

# Una libreria di funzioni per la geometria analitica

Michele Impedovo

La geometria analitica del piano costituisce uno dei più importanti e consolidati argomenti di matematica.

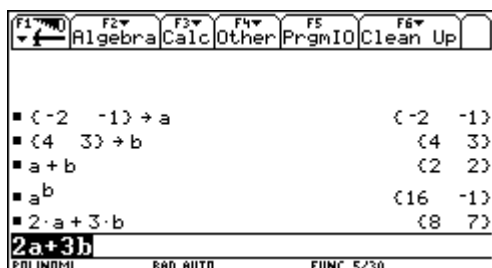
Un lavoro interessante parallelo alla trattazione dei temi centrali (punti, rette, parallelismo e perpendicolarità, distanze, luoghi, coniche) può essere quello di costruire, insieme agli allievi, una **libreria di funzioni** per la geometria analitica del piano. Gli obiettivi sono molteplici: automatizzare i calcoli di routine, consolidare il concetto di algoritmo, svolgere semplici esempi di programmazione, utilizzare in ambienti significativi strutture dati come i vettori e le liste.

Può essere utilizzato qualunque programma; il lavoro proposto utilizza l'ambiente di programmazione della calcolatrice TI-92.

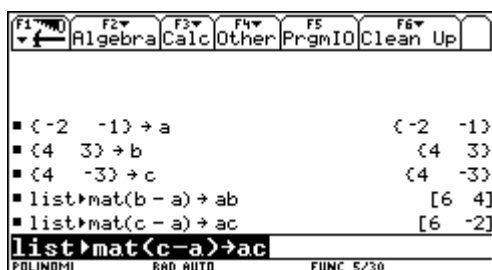
## Punti e vettori

Decidiamo innanzitutto di indicare in modo diverso, anche sintatticamente, **punti** e **vettori**; la TI-92 possiede due strutture dati distinte: le *liste* (con parentesi graffe) e le *matrici* (con parentesi quadre); i vettori non sono altro che matrici ad una sola riga. Rappresenteremo dunque i punti mediante *liste*  $\{x,y\}$ , e i vettori mediante *matrici*  $[x,y]$  ad una sola riga. (Per i vettori utilizzeremo nel seguito la notazione  $\underline{AB}$ , tipograficamente più semplice della notazione  $\overline{AB}$ ).

Le liste sono molto utili e potenti perché una qualunque operazione tra liste (della stessa dimensione) dà come risultato una lista i cui elementi sono il risultato dell'operazione sugli elementi delle due liste.



Ecco per esempio definiti i punti  $A(-2,-1)$ ,  $B(4,3)$ ,  $C(4,-3)$  e i vettori  $\underline{AB}$ ,  $\underline{AC}$ .



I comandi

list▶mat e mat▶list

consentono di trasformare liste in vettori e viceversa.

## La retta

Tradizionalmente la **retta** viene definita algebricamente in forma cartesiana, mediante l'equazione  $ax+by+c=0$ , oppure nella forma  $y=mx+q$ ; decidiamo di utilizzare questa seconda forma, che

fornisce direttamente informazioni geometriche sulla retta. Una retta parallela all'asse  $y$  avrà invece equazione  $x=k$ .

La **pendenza** (o coefficiente angolare)  $p$  di una retta non parallela all'asse  $y$  viene definita come il rapporto  $p = b/a$  di un suo vettore direzione  $[a,b]$ . La retta  $AB$  è dunque il luogo dei punti  $P(x,y)$  tali che

$$\text{pendenza}(AP) = \text{pendenza}(AB)$$

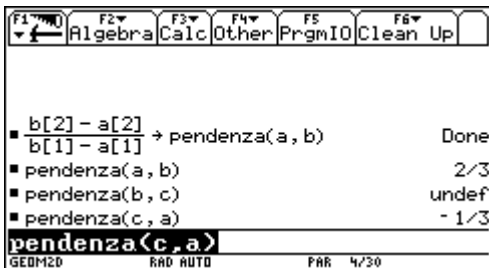
$$\frac{y - y_A}{x - x_A} = \frac{y_B - y_A}{x_B - x_A}$$

$$\frac{y - y_A}{x - x_A} = p,$$

da cui si ottiene l'equazione cartesiana della retta:

$$y = p(x - x_A) + y_A.$$

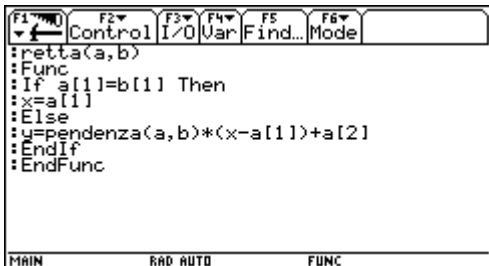
Siamo pronti per costruire la prima funzione: definiamo la pendenza della retta per due punti.



