# Manuale del pacchetto `liLaTeXtikz`

Federico Miceli

estate 2024

**Sommario**

In questo manuale viene descritto il pacchetto grafico di `liLaTeX`, chiamato `liLaTeXtikz`. Si tratta di un pacchetto LaTeX che estende `tikz`. Il manuale contiene quindi anche alcune indicazioni per l'uso di tikz.

Questo manuale conterrà alcuni esempi minimali, in genere a conclusione di ciascuna sezione. Per esempi più estesi e articolati si rimanda alla cartella denominata *esempi_immagini*.

# Indice

# 1 Il pacchetto tikz di LaTeX

## 1.1 Creare coordinate e nodi

### 1.1.1 Il comando `\coordinate`

Per creare dei punti in tikz ci sono due comandi principali, con le seguenti sintassi:

```
\coordinate (nome) at (x,y);
```

crea una coordinata di nome nome nella posizione indicata.

La posizione può essere indicata con coordinate cartesiane $(x, y)$, oppure con coordinate polari $(\vartheta : r)$.

### 1.1.2 Il comando `\node`

In alternativa, possiamo creare il punto come

```
\node[opzioni] (nome) at (x,y) {label};
```

Il comando `\node` funziona in modo analogo a `\coordinate`, ma aggiunge un label nella posizione indicata (dove si trova il punto).

Fra le `opzioni` è possibile specificare alcune caratteristiche da dare al label. La più comune è il colore. Ad esempio, se vogliamo creare un nodo chiamato `P` in posizione $(3, 2)$, in cui vogliamo scrivere la lettera $A$ in rosso, scriveremo

```
\node[red] (P) at (3,2) {$A$};
```

### 1.1.3 Le opzioni `-|` e `|-`

Se abbiamo due punti $A = (x_A, y_A)$ e $B = (x_B, y_B)$, la scrittura $A| - B$ crea il punto di coordinate $(x_A, y_B)$. La scrittura $A - |B$ crea invece il punto di coordinate $(x_B, y_A)$.

### 1.1.4 Operazioni algebriche col pacchetto `calc`

`calc` è un pacchetto che ci permette di svolgere semplici operazioni algebriche con le coordinate dei punti. Se ad esempio abbiamo definito due coordinate chiamate `A` e `B`, possiamo definire una nuova coordinata `C` in posizione `A+B` scrivendo

```
\coordinate (C) at ($(A)+(B)$);
```

Le coordinate possono anche essere moltiplicate per dei numeri. Ad esempio se `A` ha coordinate $(1, 3)$ e `B` ha coordinate $(2, -1)$, allora il punto `C`, definito come

```
\coordinate (C) at ($2*(A)-0.5*(B)$);
```

avrà coordinate $(1, 6.5)$.

Non sembrano esserci problemi a sommare coordinate cartesiane con coordinate polari.

## 1.2 Disegnare con il comando `\draw`

Il comando `\draw` è il comando di `tikz` utilizzato per disegnare semplici linee.

Per disegnare una linea spezzata $A - B - C$ utilizziamo la sintassi

```
\draw[opzioni] (A)--(B)--(C);
```

Se vogliamo disegnare una linea verticale che parte da $A$ e sale di 1 (verso $y$) possiamo invece usare la scrittura

```
\draw[opzioni] (A)--++(0,1);
```

Questo tipo di operazione può essere continuata, disegnando via via nuovi pezzi.

### 1.2.1 Il comando `\path`

Il comando `\path` è identico al comando `\draw`, con la differenza che il tratto non viene realmente disegnato. Questo può essere utile se ci interessa un tratto (ad esempio una circonferenza) per definire altri punti, ma non vogliamo che il tratto venga disegnato.

### 1.2.2 Piegare un tratto con `bend`

Normalmente il tratto disegnato è un segmento. È però possibile *piegare* il segmento, trasformandolo in un arco di circonferenza. La scrittura

```
\draw[opzioni] (A)to[bend left=30](B);
```

Questo disegna un arco di circonferenza da $A$ a $B$ che passa a sinistra del segmento $AB$ (percorrendolo da $A$ a $B$) e formando angoli di 30° (fra segmento e tangente all'arco di circonferenza).

ovviamente è anche possibile scegliere `band right`.

### 1.2.3 Dare un nome a un punto in un `\draw`

Possiamo anche dare un nome a un punto creato facendo un disegno. Ad esempio il seguente comando

`\draw (2,4)--(5,3)node(P){}--(7,4);`

disegna una linea spezzata da $(2, 4)$ a $(5, 3)$ e poi a $(7, 4)$. Inoltre crea un punto chiamato $P$ in posizione $(5, 3)$.

### 1.2.4 Disegnare archi di circonferenze ed ellissi con l'opzione `arc`

Possiamo anche disegnare un arco di circonferenza o di ellisse. Per farlo utilizziamo la scrittura

```
\draw[opzioni] (A) arc (ϑ_A : ϑ_B : r);
```

Questo disegna un arco di circonferenza di raggio $r$ che parte dal punto $A$. Nel disegnare l'arco si suppone che $A$ si trovi all'angolo $\vartheta_A$ della circonferenza, e che si voglia disegnare l'arco fino all'angolo $\vartheta_B$.

Per disegnare un arco di ellisse (di semiassi $a$ e $b$) è possibile utilizzare la notazione analoga

```
\draw[opzioni] (A) arc (ϑ_A : ϑ_B : a cm and b cm);
```

### 1.2.5 Disegnare rettangoli, circonferenze ed ellissi con `\draw`

Se vogliamo disegnare un rettangolo $ABCD$ possiamo scrivere

```
\draw[opzioni] (A)rectangle(C);
```

(basta scegliere due vertici opposti).

