supponiamo di avere il seguente polinomio:

In[46]:=
```
xyz = Expand[(x + y) (x + 3 z) (z - 5 y) (z - 2 x)]
```

Out[46]=
$$10\,x^3\,y + 10\,x^2\,y^2 - 2\,x^3\,z + 23\,x^2\,y\,z + 25\,x\,y^2\,z - 5\,x^2\,z^2 - 20\,x\,y\,z^2 - 15\,y^2\,z^2 + 3\,x\,z^3 + 3\,y\,z^3$$

A questo punto, possiamo decidere se scrivere questa espressione come polinomio in x, y, oppure z, specificando la variabile con cui vogliamo raccogliere i termini. Per esempio, se li volessimo raccogliere in x:

In[47]:=
```
Collect[xyz, x]
```

Out[47]=
$$x^3\,(10\,y - 2\,z) - 15\,y^2\,z^2 + 3\,y\,z^3 + x^2\,(10\,y^2 + 23\,y\,z - 5\,z^2) + x\,(25\,y^2\,z - 20\,y\,z^2 + 3\,z^3)$$

Si può notare che in questo caso abbiamo raccolto, appunto, i termini in x, con i rispettivi coefficienti. Analogamente possiamo fare per gli altri due casi:

In[48]:=
```
Collect[xyz, y]
```

Out[48]=
$$-2\,x^3\,z - 5\,x^2\,z^2 + 3\,x\,z^3 + y^2\,(10\,x^2 + 25\,x\,z - 15\,z^2) + y\,(10\,x^3 + 23\,x^2\,z - 20\,x\,z^2 + 3\,z^3)$$

In[49]:=
```
Collect[xyz, z]
```

Out[49]=
$$10\,x^3\,y + 10\,x^2\,y^2 + (-2\,x^3 + 23\,x^2\,y + 25\,x\,y^2)\,z + (-5\,x^2 - 20\,x\,y - 15\,y^2)\,z^2 + (3\,x + 3\,y)\,z^3$$

Naturalmente la maniera più conveniente dipende dal nostro problema, e da cosa dobbiamo mettere in evidenza...

Possiamo considerare l'ultimo comando del gruppo, FactorTerms, come il duale di quello che abbiamo appena visto; infatti, il suo scopo consiste nel dividere l'espressione come prodotto di altre due espressioni, in cui però in una non compare l'ingognita specificata:

In[50]:=
```
expr = x^2 y^2 + 3 x y - x z^3
```

Out[50]=
$$3\,x\,y + x^2\,y^2 - x\,z^3$$

Possiamo adesso dividere l'espressione in due termini, in cui in uno non compare, per esempio, la z:

In[51]:=
```
FactorTerms[expr, z]
```

Out[51]=
$$-x\,(-3\,y - x\,y^2 + z^3)$$

Analogamente, se volessimo che in uno dei due termini non compaia la x:

In[52]:=
```
FactorTerms[expr, x]
```

Out[52]= $3\,x\,y + x^2\,y^2 - x\,z^3$

In questo caso, evidentemente, la cosa non si può fare... Non si può pretendere l'impossibile!!! Tanto per dovere di cronaca, vediamo se si può fare nell'ultima variabile rimasta:

In[53]:=
```
FactorTerms[expr, y]
```

Out[53]= $x\,(3\,y + x\,y^2 - z^3)$

Effettivamente, c'è tutto un insieme di comandi Factor che permettono di fattorizzare il polinomio in maniera diversa a seconda dei casi:

`Factor[`*poly*`]`	fattorizza un polinomio
`FactorSquareFree[`*poly*`]`	estrapola i fattori quadrati dal polinomio
`FactorTerms[`*poly*`, `*x*`]`	fattorizza i termini che non contengono *x*
`FactorList[`*poly*`]`, `FactorSquareFreeList[`*poly*`]`, `FactorTermsList[`*poly*`]`	restituiscono il risultato come lista di fattori, invece che come prodotto

Costruiamoci il nostro polinomio:

In[54]:=
```
t = Expand[3 (x - 4) (x^2 - x + 2) (x + 3)^4]
```

Out[54]= $-1944 - 1134\,x - 567\,x^2 - 693\,x^3 - 294\,x^4 + 21\,x^6 + 3\,x^7$

Factor e FactorTerms abbiamo giàò visto come funzionano:

In[55]:=
```
Factor[t]
```

Out[55]= $3\,(-4 + x)\,(3 + x)^4\,(2 - x + x^2)$

In[56]:=
```
FactorTerms[t, x]
```

Out[56]= $3\,(-648 - 378\,x - 189\,x^2 - 231\,x^3 - 98\,x^4 + 7\,x^6 + x^7)$

Quello che non avevamo visto è FactorSquareFree:

In[57]:=
```
FactorSquareFree[t]
```

Out[57]= $3\,(3 + x)^4\,(-8 + 6\,x - 5\,x^2 + x^3)$

Come abbiamo visto, non è stato fattorizzato completamente: sono stati estratti soltanto il fattore costante, e il fattore che è un quadrato (possiamo considerare 4 come quadrato del quadrato...), moltiplicando questi per il restante polinomio.