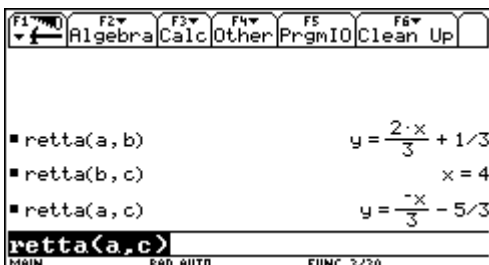
Come si vede, la TI-92 dà in uscita undef per la pendenza di una retta parallela all'asse  $y$ . Un aspetto interessante in fase di programmazione è quello di distinguere con la struttura

IF ... THEN... ELSE

il comportamento delle rette parallele all'asse  $y$ . Il primo programma è il seguente.



Ecco le rette per i punti  $A, B, C$ , dell'esempio precedente.



Una retta può essere data, anziché mediante due punti, mediante un punto e la pendenza. Vediamo allora di generalizzare la funzione precedente, utilizzando il comando

getType

che distingue se una variabile contiene una lista (quindi è un punto) oppure un numero (che è la pendenza). Ecco il programma più generale.

```

F1 Control F2 I/O F3 Var F4 Find... F5 Mode
:retta(a,b)
:Func
:If getType(a)="num" Then
:u=a*(x-b[1])+b[2]
:Elseif getType(b)="num" Then
:u=b*(x-a[1])+a[2]
:Elseif a[1]=b[1] Then
:x=a[1]
:Else
:u=pendenza(a,b)*(x-a[1])+a[2]
:EndIf
:EndFunc
MAIN          RAD AUTO          FUNC

```

```

F1 Algebra F2 Calc F3 Other F4 PrgmIO F5 Clean Up
■ retta(2 5), (2 6)          x = 2
■ retta(2, (2 5))          y = 2·x + 1
■ retta(2 5), 2)          y = 2·x + 1
■ retta(2 5), (3 6))      y = x + 3
■ retta(2,5), (3,6))
MAIN          RAD AUTO          FUNC 4/30

```

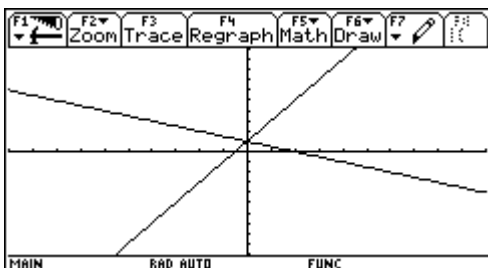
## Angoli

Utilizzando un software come Derive, o una calcolatrice grafica, è di fondamentale importanza una osservazione: mentre il **parallelismo** è un invariante per *dilatazioni* degli assi (cambiamento delle unità di misura), cioè per trasformazioni del tipo  $(x, y) \rightarrow (hx, ky)$ , non lo è la **perpendicolarità**.

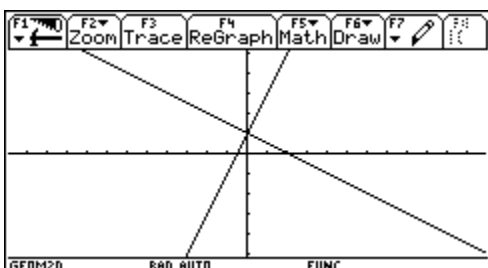
L'immagine seguente mostra per esempio i grafici delle rette

$$y = 2x + 1$$

$$y = -\frac{1}{2}x + 1.$$



Su una calcolatrice grafica ha senso parlare di rette perpendicolari solo se si utilizza uno *zoom monometrico*. La TI-92 dà la possibilità di passare ad un grafico monometrico che conservi tutte le informazioni del grafico corrente.