In alternativa, se il vettore $\overrightarrow{AC} = (5, 3)$ (quindi il rettangolo ha base 5 e altezza 3, e $A$ è il vertice in basso a sinistra), possiamo usare la scrittura

`\draw[opzioni] (A)rectangle++(5,3);`

Possiamo inoltre disegnare una circonferenza di centro $C$ e centro $r$ come

```
\draw[opzioni] (C)circle(r);
```

Analogamente, possiamo disegnare un'ellisse di centro $C$ e semiassi $a$ e $b$ con la sintassi

```
\draw[opzioni] (C)ellipse(a cm and b cm);
```

### 1.2.6 Le opzioni per `\draw`

Il comando `\draw` accetta nella sua sintassi alcune `opzioni`. Queste sono le più diffuse:

- possiamo specificare un colore (come `red`, `blue`, `Green`, `green`, `cyan`, `yellow`, `orange`, `violet`, `magenta`, `teal`, `pink`, `brown`, `gray`, `black`, `white`). Questo indicherà il colore delle linee;
- se desideriamo colorare l'interno della figura disegnata, ciò può essere specificato con l'opzione fill. Ad esempio, se vogliamo colorare l'interno del poligono disegnato di rosso, ciò può essere fatto specificando, fra le `opzioni`, la stringa fill=blue. Più comunemente vogliamo fare un blu più "trasparente". Possiamo ottenerlo come fill=blue!50, o come fill=blue!30 (se lo vogliamo ancora più trasparente);
- lo spessore del tratto può essere specificato cone width. Possiamo dare opzioni come thin, thick, very thick o ultra thick. Se vogliamo un tratto più spesso di ultra thick (o più sottile di ultra thin), ciò può essere ottenuto come line width=0.25mm (specificando lo spessore desiderato);

- possiamo definire il tipo di tratto (il pattern). Ad esempio possiamo chiedere una linea tratteggiata come dashed, oppure puntinata con dotted;
- possiamo specificare una freccia (un'arrow), come ->, <-, <-> o |<->|;
- possiamo dare un nome al bordo del disegno specificando name path=cammello (in questo caso il tratto si chiama cammello). Questo può essere utile per alcuni comandi, in particolare per trovare l'intersezione fra diversi commini usando il comando `\intpaths` trattato nella sezione 10.

## 1.3 L'opzione `\foreach`

Il pacchetto `tikz` supporta anche i loop `foreach`. Per farlo, utilizziamo la seguente sintassi

```
\foreach \i in {0,1,...,10} {body}
```

In questo caso il `body` può contenere anche più righe di codice, che possono anche usare il valore dell'iteratore `\i`. Può inoltre contenere a sua volta un `\foreach`, il che ci permette di creare nested loops.

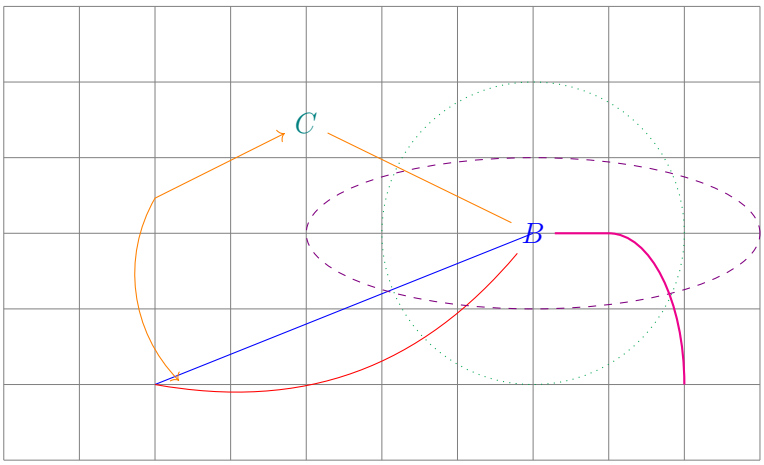
Il range `{0,1,...,10}` deve contenere i due estremi (in questo caso 0 e 10, che sono entrambi inclusi) e il secondo termine della successione (in questo caso 2), necessario per capire di quanto incrementare da un'iterazione alla successiva. Ad esempio, se specificassimo il range `{0,2,...,10}`, allora `\i` assumerebbe solo i valori pari.

Il range può anche essere fornito completamente a mano, per elencazione (se è "strano", e può anche essere costituito da una lista di punti anziché una lista di numeri).

## 1.4 Esempio di uso del pacchetto `tikz` di base

Il codice seguente produce l'immagine allegata:

```
\begin{figure}[!ht] \centering
\begin{tikzpicture}
\foreach \x in {0,1,...,10} { \draw[gray,very thin] (\x,0) --++ (0,6); }
\foreach \y in {0,1,...,6} { \draw[gray,very thin] (0,\y) --++ (10,0); }
\coordinate (A) at (2,1);
\draw[blue] (A) --++ (5,2) node(B){$B$};
\draw[red] (A) to[bend right=30] (B);
\draw[Green,dotted] (B) circle (2);
\draw[violet,dashed] (B) ellipse (3cm and 1cm);
\node[teal] (C) at ($(A)+(60:4)$) {$C$};
\draw[orange,<->] (B) --(C) --++ (-2,-1) arc(150:225:2);
\draw[magenta,thick] (B) --++ (1,0) arc(90:0:  1cm and 2cm);
\end{tikzpicture}
\end{figure}
```



# 2 Introduzione al pacchetto liLaTeXtikz

Il pacchetto liLaTeXtikz estende il pacchetto tikz con una serie di comandi. Questi sono utili soprattutto per produrre disegni di geometria euclidea ed analitica, ma in piccola parte anche per produrre immagini utili nell'ambito della fisica.

Molte delle funzioni implementate si appoggiano al pacchetto tkz-euclide, che contiene numerosi comandi per la produzione di immagini utili per la geometria euclidea. Il pacchetto tkz-euclide in realtà è molto più ampio del pacchetto liLaTeXtikz, e contiene numerose funzioni che non sono direttamente implementate in liLaTeXtikz. Tuttavia la sintassi usata in tkz-euclide a mio avviso risulta piuttosto controintuitiva e poco concisa, il che rende liLaTeXtikz preferibile per la produzione di immagini semplici. tkz-euclide risulta invece preferibile per produrre immagini particolarmente elaborate.