Gli altri comandi, invece, servono nel caso che, per qualche motivo, abbiamo bisogno della lista dei fattori, invece che del loro prodotto. Questo può essere utile quando dobbiamo estrarre particolari fattori per andare ad utilizzarli da qualche altra parte:

```
In[58]:= FactorList[t]
Out[58]= {{3, 1}, {-4 + x, 1}, {3 + x, 4}, {2 - x + x^2, 1}}

In[59]:= FactorTermsList[t, x]
Out[59]= {3, 1, -648 - 378 x - 189 x^2 - 231 x^3 - 98 x^4 + 7 x^6 + x^7}

In[60]:= FactorSquareFreeList[t]
Out[60]= {{3, 1}, {3 + x, 4}, {-8 + 6 x - 5 x^2 + x^3, 1}}
```

Inoltre, questi comandi hanno anche delle opzioni che permettono di manipolare il risultato che otteniamo. Le opzioni sono, per i comandi di *Mathematica*, degli argomenti, che si scrivono nella seguente forma:

```
f[argomenti, opzione1 → val1, opzione2 → opz2 ...]
```

Permettono di gestire meglio il comportamento di varie funzioni. Comunque lo vedremo meglio più avanti. Qua vediamo soltanto come funzionano per Factor. Questo comando ha quattro opzioni: Extension, GaussianIntegers, Modulus, Trig. Il primo serve per poter utilizzare particolari numeri nella scomposizione, che di default non verrebbero considerati. Il secondo permette, in particolare, di utilizzare nella scomposizione gli interi di Gauss. La terza opzione permette di effettuare la scomposizione con un modulo specificato, mentre la quarta permette di scomporre tramite coefficienti trigonometrici. creiamoci un altro polinomio per l'occasione:

```
In[61]:= tt = Expand[5 (4 x^2 + 1) (x^3 - 1) (x^2 - 2)]
Out[61]= 10 + 35 x^2 - 10 x^3 - 20 x^4 - 35 x^5 + 20 x^7

In[62]:= Factor[tt]
Out[62]= 5 (-1 + x) (-2 + x^2) (1 + x + x^2) (1 + 4 x^2)
```

Come vedete, tutto rientra nella norma. Tuttavia proviamo ad utilizzare l'opzione GaussianIntegers, impostandola su True, mentre di default è False:

```
In[63]:= Factor[tt, GaussianIntegers → True]
Out[63]= 5 (-1 + x) (-i + 2 x) (i + 2 x) (-2 + x^2) (1 + x + x^2)
```

Come possiamo vedere, abbiamo ottenuto una ulteriore scomposizione del fattore $(1+4x^2)$, facendolo diventare $(2x-i)(2x+i)$, scomponendo parte reale e parte immaginaria, cosa che di default Factor non fa.

Con Modulus, invece, possiamo specificare il modulo della scomposizione:

```
In[64]:= Factor[tt, Modulus → 3]
Out[64]= 2 (2 + x)^3 (1 + x^2)^2
```

Il risultato è scomposto tenendo conto del modulo dei polinomi.

Extension, dal canto suo, permette di utilizzare altri valori per la scomposizione, introducendo dei numeri voluti da noi:

```
In[65]:= Factor[tt, Extension → Sqrt[2]]
Out[65]= -5 (√2 - x) (-1 + x) (√2 + x) (1 + x + x^2) (1 + 4 x^2)
```

Come abbiamo potuto vedere, dicendo a Factor che poteva utilizzare anche $\sqrt{2}$ nelle sue operazioni, ha correttamente scomposto in questo modo anche il fattore (x^2-2).

Inoltre, potete anche dire a *Mathematica* di provare da solo ad ampliare il numero di valori possibili di fattorizzazione, aggiungendo altri tipi di numeri oltre a quello interi, ponento Extension->-Automatic.