Attraverso il prodotto scalare tra due vettori  $\mathbf{a}=[x_1, y_1]$ ,  $\mathbf{b}=[x_2, y_2]$ :

$$\mathbf{a} \cdot \mathbf{b} = [x_1, y_1] \cdot [x_2, y_2] = x_1x_2 + y_1y_2$$

è possibile definire nel modo più naturale l'ampiezza di un angolo, come **angolo convesso tra due vettori**: da

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \cdot \|\mathbf{b}\| \cdot \cos\alpha$$

ricaviamo

$$\cos(\alpha) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}, \quad \alpha = \arccos\left(\frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}\right)$$

La TI-92 mette a disposizione il comando dotP per il prodotto scalare di due vettori, e il comando norm per il modulo (la *norma*) di un vettore.

```
F1 Algebra F2 Calc F3 Other F4 PrgmIO F5 Clean Up
▪ dotP([2 1],[1 3]) 5
▪ norm([2 1]) √5
▪ norm([a b]) √a²+b²
norm([a,b])
GEO M2D DEG AUTO PAR 3/30
```

È possibile quindi definire le funzioni ang2v e ang3p che prendono in ingresso rispettivamente due vettori e tre punti, e danno in uscita l'ampiezza (in gradi) del relativo angolo convesso.

```
F1 Control F2 I/O Var F3 Find... F4 Mode
:ang2v(a,b)
:Func
:approx(cos⁻¹(dotP(a,b)/(norm(a)*norm(b)))
:)
:EndFunc
GEO M2D DEG AUTO PAR
```

```
F1 Control F2 I/O Var F3 Find... F4 Mode
:ang3p(a,b,c)
:Func
:ang2v(list▶mat(b-a),list▶mat(c-a))
:EndFunc
MAIN DEG AUTO FUNC
```

```
F1 Algebra F2 Calc F3 Other F4 PrgmIO F5 Clean Up
▪ ang2v([2 1],[1 3]) 45.
▪ ang2v([a b],[ -b a]) 90.
▪ ang2v([a b],[ -a -b]) 180.
▪ ang2v([a b],[a b]) 0.
▪ ang3p(⟨1 1⟩,⟨3 1⟩,⟨4 5⟩) 104.
▪ ang3p(⟨1 1⟩,⟨-3 1⟩,⟨4 5⟩) 29.74
ang3p(⟨1,1⟩,⟨-3,1⟩,⟨4,5⟩)
GEO M2D DEG AUTO PAR 6/30
```

### Altezze, mediane, assi

Possiamo ora definire la funzione altezza, che prende in ingresso tre punti ordinati (i vertici del triangolo) e dà in uscita l'equazione dell'altezza passante per il primo punto e ortogonale al lato degli altri due .

```

F1 F2 F3 F4 F5 F6
Algebra Calc Other PrgmIO Clean Up

■ a (-2 -1)
■ b (4 3)
■ c (4 -3)
■ altezza(a, b, c) y = -1
■ altezza(b, c, a) y = 3·x - 9
■ altezza(c, a, b) y = 3 -  $\frac{3 \cdot x}{2}$ 
altezza(c, a, b)
MAIN RAD AUTO FUNC 6/30

```

Una volta definito il punto medio tra due punti:

```

F1 F2 F3 F4 F5 F6
Algebra Calc Other PrgmIO Clean Up

■  $\frac{a + b}{2} \rightarrow \text{ptomedio}(a, b)$  Done
■ ptomedio(a, b) (1 1)
■ ptomedio(b, c) (4 0)
■ ptomedio(c, a) (1 -2)
ptomedio(c, a)
MAIN RAD AUTO FUNC 4/30

```

è possibile definire la funzione che prende in ingresso tre punti ordinati e fornisce in uscita l'equazione della mediana passante per il primo vertice.

```

F1 F2 F3 F4 F5 F6
Control I/O Var Find... Mode

:mediana(B, b, c)
:Func
:retta(a, ptomedio(b, c))
:EndFunc
MAIN RAD AUTO FUNC

```

```

F1 F2 F3 F4 F5 F6
Algebra Calc Other PrgmIO Clean Up

■ mediana(a, b, c) y =  $\frac{x}{6} - 2/3$ 
■ mediana(b, c, a) y =  $\frac{5 \cdot x}{3} - 11/3$ 
■ mediana(c, a, b) y =  $7/3 - \frac{4 \cdot x}{3}$ 
mediana(c, a, b)
GEOIM2D RAD AUTO PAR 3/30