# 3 L'ambiente immagine

In tikz le immagini vengono normalmente racchiuse nell'ambiente tikzpicture. Questo vuol dire che un'immagine viene racchiusa fra una stringa \begin{tikzpicture}[opzioni] e una stringa \end{tikzpicture}

In liLaTeXtikz utilizziamo invece l'ambiente immagine. Un'immagine viene quindi racchiusa fra una stringa \begin{immagine}[opzioni] e una stringa \end{immagine}. La sintassi è quindi

```
\begin{immagine}[opzioni] body \end{immagine}
```

Le opzioni possono specificare il colore o la scala. Ad esempio, scrivendo
\begin{immagine}[red,scale=0.5] body \end{immagine}
produrremo un'immagine in cui il colore di default è il rosso (anziché il nero) e in cui tutte le misure sono dimezzate.

## 3.1 Gli ambienti immaginecap e immaginecap*

Gli ambienti immaginecap e immaginecap* funzionano in modo analogo all'ambiente immagine e vanno usati quando intendiamo inserire una caption all'immagine. La sintassi (analoga per i due ambienti) è

```
\begin{immaginecap}[opzioni]{caption} body \end{immaginecap}
```

La differenza è che immaginecap numera le caption, mentre immaginecap* non include la numerazione delle caption.

# 4 L'ambiente rotazione

Un altro importante ambiente è l'ambiente rotazione, definito tramite la seguente sintassi

```
\begin{rotazione}[centro := (0,0)]{angolo} body \end{rotazione}
```

L'angolo indica l'angolo di rotazione (in gradi in senso antiorario), mentre il centro indica il centro di rotazione (se non specificato si assume una rotazione attorno all'origine).

**Attenzione!** se all'interno del body si inserisce una coordinata come $(2,1)$ questa verrà regolarmente ruotata. Se però si crea un punto $A = (2,1)$ fuori dall'ambiente rotazione e si usa $A$ dentro all'ambiente rotazione questo non verrà ruotato. Il comportamento (controintuitivo) è illustrato nel seguente esempio:

```
\node (O) at (2,0) {$O$};
\coordinate (A) at (3,0.5);
\begin{rotazione}[O]{+90}
  \node[red] (Al) at (A) {$A$};
  \node[blue] (Bl) at (3,0.5) {$B$};
\end{rotazione}
```

# 5 L'uso delle keys in liLaTeXtikz

Il pacchetto liLaTeXtikz contiene diversi comandi. Alcuni di questi comandi hanno delle opzioni come argomenti opzionali. Questi tendenzialmente sono gli stessi previsti per \draw e illustrati nella sezione 1.2.6.

Altri comandi come argomento opzionale accettano invece delle keys.

Per le keys è prevista una sintassi un po' più articolata. Ad esempio, per disegnare un segmento con \draw (che usa delle opzioni) scriviamo \draw[red]...;. Per disegnare un segmento rosso col comando \segmento (che usa delle keys) scriviamo invece \segmento[col=red]...;. Così facendo si sta dando il valore red alla chiave chiamata col (che nel caso di \comando rappresenta il colore del segmento).

## 5.1 Chiavi comuni e loro valori predefiniti

Questo è un elenco delle keys più comuni e dei valori di default che contengono di solito:

- col indica il colore dell'oggetto da disegnare, e di un suo eventuale label. Se non specificato è in genere inizializzato come il colore dell'immagine (generalmente black).
- col draw in alcuni casi specifica il colore delle linee. Se non specificato copia col.
- col lbl e il suo sinonimo lbl col indicano il colore del label. Se non specificato copia il valore in col.
- col fill il colore del filling. Se non specificato copia il valore in col.
- opacity se specificato col fill (se lasciato vuoto) copia col!opacity anziché col.
- fill := false può essere settato a true se vogliamo colorare l'interno della figura. Diventa automaticamente true se specifichiamo col fill.
- width lo spessore della linea. Se non specificato la linea viene disegnata di spessore normale (ovvero semithick). Accetta valori come thin e thick, ma anche very thick e ultra thick (o very thin e ultra thin).
- pattern è il pattern della linea. Il valore di default è la linea continua (ovvero solid) ma può assumere anche valori come dashed o dotted.
- arrow permette di specificare se vogliamo che la linea si concluda con una freccia. In genere il default è vuoto, ma si possono specificare frecce come <-, ->, <->, |<->| o |->.
- lbl il nome da dare al label. Se non specificato in genere non viene creato un label.
- lbl style può cambiare lo stile con cui viene scritto il label. In genere è vuoto, ma può assumere i valori rm, bf o cal. In questi casi il label sarà scritto, rispettivamente, come \mathrm, \mathbf, \mathcal). Se si specifica lbl style non si deve scrivere il lbl in math mode (non si devono scrivere i $).
- lbl size vuoto di default. Se specificato (con valori come large, small o tiny) cambia la dimensione del label.
- lbl dist := 0.3 la distanza del label dal nodo.
- lbl name il nome da dare alla posizione del label. Il valore di default cambia di funzione in funzione.
- mark := none il segno da mettere su un segmento / angolo / arco. Le opzioni sono none (se non vogliamo nessun segno) oppure: s, z, x, o, oo, |, ||, ||| (lista esaustiva).
- mark size := 0.2 la dimensione del segno definito con mark.
- col mark il colore del segno (se non specificato diventa black).
- side := -> può assumere anche il valore <- e indica come deve essere orientata l'immagine (utile per alcune immagini "complesse").