Invece, se impostiamo Trig su True, possiamo trattare anche coefficienti trigonometrici:

```
In[66]:= tr = x^4 - Sin[π / 7] x^2 + 3
Out[66]= 3 + x^4 - x^2 Sin[π/7]
```

```
In[67]:= Factor[tr]
Out[67]= 3 + x^4 - x^2 Sin[π/7]
```

In[68]:=
```
Factor[tr, Trig → True]
```

Out[68]=
$$-\frac{1}{2}\,(-1)^{6/7}\,(6\,(-1)^{1/7} - i\,x^2 + (-1)^{11/14}\,x^2 + 2\,(-1)^{1/7}\,x^4)$$

- **Manipolazioni avanzate**

A volte è necessario creare delle manipolaizoni più avanzate di quelle che abbiamo visto prima, ed effettuare operazioni particolari, come trovare il massimo comun divisore fra due polinomi, per esempio. Alcuni comandi avanzati sono mostrati di seguito:

`PolynomialQuotient[`$poly_1$`,` $poly_2$`,` x`]`	trova il quoziente ottenuto dividendo $poly_1$ nella variabile x con $poly_2$, trascurando l'eventuale resto
`PolynomialRemainder[`$poly_1$`,` $poly_2$`,` x`]`	trova il resto della divisione fra il polinomio $poly_1$ in x ed il polinomio $poly_2$
`PolynomialGCD[`$poly_1$`,` $poly_2$`]`	trova il massimo comun divisore fra i due polinomi
`PolynomialLCM[`$poly_1$`,` $poly_2$`]`	trova il minimo comune multiplo fra i due polinomi
`PolynomialMod[`$poly$`,` m`]`	riduce il polinomio $poly$ utilizzando il modulo m
`Resultant[`$poly_1$`,` $poly_2$`,` x`]`	trova la risultante fra i die polinomi
`Subresultants[`$poly_1$`,` $poly_2$`,` x`]`	trova i principali valori delle sottorisultanti fra i due polinomi
`GroebnerBasis[{`$poly_1$`,` $poly_2$`, … }, {`x_1`,` x_2`, … }]`	trova la base di Gröbner per i polinomi $poly_i$
`GroebnerBasis[{`$poly_1$`,` $poly_2$`, … }, {`x_1`,` x_2`, … }, {`y_1`,` y_2`,`	trova la base di Gröbner ottenuta eliminando le y_i
`PolynomialReduce[`$poly$`,` `{`$poly_1$`,` $poly_2$`, … }, {`x_1`,` x_2`, … }]`	trova la rappresentazione minima di $poly$ nei termini $poly_i$

Come potete vedere, questi comandi permettono di risolvere parecchi problemi che hanno a che fare con polinomi. Danno il meglio di sè, però, quando i polinomi sono ordinari, cioè con esponenti interi e con numeri razionali come coefficienti. Niente vieta però di vedere come va a finire utilizzandoli con altri tipi, naturalmente.

Il primo comando, Polynomial quotient, restituisce il risultato della divisione fra due polinomi. Infatti, come ben sapete, possiamo scrivere il rapporto di due polinomi in questa maniera:

$$\frac{p(x)}{q(x)} = a(x) + \frac{b(x)}{q(x)}$$

Il comando restrituisce proprio $a(x)$ che sarebbe il risultato della divisione, mentre $b(x)$ è il risultato restituito da PolynomialRemainder, che restituisce il resto. Vediamo di definire i due polinomi:

In[69]:=
```
p = x^7 - 5 x^2 + 4 x - 10; q = -2 x^5 - x^4 + 7 x^3 - 2 x^2 + 9 x - 4;
```

Vediamo di vedere il quoziente di questo rapporto:

In[70]:=
```
a = PolynomialQuotient[p, q, x]
```

Out[70]=
$$-\frac{15}{8}+\frac{x}{4}-\frac{x^2}{2}$$

Vediamo, adesso, di trovare $b(x)$, cioè il resto:

In[71]:=
```
b = PolynomialRemainder[p, q, x]
```

Out[71]=
$$-\frac{35}{2}+\frac{175\,x}{8}-13\,x^2+\frac{145\,x^3}{8}-\frac{37\,x^4}{8}$$

Come potete vedere, il polinomio ha un grado inferiore a quello di q, cosa che fra l'altro era scontata, dato che nel resto l'ordine del muneratore è per forza di cose inferiore a quello del denominatore: vediamo adesso se effettivamente quoziente e resto sono quelli che abbiamo trovato:

In[72]:=
```
p / q
```

Out[72]=
$$\frac{-10+4\,x-5\,x^2+x^7}{-4+9\,x-2\,x^2+7\,x^3-x^4-2\,x^5}$$

Applichiamo la definizione:

In[73]:=
```
a + b / q
```

Out[73]=
$$-\frac{15}{8}+\frac{x}{4}-\frac{x^2}{2}+\frac{-\frac{35}{2}+\frac{175\,x}{8}-13\,x^2+\frac{145\,x^3}{8}-\frac{37\,x^4}{8}}{-4+9\,x-2\,x^2+7\,x^3-x^4-2\,x^5}$$

In[74]:=
```
Simplify[%]
```

Out[74]=
$$\frac{10-4\,x+5\,x^2-x^7}{4-9\,x+2\,x^2-7\,x^3+x^4+2\,x^5}$$

Come avete potuto vedere, i due rapporti sono uguali, segno che abbiamo trovato realmente il risultato della divisione con quei due comandi. D'altronde, avevate qualche dubbio in proposito??? Spero proprio di no, altrimenti buttate questi appunti e datevi all'ippica!!!