```

In modo analogo si definisce la funzione che fornisce l'equazione dell'asse di un segmento.

```

F1 F2 F3 F4 F5 F6
Algebra Calc Other PrgmIO Clean Up

■ asse(a, b) y =  $5/2 - \frac{3 \cdot x}{2}$ 
■ asse(b, c) y = 0
■ asse(c, a) y = 3·x - 5
asse(c, a)
MAIN RAD AUTO FUNC 3/30

```

```

F1 F2 F3 F4 F5 F6
Algebra Calc Other PrgmIO Clear a-z...

■ asse(a, b) y =  $-\frac{3 \cdot x}{2} + 5/2$ 
■ asse(b, c) y = 0
■ asse(c, a) y = 3·x - 5
asse(c, a)
GEOIM2D RAD AUTO FUNC 3/40

```

## Intersezione tra due rette

Una funzione utile è quella che calcola il punto di intersezione tra due rette, date in ingresso come equazioni. Per semplificare, supponiamo che le due rette non siano parallele.

```

F1  F2  F3  F4  F5  F6
Control I/O Var Find.. Mode
:inters(r,s)
:Func
:Local xx,yy
:If string(left(r))="x" Then
:r→xx:sln→yy
:Elseif string(left(s))="x" Then
:s→xx:rln→yy
:Else
:solve(r|s,x)→xx:solve(r|xx,y)→yy
:EndIf
:(right(xx),right(yy))
:EndFunc
MAIN          RAD AUTO          FUNC
  
```

```

F1  F2  F3  F4  F5  F6
Algebra Calc Other PrgmIO Clean Up
■ inters(y = 2·x - 1, y = 5)          (3  5)
■ inters(x = 2, y = 2·x - 1)         (2  3)
■ inters(y = 2, x = 3)               (3  2)
■ inters(y = 2·x - 1, x = 2)         (2  3)
■ inters(x = 2, y = 2·x - 1)         (2  3)
■ inters(y = 2, x = 3)               (3  2)
■ inters(y = 2·x + 1, y = -3·x + 1)  (0  1)
inters(y=2*x+1, y=-3*x+1)
MAIN          RAD AUTO          FUNC 2/30
  
```

## Il teorema di Eulero

Possiamo ora determinare il circocentro, l'ortocentro e il baricentro del triangolo  $ABC$ .

```

F1  F2  F3  F4  F5  F6
Algebra Calc Other PrgmIO Clean Up
■ inters(asse(a, b), asse(b, c)) → o
                                         (5/3  0)
■ inters(altezza(a, b, c), altezza(b, c, a))▶
                                         (8/3  -1)
■ inters(media(a, b, c), media(b, c, a))▶
                                         (2  -1/3)
media(a, b, c), media(b, c, a)) → g
GEOIM2D      RAD AUTO      PAR 3/30
  
```

Si può verificare facilmente il teorema di Eulero: il circocentro  $O$ , il baricentro  $G$ , l'ortocentro  $H$  di un triangolo sono allineati, e  $OH = 3OG$ .

```

F1  F2  F3  F4  F5  F6
Algebra Calc Other PrgmIO Clean Up
■ inters(asse(a, b), asse(b, c)) → o
                                         (5/3  0)
■ inters(altezza(a, b, c), altezza(b, c, a))▶
                                         (8/3  -1)
■ inters(media(a, b, c), media(b, c, a))▶
                                         (2  -1/3)
■ g - o
                                         (1/3  -1/3)
■ h - o
                                         (1  -1)
h-o
GEOIM2D      RAD AUTO      PAR 5/30
  
```

## Parabole

Il lavoro può continuare in modo del tutto analogo per le parabole. Per esempio risolviamo il problema di determinare l'equazione della parabola (ad asse parallelo all'asse  $y$ ) per tre punti; si tratta di risolvere un sistema lineare di tre equazioni in tre incognite: i coefficienti  $a$ ,  $b$ ,  $c$  dell'equazione  $y = ax^2 + bx + c$ . Tale sistema è abbastanza laborioso da risolvere con carta e penna; è

bene che lo studente cerchi di cogliere la *struttura* dei calcoli, e quando è possibile, di implementarli in un programma. Nel caso di un sistema lineare si può usare il comando `simult(m,v)`

che prende in ingresso la matrice dei coefficienti e il vettore colonna dei termini noti, e restituisce (se esiste) la soluzione del sistema.