## 5.2 Il key-manager in liLaTeXtikz

Il key-manager di liLaTeXtikz è una parte interna a liLaTeXtikz di cui l'utente normale non deve preoccuparsi, ma che risulta importante per chi intende modificare/estendere il pacchetto.

Contiene alcune funzioni importanti che vengono elencate qui di seguito:

```
\ifkeyempty{KEY}{TRUE}{FALSE};
```

controlla se la key KEY è vuota. Se è vuota esegue il comando TRUE, altrimenti esegue il comando FALSE.

```
\ifkeyequal{KEY}{str}{TRUE}{FALSE};
```

controlla se la key KEY è uguale alla stringa str. Se è uguale esegue il comando TRUE, altrimenti esegue il comando FALSE.

Il comando

```
\setkeyvalue{KEY}{val};
```

assegna il valore val alla key KEY, mentre il comando

```
\copykeyvalue{KEYto}{KEYfr};
```

copia il valore della key KEYfr nella key KEYto.

I comandi `\setkeyvalueifempty{KEY}{val};` e `\copykeyvalueifempty{KEYto}{KEYfr};` funzionano in modi analoghi, ma si attivano solo se la key da sovrascrivere (quella nel primo input) è vuota.

Ci sono infine i comandi

```
\setkeyvaluesifempty{KEY1}{val1}{KEY2}{val2}{KEY3}{val3}...;
```

```
\copykeyvaluesifempty{KEYto1}{KEYfr1}{KEYto2}{KEYfr2}{KEYto3}{KEYfr3}...;
```

che ripetono rispettivamente `\setkeyvalueifempty` e `\copykeyvalueifempty` per tutte le coppie specificate.

## 5.3 Le funzioni PRIV

All'interno del pacchetto liLaTeXtikz sono presenti alcune funzioni il cui nome inizia con la stringa PRIV. Queste sono funzioni *private* e non devono essere utilizzate dall'utente. Il loro scopo è di essere utilizzate internamente da altre funzioni all'interno del pacchetto liLaTeXtikz.

Le keys di queste funzioni hanno dei nomi un po' particolari (che iniziano, a loro volta, con la stringa PRIV), in modo da non poter essere sovrascritte accidentalmente dall'utente quando vengono specificati i valori di keys di funzioni pubbliche (quelle non private).

L'utente medio non deve ignorare queste funzioni, che risultano invece molto importanti per chi desidera modificare/estendere il pacchetto liLaTeXtikz.

# 6 La funzione \griglia

A volte siamo interessati a disegnare una griglia (una parte di foglio a quadretti). Per farlo utilizziamo il comando

```
\griglia[keys]{Ovest}{Est}{Sud}{Nord};
```

Dove gli argomenti `Ovest`, `Est`, `Sud` e `Nord` indicano i confini della griglia.

Se dobbiamo porre `Est` e `Sud` entrambi a zero, possiamo usare anche la seguente sintassi equivalente:

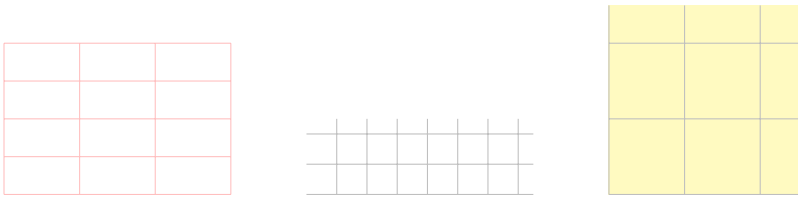
```
\griglia0[keys]{Est}{Nord};
```

Le `keys` sono:

- `col := gray!50`, `width := thin`, `pattern` come spiegate nella sezione 5.1;
- `col fill`. Se specificato l'interno delle griglia viene colorato col colore specificato;
- `s = step := 1` specificano la distanza fra due linee. Volendo possono essere definite separatamente per le due direzioni come `s x = step x` e `s y = step y`.

Vediamo un esempio:

```
\griglia0[s y=0.5,col=pink]{3}{2};
\griglia[s=0.4]{4}{7}{0}{1};
\griglia[col fill=yellow!30]{8}{10.5}{0}{2.5};
```



## 6.1 La carta millimetrata

In fisica siamo spesso interessati a riprodurre parti di carta millimetrata. Ciò può essere fatto utilizzando il comando

```
\cartamillimetrata[keys]{Ovest}{Est}{Sud}{Nord};
```

dove `Ovest`, `Est`, `Sud` e `Nord` hanno gli stessi significati appena spiegati per la funzione `\griglia`.

In questo caso le `keys`sono:

- `col1 := pink!50`, `width1 := ultra thin`, `pattern1` e `s1 = step1 := 0.1` analoghi ai loro analoghi in `\griglia` (per il tratto più sottile).
- `col2 := gray!70`, `width2` := `thin`, `pattern2` e `s2 = step2 := 0.5` analoghi per il tratto più spesso.

C'è anche la possibilità di creare un'intera pagina di carta millimetrata usando il comando

```
\cartamillimetratapagina[keys];
```

con le stesse `keys`.

**Attenzione!** Il comando `\cartamillimetratapagina` funziona solo se `\TipoDoc = immagine`. Per inserire una pagina di carta millimetrata all'interno di un altro documento conviene quindi generare la pagina di carta millimetrata in un altro documento, e poi includere la pagina in questione all'interno del documento reale usando un `\includepdf`.

# 7 Gli assi cartesiani

Spesso è utile disegnare sistemi di riferimento cartesiani. In questa sezione del manuale vedremo alcuni comandi utili per disegnare assi cartesiani di vari tipi.

## 7.1 Gli assi del piano cartesiano

Se disegniamo un piano cartesiano siamo interessati anche a disegnare gli assi cartesiani.