PolynomialGCD, invece, restituisce il massimo comun divisore, come potete facilmente intuire dal nome del comando:

In[75]:=
```
PolynomialGCD[p, q]
```

Out[75]=
```
1
```

In questo caso, possiamo dire che i polinomi sono primi fra di loro... non sono stato fortunatissimo nella scelta, per l'esempio. Pazienza, vorrà dire che mi farò altri due polinomi ad hoc:

```
In[76]:= pp = 2 - x - 3 x^2 + x^3 + x^4; qq = 6 - x - 4 x^2 - x^3;

In[77]:= PolynomialGCD[pp, qq]

Out[77]= -2 + x + x^2
```

Questa volta l'ho azzeccata... Possiamo vedere come, in questo caso, non ci sia bisogno di specificare una variabile per la creazione del massimo comun divisore.

PolynoialLCM restituisce il minimo comune multiplo (che in italiano è MCM, ma in inglese è **L**east **C**ommon **M**ultiple) fra due polinomi:

```
In[78]:= PolynomialLCM[pp, qq]

Out[78]= (-1 + x^2) (6 - x - 4 x^2 - x^3)
```

Niente di particolarmente complicato. Penso che capiate tutti come usare questo comando.

PolynomialMod è un comando che praticamente non ho usato mai, perchè non mi è mai servito. E' l'equivalente di Mod per i numeri, e permette la riduzione in modulo M di un polinomio, sia esso un numero oppure un altro polinomio (in quest'ultimo caso, però, le cose si complicano).

```
In[79]:= pol = 4 x^3 + 7 x - 2;

In[80]:= PolynomialMod[pol, 3]

Out[80]= 1 + x + x^3
```

In questo caso, abbiamo applicato semplicemente Mod ai coefficienti numerici, mediante un fattore anch'esso intero. Naturalmente possiamo fare anche cose più complicate...

```
In[81]:= PolynomialMod[x^5 - x^4 + 5 y x^2 - x, x^2]

Out[81]= -x
```

Come possiamo vedere, abbiamo ottenuto una riduzione più elaborata, di cui fra l'altro non conosco molto bene la teoria, perchè non ne ho mai avuto niente a che fare con questo tipo di operazioni. Per chi ci bazzica, probabilmente ci ha trovato qualcosa con cui divertirsi parecchio.

Un altro comando che non ho mai utilizzato, ma che qualcuno di voi può trovare interessante, è

Resultant, che restituisce la risultante fra due polinomi, che è dato dal prodotti dei termini $a_i - b_j$ che sono le radici dei polinomi. Riprendiamo quelle che avevamo prima:

```
In[82]:= p
Out[82]= -10 + 4 x - 5 x^2 + x^7

In[83]:= q
Out[83]= -4 + 9 x - 2 x^2 + 7 x^3 - x^4 - 2 x^5

In[84]:= Resultant[p, q, x]
Out[84]= 962920402
```

Naturalmente questo funziona anche se abbiamo dei coefficienti letterali:

```
In[85]:= Resultant[p, x^3 + r x^2 - l x + 1, x] // Simplify
Out[85]= 500 + 4 l^6 - 10 l^7 + 575 r + 495 r^2 - 462 r^3 + 680 r^4 - 15 r^5 - 84 r^6 + 100 r^7 +
          l^5 (-5 + 70 r) + 2 l^4 (-5 + 113 r) + l^3 (258 + 625 r - 500 r^2 + 250 r^3) +
          5 l^2 (167 - 153 r - 14 r^2 + 260 r^3 - 48 r^4 + 10 r^5) +
          l (475 - 610 r + 1895 r^2 - 245 r^3 - 404 r^4 + 680 r^5 - 40 r^6)
```

come possiamo vedere, il concetto è identico anche se, in forma letterale, le cose possono cominciare a diventare complicate. Le sottorisultanti si posso avere tramite Subresultants, che contiene gli stessi argomenti:

```
In[86]:= Subresultants[p, q, x]
Out[86]= {962920402, -324201, 133564, 3767, -37, 4}
```

Se il coefficiente del termine di grado più elevato di entrambi i polinomi è pari ad uno, allora il numero delle radici in comune fra i due polinomi è pari al numeri dei primi elementi della lista pari a 0. Supponiamo di avere questi polinomi:

```
In[87]:= p = -3 x + 2 x^2 + x^3 + 6 z - 4 x z - 2 x^2 z;
          q = 48 - 2 x - 57 x^2 + x^3 + 9 x^4 + x^5;
```