A titolo di esempio mostriamo la risoluzione passo-passo. Innanzitutto si costruisce la matrice dei coefficienti.

```

F1 Algebra F2 Calc F3 Other F4 PrgmIO F5 Clean Up
▪ ⟨0 1⟩ → a           ⟨0 1⟩
▪ ⟨1 -1⟩ → b          ⟨1 -1⟩
▪ ⟨2 0⟩ → c           ⟨2 0⟩
▪  $\begin{bmatrix} a[1]^2 & a[1] & 1 \\ b[1]^2 & b[1] & 1 \\ c[1]^2 & c[1] & 1 \end{bmatrix} \rightarrow \text{mat} :$   $\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 4 & 2 & 1 \end{bmatrix}$ 
▪ b[1],1][c[1]^2,c[1],1]]→mat:
MAIN          RAD AUTO          FUNC 4/30

```

Poi il vettore colonna dei termini noti.

```

F1 Algebra F2 Calc F3 Other F4 PrgmIO F5 Clean Up
▪  $\begin{bmatrix} a[1]^2 & a[1] & 1 \\ b[1]^2 & b[1] & 1 \\ c[1]^2 & c[1] & 1 \end{bmatrix} \rightarrow \text{mat} :$   $\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 4 & 2 & 1 \end{bmatrix}$ 
▪  $\begin{bmatrix} a[2] \\ b[2] \\ c[2] \end{bmatrix} \rightarrow \text{vetcol} :$   $\begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$ 
▪ [[a[2]][b[2]][c[2]]]→vetcol:
MAIN          RAD AUTO          FUNC 5/30

```

L'inversa della matrice per il vettore colonna ci dà il vettore delle soluzioni.

```

F1 Algebra F2 Calc F3 Other F4 PrgmIO F5 Clean Up
▪  $\begin{bmatrix} a[1]^2 & a[1] & 1 \\ b[1]^2 & b[1] & 1 \\ c[1]^2 & c[1] & 1 \end{bmatrix} \rightarrow \text{mat} :$   $\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 4 & 2 & 1 \end{bmatrix}$ 
▪  $\begin{bmatrix} a[2] \\ b[2] \\ c[2] \end{bmatrix} \rightarrow \text{vetcol} :$   $\begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$ 
▪  $\text{mat}^{-1} \cdot \text{vetcol}$   $\begin{bmatrix} 3/2 \\ -7/2 \\ 1 \end{bmatrix}$ 
▪ mat^-1*vetcol
MAIN          RAD AUTO          FUNC 6/30

```

Trasformiamo il vettore in lista e usiamo il potente comando `polyeval`, che costruisce un polinomio a partire dalla lista dei coefficienti.

```

F1 Algebra F2 Calc F3 Other F4 PrgmIO F5 Clean Up
▪  $\text{mat} \rightarrow \text{list} \left( \begin{bmatrix} 3/2 \\ -7/2 \\ 1 \end{bmatrix} \right)$   $\langle 3/2 \ -7/2 \ 1 \rangle$ 
▪  $y = \text{polyEval}(\langle 3/2 \ -7/2 \ 1 \rangle, x)$ 
 $y = \frac{3 \cdot x^2}{2} - \frac{7 \cdot x}{2} + 1$ 
▪ y=polyeval<<3/2,-7/2,1>,x>
MAIN          RAD AUTO          FUNC 8/30

```

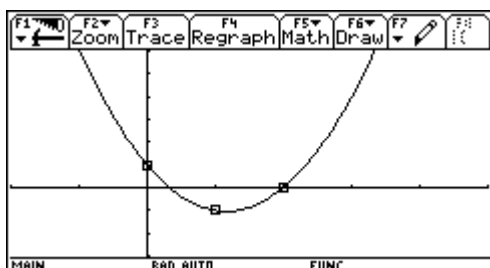
L'intero processo può essere automatizzato da un programma.

```

F1 Control F2 I/O F3 Var F4 Find... F5 Mode F6
:parabola(a,b,c)
:Func
:Local mat,vetcol,coef
:{{a[1]^2,a[1],1}}{b[1]^2,b[1],1}{c[1]^2,
c[1],1}}→mat
:{{a[2]}{b[2]}{c[2]}}→vetcol
:mat→list(simult(mat,vetcol))→coef
:q=polyEval(coef,x)
:EndFunc

```

MAIN      END AUTO      FUNC



L'attività di programmazione può proseguire a diversi livelli di approfondimento.