Gli assi cartesiani possono essere disegnati uno per volta usando i comandi

```
\assex[keys]{Ovest}{Est};
```

```
\assey[keys]{Sud}{Nord};
```

con le keys:

- col, width, pattern, arrow := ->, lbl, lbl style, lbl name, lbl size, lbl dist, lbl col = col lbl coi significati della sezione 5.1.
- y := 0 (solo per \assex) specifica la coordinata $y$ dell'asse delle ascisse. Analogamente x := 0 (solo per \assey) specifica la coordinata $x$ dell'asse delle ordinate.
- flip lbl := false se reso true il label viene ruotato di 180°.
- ang shift := 0 una rotazione extra per il label (oltre a quella prevista dalla posizione iniziale).

Volendo è anche possibile definire i due assi cartesiani insieme col comando

```
\assexy[keys]{Ovest}{Est}{Sud}{Nord};
```

o equivalentemente (se vogliamo traslare il centro) come

```
\assexyO[keys]{centro}{Ovest}{Est}{Sud}{Nord};
```

Entrambi i comandi hanno le seguenti keys:

- col, width, pattern, arrow := ->, lbl dist := 0.3, lbl col = col lbl coi significati della sezione 5.1.
- lbl x := $x$ e lbl y := $y$ sono i due label.
- lbl x name e lbl y name sovrascrivono sono i nomi dei due label.
- lbl x style e lbl y style sovrascrivono lbl style per i due diversi label.
- x := 0 e y := 0 (solo per \assexy) specifica le coordinate del centro del sistema di riferimento.

### 7.1.1 Le barre per gli assi cartesiani

Spesso vogliamo aggiungere delle barre per indicare alcuni valori numerici sugli assi cartesiani. Questi in genere vengono usati all'interno di un loop \foreach. Questo può essere fatto coi comandi:

```
\assexbar[keys]{\x}{lbl};
```

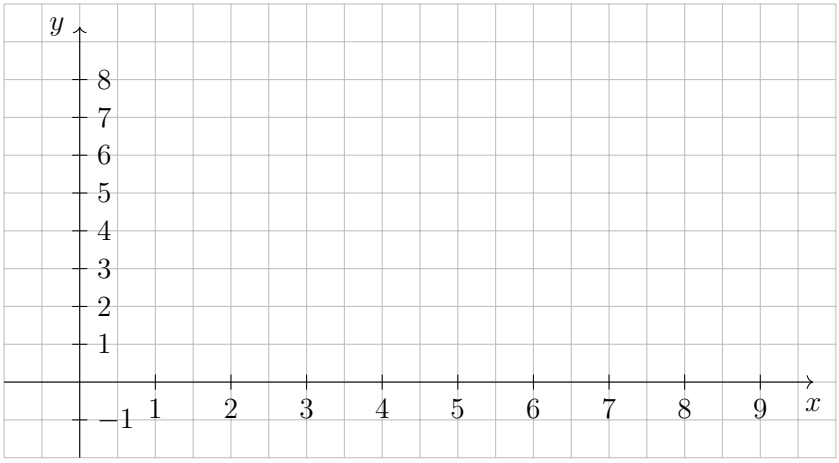
```
\asseybar[keys]{\y}{lbl};
```

Le coordinate \x e \y nei due comandi indicano le posizioni della tacca, mentre lbl indica il label che vogliamo scrivere. Le keys sono:

- col, width := very thin, pattern, arrow, lbl col = col lbl coi significati della sezione 5.1.
- l = length := 0.2 la lunghezza della barretta.
- \flip := true se posto uguale a true pone i label verso l'alto (nel caso di \assexbar) o verso destra (nel caso di \asseybar).
- y := 0 (solo per \assexbar) e x := 0 (solo per \asseybar) è l'altra coordinata del centro della barretta.

Il seguente esempio illustra come utilizzare il sistema cartesiano in un caso semplice.

```
\griglia[s=0.5]{-1}{10}{-1}{5};
\assixy{-1}{9.7}{-1}{4.7};
\foreach \x in {1,2,...,9} \assexbar{\x}{$\x$};
\foreach \y in {-1,1,2,...,8} \asseybar[flip=true]{0.5*\y}{$\y$};
```




## 7.2 Sistemi di riferimento in fisica col comando `\sistrif`

Il comando `\sistrif` consente di inserire un sistema di riferimento (eventualmente ruotato) all'interno di un'immagine di fisica. Lo si usa con la sintassi seguente:

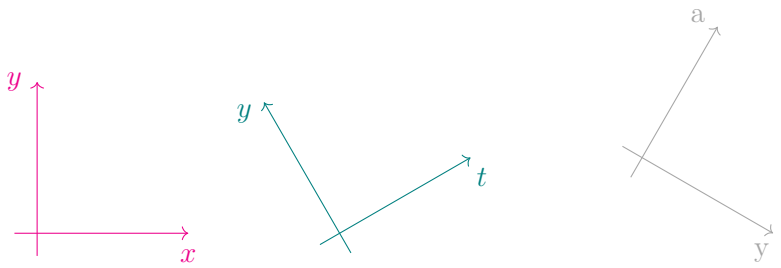
```
\sistrif[keys]{centro}{angolo};
```

dove `centro` indica il centro del sistema di riferimento (dove vogliamo disegnare l'origine), mentre `angolo` indica l'angolo di inclinazione dell'asse $x$. Le `keys` sono:

- `col := magenta`, `lbl col = col lbl`, `width`, `pattern`, `arrow := ->`, `lbl dist := 0.3`, `lbl style` come spiegate nella sezione 5.1. Si riferiscono a entrambi gli assi cartesiani.
- `front := 2` e `back := 0.3` indicano le lunghezze dei due assi cartesiani in avanti (per valori positivi) e indietro (per valori negativi). Possono essere sovrascritti separatamente per i due assi con le `keys` `front x`, `front y`, `back x` e `back y`.
- `lbl x := $x$` e `lbl y := $y$` sono i label per i due assi cartesiani. Se si vuole togliere un label scrivere, ad esempio, `lbl x=`.
- `label := true`. Se reso `false` vengono rimossi entrambi i label.
- `lbl style` può essere sovrascritto con `lbl x style` e `lbl y style` per specificarlo separatamente per le due keys.
- `filp := false` è da rendere uguale a `true` se vogliamo che l'asse $y$ si trovi ruotato di 90° in senso *orario* (anziché *antiorario*) rispetto all'asse $y$.