A questo punto, possiamo calcolarci le sottorisultanti dei polinomi:

```
In[89]:= Subresultants[p, q, x]
Out[89]= {0, 0, -16 - 20 z + 28 z^2 + 8 z^3, 1}
```

Come possiamo vedere, abbiamo due elementi nulli, il che ci suggerisce che i due polinomi hanno due radici in comune. Infatti possiamo vederlo direttamente:

```
In[90]:= Factor[p]
Out[90]= (-1 + x) (3 + x) (x - 2 z)

In[91]:= Factor[q]
Out[91]= (-2 + x) (-1 + x) (1 + x) (3 + x) (8 + x)
```

Come possiamo vedere, le radici che hanno in comune sono 1 e -3, che sono soluzione di entrambe:

```
In[92]:= {p, q} /. {{x → 1}, {x → -3}}
Out[92]= {{0, 0}, {0, 0}}
```

Come possiamo vedere, sostituendo ai due polinomi le radici, in entrambi i casi sono nulli, come d'altronde era ovvio... La matematica non è un'opinione.

La base di Gröbner ha assunto particolare rilevanza specialmente negli ultimi anni, avendo avuto applicazione in molti campo. In parole povere, trovando la base di Gröbner di un insieme di polinomi, ne ricaviamo altri polinomi, che hanno particolari caratteristiche dipendenti dai polinomi presi in esame. Per esempio, hanno lo stesso numero di radici in comune del set originale, assieme ad altre proprietà che vi conviene andarvi a studiare; qua non si fanno giochini e cosette varie, e *Mathematica* è fatto per chi sa quello che fa, altrimenti che ce li spendete a fare i soldi per comprarlo, dico io?!?!?!?

```
In[93]:= GroebnerBasis[{x + y + 3 z, x^2 - y}, {x, y}]
Out[93]= {-y + y^2 + 6 y z + 9 z^2, x + y + 3 z}
```

Come possiamo vedere, abbiamo ottenuto la stessa soluzione di quella del libro dove è riportato questo esempio (quale libro?!?!? Andate ed esplorate, cari miei adepti....).

Vediamo questo altro esempio:

```
In[94]:= p1 = x^3 - y^2 x^2 + y ;
         p2 = 2 x^3 - 4 y^3;
         p3 = -4 x^3 + 2 y - 2 x^2 y^2 + 12 y^3;
```

Andiamo a trovarci la base di questi tre:

```
In[97]:= GroebnerBasis[{p1, p2, p3}, {x}] // FullSimplify

Out[97]= {y (-1 - 6 y^2 - 12 y^4 - 8 y^6 + 4 y^9),
   y (x + 2 y (2 + y^2 (8 + y (-1 + 2 y (4 + y - 2 y^3))))), x^3 - 2 y^3}
```

Possiamo anche trovare la base ottenuta eliminando le variabili che non sono di nostro interesse, mettendole come terzo argomento della funzione:

```
In[98]:= GroebnerBasis[{p1, p2, p3}, {x}, {y}]

Out[98]= {-2 x^3 - 6 x^8 - 4 x^9 + x^12}
```

L'ultimo comando di questo gruppo che ci rimane da vedere è PolynomialReduce, che restituisce una lista, il cui primo elemento è dato da una lista di una serie di polinomi, ed il secondo elemento da un polinomio minimo, che hanno una particolarità: vedendo l'esempio si chiarisce meglio:

```
In[99]:= p

Out[99]= -3 x + 2 x^2 + x^3 + 6 z - 4 x z - 2 x^2 z

In[100]:= PolynomialReduce[p, {x - 4, x^2 + 5}, {x, z}]

Out[100]= {{21 + 6 x + x^2 - 12 z - 2 x z, 0}, 84 - 42 z}
```

Il risultato è la lista che abbiamo specificato. Se adesso moltiplichiamo ogni elemento della lista che compare nel risultato, ognuno per il corrispettivo polinomio nella lista che abbiamo scritto come secondo argomento della funzione, ed a questo ci sommiamo il secondo elemento della lista ottenuta come risultato, riotteniamo il polinomio originale:

```
In[101]:= %[[1, 1]] (x - 4) + %[[1, 2]] (x^2 - 5) + %[[2]]

Out[101]= 84 - 42 z + (-4 + x) (21 + 6 x + x^2 - 12 z - 2 x z)

In[102]:= Expand[%]

Out[102]= -3 x + 2 x^2 + x^3 + 6 z - 4 x z - 2 x^2 z
```

Come potete vedere, abbiamo riottenuto il polinomio originale.

Analoghe considerazioni possiamo fare per gli altri, considerando che FactorTerms accetta solo le opzioni Modulus e Trig, mentre FactorSquareFree, oltre a questi due, accetta anche Extension, ma non GaussianIntegers. Comunque credo che raramente andrete ad usare queste opzioni...

- **Altre Manipolazioni**

Questi comandi che abbiamo analizzato, tuttavia, sono specifici per espressioni razionali fratte. In genere, invece, ci sono altri tipi di espressioni, come quelle trigonometriche. Naturalmente *Mathematica* include anche comandi per trattare questo tipo di espressioni, che possono essere considerate equivalenti a quelle per le espressioni fratte:

`TrigExpand[`*expr*`]`	espande i prodotti
`TrigFactor[`*expr*`]`	fattorizza in prodotti di fattori
`TrigReduce[`*expr*`]`	riduce le espressioni che utilizzano angoli multipli
`TrigToExp[`*expr*`]`	converte forme trigonometriche in esponenziali
`ExpToTrig[`*expr*`]`	converte forme esponenziali in trigonomentriche
`FunctionExpand[`*expr*`]`	espande funzioni speciali
`ComplexExpand[`*expr*`]`	effettua espansioni considerando le incognite reali
`PowerExpand[`*expr*`]`	trasforma $(x\,y)^p$ in $x^p\,y^p$, etc.

Sono tutte molto utili ma, ultimamente, ho usato spesso (e la uso come esempio), la funzione ComplexExpand. Permette, una volta che si ha l'espressione, di dividerla in parte reale e parte immaginaria. Risulta particolarmente utile, tanto per fare un esempio, per calcolare le interstezioni con gli assi dei diagrammi di Nyquist. Supponiamo di avere la seguente espressione:

In[103]:=
```
W = (1 + I ω Z) / ((1 + I ω P1) (1 + I ω P2))
```

Out[103]=
$$\frac{1 + i\,Z\,\omega}{(1 + i\,P1\,\omega)\,(1 + i\,P2\,\omega)}$$

Se, adesso, mi piacerebbe per esempio trovare la parte reale di questa espressione, mi verrebbe naturale scrivere:

In[104]:=
```
Re[W]
```

Out[104]=
$$\mathrm{Re}\left[\frac{1 + i\,Z\,\omega}{(1 + i\,P1\,\omega)\,(1 + i\,P2\,\omega)}\right]$$

Tuttavia, come possiamo vedere, non ci dà il risultato sperato. Perchè? Abbiamo già detto che *Mathematica* ha uno dei sui punti di forza nel calcolo simbolico, per cui, dato che non abbiamo definito le varie incognite come P1, P2, per il programma potrebbero essere numeri reali, complessi, funzioni, espressioni e così via. Con il comando ComplexExpand, invece, forziamo *Mathematica* a trattarli come se fossero numeri reali, permettendodi di avere il risultato desiderato:

In[105]:=
```
ComplexExpand[Re[W]]
```

Out[105]=
$$\frac{1}{(1+P1^2\,\omega^2)\,(1+P2^2\,\omega^2)}-\frac{P1\,P2\,\omega^2}{(1+P1^2\,\omega^2)\,(1+P2^2\,\omega^2)}+\frac{P1\,Z\,\omega^2}{(1+P1^2\,\omega^2)\,(1+P2^2\,\omega^2)}+\frac{P2\,Z\,\omega^2}{(1+P1^2\,\omega^2)\,(1+P2^2\,\omega^2)}$$

Vediamo che il risultato è quello desiderato. Ma è troppo lungo, vero? Basta usare la magica funzione, allora:

In[106]:=
```
Simplify[%]
```

Out[106]=
$$\frac{1+P2\,Z\,\omega^2+P1\,(-P2+Z)\,\omega^2}{(1+P1^2\,\omega^2)\,(1+P2^2\,\omega^2)}$$

Automaticamente, possiamo calcolarci la parte immaginaria:

In[107]:=
```
Simplify[ComplexExpand[Im[W]]]
```

Out[107]=
$$-\frac{\omega\,(P1+P2-Z+P1\,P2\,Z\,\omega^2)}{(1+P1^2\,\omega^2)\,(1+P2^2\,\omega^2)}$$

Ah, ora sì che posso fare tranquillamente i grafici delle esercitazioni di Controlli Automatici e di Elettronica!!!