Vediamo un semplice esempio:

```
\sistrif{0,0}{0};
\sistrif[col=teal,lbl x=$t$]{4,0}{30};
\sistrif[col=gray!70,lbl x=a,lbl style=rm,flip=true]{8,1}{60};
```



# 8 I punti con \punto

La funzione principale di liLaTeXtikz è la funzione che crea dei punti usando la sintassi

```
\punto[keys]{coordinate};
```

con le keys:

- col, lbl col = col lbl, lbl, lbl style, lbl name, lbl size, lbl dist := 0.3 come spiegate nella sezione 5.1.
- lbl ang := -90 l'angolo a cui si trova il label (se specificato).
- name := puntoNome il nome da dare al punto (non al label).
- size := 1.5 la dimensione del punto (in pt).
- shape := circle la forma del punto. Può essere posto a rectangle.

## 8.1 I punti con sintassi concisa come \pnt, \pntlbl e \pntl

Spesso vogliamo specificare solo alcune delle keys di \punto. Per farlo ci sono i seguenti comandi più concisi, che internamente richiamano \punto. Vogliamo invece chiamare esplicitamente \punto se vogliamo personalizzare altri parametri:

```
\pnt[col]{name}{coordinate};
```

utile se non vogliamo inserire un label.

Se invece vogliamo un label, la versione più generale è

```
\pntlbl[col]{name}{lbl}{lbl ang}{coordinate};
```

Ci sono infine versioni di \pntlbl ancora più concise, in cui il label coincide col nome (circa):

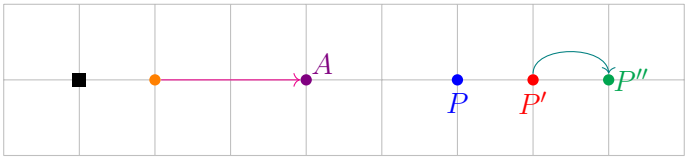
```
\pntl[col]{name}{lbl ang}{coordinate};      % lbl := $name$
```

```
\pntlp[col]{nome}{lbl ang}{coordinate};      % name := nomep, lbl := $nome'$
```

```
\pntls[col]{nome}{lbl ang}{coordinate};      % name := nomes, lbl := $nome"$
```

Vediamo un semplice esempio illustrativo.

```
\grigliaO{9}{2};
\punto[shape=rectangle,size=2.5]{1,1};
\pnt[orange]{T}{2,1};
\pntlbl[violet]{A}{$A$}{45}{4,1};
\draw[->,magenta] (T)--(A);
\pntl[blue]{P}{-90}{6,1};
\pntlp[red]{P}{-90}{7,1}; % il punto si chiama Pp
\pntls[Green]{P}{0}{8,1}; % il punto si chiama Ps
\draw[->,teal] (Pp) to[bend left=90] (Ps);
```




**Attenzione!** I punti definiti con \punto e i suoi derivati non sono compatibili con la chiave fill di \draw. Se vogliamo usare la chiave fill serve quindi definire i punti tramite \coordinate, e poi ripassarli in un secondo momento (dopo aver usato la chiave fill dove ci serve).

# 9 L'aggiunta dei label con \stylelbl

Aggiungere label può essere molto scomodo. Soprattutto se vogliamo scrivere il label usando un ambiente matematico (\mathrm, \mathbf o \mathcal). Per fare ciò, nel pacchetto liLaTeXtikz è definito un comando specifico:

```
\stylelbl[keys]{posizione};
```

dove coord è la posizione del punto dove creare il label. Le keys sono:

- lbl, lbl size, lbl name := scriptlblName, col lbl = lbl col = col come spiegate nella sezione 5.1.
- lbl style è il label style da usare. Può essere lasciato vuoto se non vogliamo nessuno stile, o può essere posto ai seguenti valori:
  - se lbl style := rm il label viene scritto come \mathrm;
  - se lbl style := bf il label viene scritto come \mathbf;
  - se lbl style := cal il label viene scritto come \mathcal.

**Attenzione!** se il lbl style viene specificato (non viene lasciato vuoto), allora il lbl non può essere scritto in ambiente matematico (che verrà aggiunto automaticamente dall'ambiente \stylelbl). Le uniche eccezioni sono se lbl := $x$ e lbl := $y$ (per cui \stylelbl rimuove i simboli di $).

Quindi un comando come
`\stylelbl[lbl=$a$,lbl style=rm]{0,0};`
genera un errore. Possiamo invece usare impunemente un comando come
`\stylelbl[lbl=a,lbl style=rm]{0,0};`

**Attenzione!** Per chi vuole modificare/estendere il pacchetto liLaTeXtikz non bisogna usare la funzione \stylelbl, ma la sua versione privata \PRIVstylelbl (vedi la sezione 5.3 per una spiegazione più dettagliata sulle funzioni private).