Anche le altre funzioni sono, naturalmente, molto utili, e permettono di semplificare parecchio le espressioni, oppure di scriverle in particolari forme

In[108]:=
```
TrigReduce[Tan[x] Cos[2 x]]
```

Out[108]=
$$-\frac{1}{2}\,\text{Sec}[x]\,(\text{Sin}[x]-\text{Sin}[3\,x])$$

In[109]:=
```
TrigFactor[%]
```

Out[109]=
```
(Cos[x] - Sin[x]) (Cos[x] + Sin[x]) Tan[x]
```

Come possiamo vedere, possiamo facilmente manipolare anche espressioni algebriche. Possiamo anche semplificare i casi in cui compaiono angoli multipli, riducendo opportunamente l'angolo per restituire lo stesso risultato. Inoltre permette anche di poter effettuare trasformazioni che invocano gli argomenti

In[110]:=
```
TrigReduce[Sin[5 x π]^2 + Cos[5 x π]^2]
```

Out[110]=
```
1
```

Con il comando seguente, TrigExpand, possiamo espandere le espressioni trigonometriche utilizzando le regole che ben conosciamo, come quelle di prostaferesi e le altre

```
In[111]:= Clear[a, b, c];

In[112]:= TrigExpand[Cos[a + b + c]]

Out[112]= Cos[a] Cos[b] Cos[c] - Cos[c] Sin[a] Sin[b] -
           Cos[b] Sin[a] Sin[c] - Cos[a] Sin[b] Sin[c]

In[113]:= TrigExpand[Cos[2 x]]

Out[113]= Cos[x]^2 - Sin[x]^2

In[114]:= TrigExpand[Sin[2 x]]

Out[114]= 2 Cos[x] Sin[x]
```

Possiamo anche convertire le espressioni da trigonometriche ad esponenziali e viceversa, tramite i due comandi appositi.

```
In[115]:= tr = Tan[x] Sin[x]

Out[115]= Sin[x] Tan[x]

In[116]:= TrigToExp[tr]

Out[116]= -(e^(-i x) - e^(i x))^2 / (2 (e^(-i x) + e^(i x)))
```

E da questa passare di nuovo alla corrispondente trigonometrica:

```
In[117]:= ExpToTrig[%]

Out[117]= Sin[x] Tan[x]
```

Possiamo anche effettuare semplici conversioni avendo soltanto dati numerici, cosa che comunque risulta essere parecchio utile in molti casi:

```
In[118]:= ExpToTrig[(-2)^(1/13)]

Out[118]= 2^(1/13) Cos[π/13] + i 2^(1/13) Sin[π/13]
```

Un modo potente per poter eseguire le semplificazioni in casi particolari sono le *supposizioni*: permettono di definire delle caratteristiche particolari alle variabili, in modo da rendere univoco il

significato delle espressioni, in modo da poter effettuare le opportune semplificazioni. Prendiamo il seguente esempio, per capirci meglio:

```
In[119]:= a = Sqrt[x^2]
```

$$\text{Out[119]=} \quad \sqrt{x^2}$$

```
In[120]:= Simplify[%]
```

$$\text{Out[120]=} \quad \sqrt{x^2}$$

Vediamo che non viene semplificato in x: questo è normale, perchè il valore x potrebbe essere, sia positivo che negativo, e quindi potrebbe essere semplificato sia in x che in -x. Per evitare queste ambiguità, possiamo definire il le caratteristiche della incognita x, come argomento del comando Simplify:

```
In[121]:= Simplify[a, x > 0]
```

$$\text{Out[121]=} \quad x$$

In questo caso abbiamo detto a *Mathematica* che ci interessa solamente il caso in cui x è un valore positivo: anche se non abbiamo assegnato un valore specifico, la condizione che abbiamo imposto basta a *Mathematica* per non aver dubbi sul tipo di semplificazione da effettuare; potevamo anche considerare l'altro caso, naturalmente, in cui ci interessavano soltanto i valori negativi:

```
In[122]:= Simplify[a, x < 0]
```

$$\text{Out[122]=} \quad -x$$

Possiamo anche identificare il tipo di dominio dell'incognita: per esempio, se deve essere un numero reale oppure intero, usando il comando Element all'interno di Simplify, come nel seguente esempio:

```
In[123]:= Simplify[Sqrt[x^2], Element[x, Reals]]
```

$$\text{Out[123]=} \quad \text{Abs}[x]$$

In questo caso, abbiamo detto a *Mathematica* di semplificare l'espressione considerando x un numero reale. Facciamo un altro esempio:

```
In[124]:= Simplify[Sin[x + 2 n Pi], Element[n, Integers]]
```

$$\text{Out[124]=} \quad \text{Sin}[x]$$

In quest'altro esempio abbiamo detto al programma di considerare n un numero intero. In questo modo abbiamo potuto effetutare la giusta semplificazione, dato che il valore è lo stesso a meno di

valori di n uguali, come sappiamo praticamente da sempre.

Elements può anche essere scritto mediante il suo simbolo corrispondente ∈:

```
In[125]:= Simplify[a, {x ∈ Integers, x > 0}]
Out[125]= x
```

I due comandi citati, Simplify e FullSimplify, tentano sempre di semplificare al massimo l'espressione ottenuta, con varie manipolazioni ed applicando varie regole. Con le assunzioni possiamo anche semplificare i casi, per così dire, ambigui. Tuttavia, a volte in realtà non vogliamo modificare la struttura, e semplificare l'espressione; potremmo, invece, voler soltanto vedere il valore che assume l'espressione quando vengono definiti certe imposizioni, tramite le assunzioni. Per questo compito c'è il comando seguente:

`Refine[`*expr*`, `*assum*`]`	riscrive *expr* usando le assunzioni

Supponiamo di avere la seguente espressione:

```
In[126]:= expr = Cos[x + π y] Log[x];
```

Vediamo di semplificarla:

```
In[127]:= Simplify[expr]
Out[127]= Cos[x + π y] Log[x]

In[128]:= FullSimplify [expr]
Out[128]= Cos[x + π y] Log[x]
```

Come potete vedere, non cambia niente. Usiamo delle assunzioni. Per esempio, ipotizziamo che *y* sia un numero intero:

```
In[129]:= Simplify[expr, y ∈ Integers]
Out[129]= (-1)^y Cos[x] Log[x]

In[130]:= FullSimplify[expr, y ∈ Integers]
Out[130]= (-1)^y Cos[x] Log[x]
```

In[131]:=
```
Refine[expr, y ∈ Integers]
```

Out[131]=
$$(-1)^{y}\,\text{Cos}[x]\,\text{Log}[x]$$

Come possiamo vedere, tutti e tre i comandi danno la stessa definizione, perchè non si può semplificare altrimenti. Imponiamo adesso una condizione sulla *x*:

In[132]:=
```
Simplify[expr, x < 0]
```

Out[132]=
$$\text{Cos}[x+\pi\,y]\,\text{Log}[x]$$

In[133]:=
```
FullSimplify[expr, x < 0]
```

Out[133]=
$$\text{Cos}[x+\pi\,y]\,\text{Log}[x]$$

In[134]:=
```
Refine[expr, x < 0]
```

Out[134]=
$$\text{Cos}[x+\pi\,y]\,(i\,\pi+\text{Log}[-x])$$

Questa volta qualcosa è cambiato; è stato imposto un logaritmo di un numero negativo e, mentre Simplify e FullSimplify semplificavano automaticamente il risultato, con Refine questo viene scritto in maniera esplicita, senza ulteriori semplificazioni.

- **Nota per dimensioni fisiche**

Un aspetto interessante di *Mathematica*, che viene spesso sottovalutato, è il fatto che, per la sua stessa natura di calcolo simbolico, ci permette di poter effettuare operazioni con quantità fisiche mantenendo la consistenza delle unità di misura. Per esempio, se vogliamo scrivere una quantità in metri, possiamo semplicemente scrivere:

In[135]:=
```
a = 13 m;
```

Se vogliamo una velocità, basta scrivere, per esempio:

In[136]:=
```
% / (7 s)
```

Out[136]=
$$\frac{13\ \text{m}}{7\ \text{s}}$$

E possiamo anche effettuare velocemente delle conversioni, per esempio scrivendo:

In[137]:=
```
15 m /. m → 3.2808 feet
```

Out[137]=
```
49.212 feet
```

Si deve notare che questa possibilità non è dovuta al fatto che *Mathematica* implementa un qualche metodo per tener conto delle unità di misura, ma viene come naturale conseguenza del calcolo simbolico; a tutti gli effetti, quelle che per noi sono unità di misura, per *Mathematica* sono solo variabili, e sono trattate come tali. Tuttavia, può essere utile usare questo trucchetto quando si vuole lavorare con le unità di misura, anche se bisogna magari scriversi prima da qualche parte le varie conversioni. Tuttavia, una volta fatto la prima volta, dopo semplifica notevolmente i conti. Consideriamo anche che c'è un pacchetto di *Mathematica* che tratta le unità di misura. I pacchetto sono dei file che contengono nuove definizioni e nuovi comandi per avere funzionalità in più, che *Mathematica* non offre. Con l'installazione standard ce ne sono parecchi, specializzati in calcolo statistico etc, e ce n'è anche uno per le unità di misura, richiamabile con

In[138]:=
```
<< Miscellaneous`Units`
```

Tuttavia, in queste pagine non tratterò dei pacchetti, perchè per apprezzarli bisogna conoscere prima quello che si può avere da *Mathematica*, e usarlo sono in caso che ce ne sia bisogno. Anche in questo caso, leggere la documentazione del programma è sempre la cosa migliore. Vi ho già detto che per noi l'inglese è importante (nel bene e nel male)...