# 10 Intersezioni con \intrette e \intpaths

Il comando \interette ci permette di creare un punto $P$ come l'intersezione fra due rette $AB$ e $CD$. La sintassi è

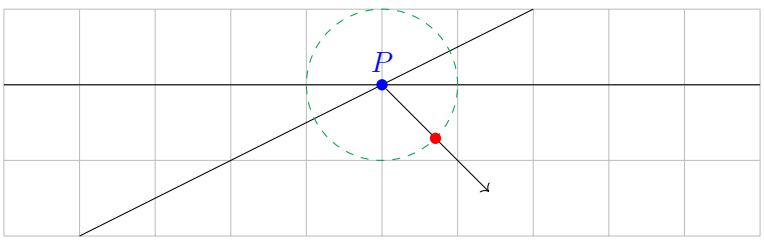
```
\intrette{P}{A}{B}{C}{D};
```

Possiamo anche trovare un punto $P$ di intersezione fra due paths chiamati path1 e path2 usando la sintassi

```
\intpaths{P}{path1}{path2};
```

I due comandi sono illustrati nel seguente esempio:

```
\grigliaO{10}{3};
\draw(1,0)--(7,3) (0,2)--++(10,0);
\intrette{P}{1,0}{3,1}{0,2}{1,2};
\pntl[blue]{P}{90}{P};
\draw[Green,dashed,name path=circ] (P) circle (1);
\draw[name path=line,->] (P)--++(-45:2);
\intpaths{Q}{circ}{line};
\punto[col=red]{Q};
```



# 11 Funzioni con punti in input e un numero come output

All'interno del pacchetto `liLaTeXtikz` sono presenti alcune funzioni che restituiscono un numero in output (prendendo dei punti come input). Queste sono le funzioni principali:

```
\distanza{\d}{A}{B};
```

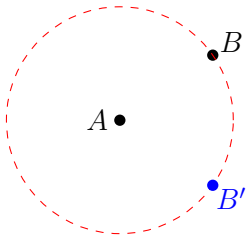
scrive in `\d` la distanza fra $A$ e $B$.

```
\angolopolare{\and}{P}{O};
```

scrive in `\ang` l'angolo del vettore $\overrightarrow{OP}$ (in gradi).

Vediamo un semplice esempio:

```
\pntl{A}{180}{0,0};
\pntl{B}{35}{35:1.5};
\distanza{\r}{A}{B};
\angolopolare{\a}{B}{A};
\draw[red,thin,dashed] (A) circle (\r);
\pntlp[blue]{B}{-35}{-\a:\r};
```




# 12 Manipolazione di coordinate

Il pacchetto `liLaTeXtikz` contiene diverse funzioni che ci permettono di manipolare le coordinate.

Le principali sono descritte in questa sezione:

```
\verso[L := 1]{P}{A}{B};
```

crea un punto $P$ sulla semiretta $AB$ (a distanza `L` da $A$).

```
\allont[L := 1]{P}{A}{B};
```

crea un punto $P$ sulla semiretta $BA$ (a distanza `L` da $A$).

```
\parallelo[L := 1]{P}{S}{A}{B};
```

crea un punto $P$, ottenuto traslando $S$ per una lunghezza $L$ nel verso nel vettore $\overrightarrow{AB}$.

```
\perpendicolare[L := 1]{P}{S}{A}{B};
```

crea un punto $P$, ottenuto traslando $S$ per una lunghezza $L$ nel verso perpendicolare a $\overrightarrow{AB}$ (si suppone $SAB$ in senso antiorario).

```
\medio[k := 0.5]{M}{A}{B};
```

definisce $M$ come il punto $(1-k)*A+k*B$. Nel caso speciale di $k=0.5$, $M$ è il punto medio del segmento $AB$. Quindi se $k=0$ risulta $M:=A$, mentre se $k=1$ risulta $M:=B$.

```
\opposto{P'}{P}{O};
```

definisce il punto $P'$ come l'opposto di $P$ rispetto ad $O$ (quindi $O$ risulterà essere il punto medio di $PP'$).

```
\simmetrico{P'}{P}{A}{B};
```

definisce il punto $P'$ come il simmetrico di $P$ rispetto alla retta $AB$.

```
\proj{H}{P}{A}{B};
```

definisce il punto $H$ come la proiezione ortogonale di $P$ sulla retta $AB$.

```
\sumvett{C}{A}{B}{D};
```

definisce il punto $C$ in modo che $\overrightarrow{AC}=\overrightarrow{AB}+\overrightarrow{AD}$. Questo vuol dire in particolare che $C$ sarà definito in modo

che il quadrilatero $ABCD$ sia un parallelogramma.

```
\ruotapunto{R}{P}{O}{a};
```

definisce il punto $R$ ottenuto ruotando $P$ in senso antiorario di un angolo `a` (con centro di rotazione in $O$).

```
\ruotapuntoN[N := 1]{R}{P}{O}{a};
```

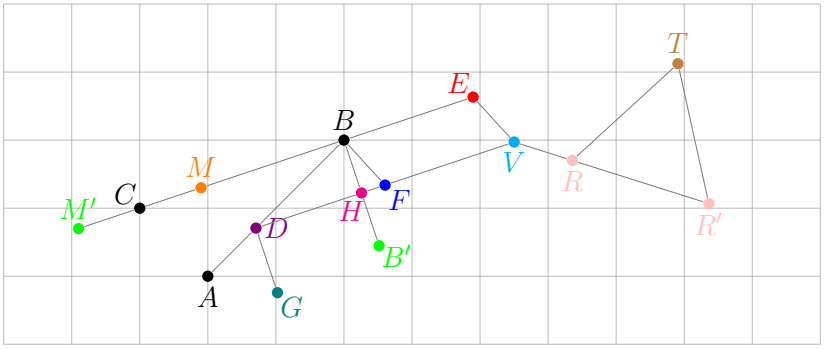
analogo a `\ruotapunto` ma normalizza il vettore $\overrightarrow{OR}$ in modo che misuri `L`.

```
\triangoloequilatero{C}{A}{B};
```

crea un punto $C$ in modo che il triangolo $ABC$ sia equilatero (con vertici in senso antiorario).

Vediamo un esempio in cui sono illustrate le funzioni di questa sezione.

```
\grigliaO{12}{5};
\pntl{A}{-90}{3,1};
\pntl{B}{90}{5,3};
\pntl{C}{135}{2,2};
\verso{D}{A}{B};
\pntl[violet]{D}{0}{D};
\allont[2]{E}{B}{C};
\pntl[red]{E}{135}{E};
\parallelo[2]{F}{D}{B}{E};
\pntl[blue]{F}{-45}{F};
\perpendicolare{G}{D}{B}{E};
\pntl[teal]{G}{-45}{G};
\medio[0.7]{M}{B}{C};
\pntl[orange]{M}{90}{M};
\opposto{Mp}{M}{C};
\pntlp[green]{M}{90}{Mp};
\simmetrico{Bp}{B}{D}{F};
\pntlp[green]{B}{-30}{Bp};
\proj{H}{B}{D}{F};
\pntl[magenta]{H}{-120}{H};
\sumvett{V}{B}{E}{F};
\pntl[cyan]{V}{-90}{V};
\ruotapunto{R}{E}{V}{-150};
\pntl[pink]{R}{-90}{R};
\ruotapuntoN[3]{Rp}{E}{V}{-150};
\pntlp[pink]{R}{-90}{Rp};
\triangoloequilatero{T}{R}{Rp};
\pntl[brown]{T}{90}{T};
\draw[gray] (Mp)--(C)--(M)--(B)--(E)--(V)--(F)--(H)--(D)--(G);
\draw[gray] (A)--(D)--(B) (F)--(B)--(H)--(Bp) (V)--(R)--(Rp)--(T)--(R);
```



# 13 Il triangolo

Se abbiamo un triangolo $ABC$ (si assume sempre che i vertici siano in ordine antiorario), possiamo trovare i piedi $P$ di altezza, mediana e bisettrice condotte da $B$ con i comandi

```
\piedealt{P}{A}{B}{C};
```

```
\piedemed{P}{A}{B}{C};
```

```
\piedebis{P}{A}{B}{C};
```

In modi analoghi possiamo trovare il baricentro $M$, l'ortocentro $H$, il circocentro $O$, e l'incentro $I$ del triangolo $ABC$ coi seguenti comandi:

```
\baricentro{M}{A}{B}{C};
```

```
\ortocentro{H}{A}{B}{C};
```

```
\circocentro{O}{A}{B}{C};
```

```
\incentro{I}{A}{B}{C};
```

In modo analogo, possiamo trovare l'excentro $E$ riferito al vertice $B$ come

```
\excentro{E}{A}{B}{C};
```

Abbiamo infine il comando

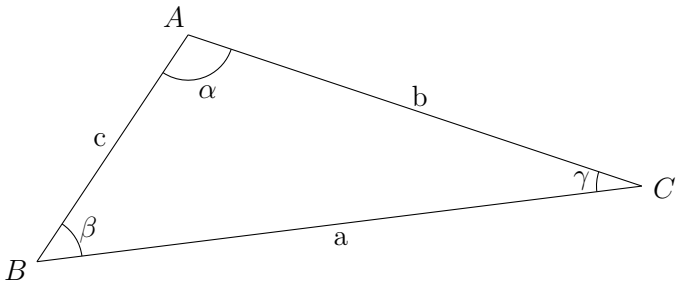
```
\triangolobase[keys]{A}{B}{C};
```

che disegna il triangolo nei vertici $A$, $B$, $C$ specificati. In questo caso le `keys` sono:

- `col`, `width`, `pattern` come descritti nella sezione 5.1.
- `col vertici`, `col lati` e `col angoli` specificano separatamente i colori per le varie componenti del triangolo (se non specificate copiano `col`).
- `lbl vertici := true`, `lbl lati := true` e `lbl angoli := true` dicono se creare dei label per vertici (nel caso saranno $A$, $B$ e $C$), lati (nel caso saranno a, b e c) e angoli (nel caso saranno $\alpha$, $\beta$ e $\gamma$).
- `lbl vertici dist := 0.3` la distanza dei label dei vertici dai vertici stessi.
- `fill angoli := false` se reso `true` gli angoli verranno riempiti, altrimenti resteranno vuoti.
- `col fill angoli` il colore del filling degli angoli (se specificato vengono riempiti).
- `opacity angoli := 30` il livello di trasparenza degli angoli. Utilizzato solo se `col fill angoli` viene lasciato vuoto ma viene specificato `fill angoli := true`.

Vediamo l'effetto di `\triangolobase`:

```
\triangolobase{0,3}{-2,0}{6,1};
```



# 14 Retta e semiretta

Possiamo disegnare una retta $AB$ o una semiretta $AB$ con i comandi

```
\retta[keys]{A}{B};
```

```
\semiretta[keys]{A}{B};
```

con le seguenti `keys`:

- `col`, `width`, `lbl`, `lbl style`, `lbl name`, `lbl dist := 0.2`, `lbl size`, `lbl col = col lbl` come illustrati nella sezione 5.1.
- `pattern draw` e `pattern prol := dashed` specificano il `pattern` per il tratto continuo e per quello tratteggiato.
- `name path` dà un nome al tratto continuo.
- `draw length`, se specificato aggiunge un tratto continuo lungo `draw length` oltre i vertici $A$ e $B$ (solo $B$ per `\semiretta`).
- `prol length := 0.8` la lunghezza del tratto tratteggiato.
- `flip lbl := false` se reso `true` ribalta i label.

# 15 Poligono e Poligonale

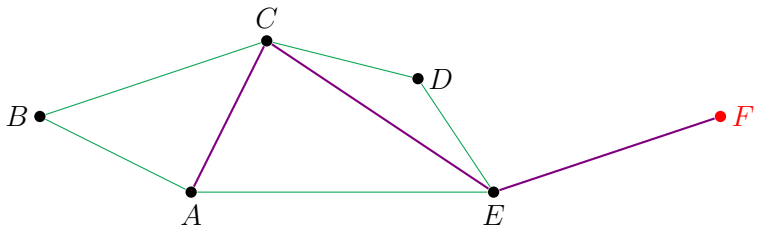
Nel pacchetto `liLaTeXtikz` ci sono anche due comandi utili per disegnare un poligono e una poligonale (una linea spezzata). Si usano con la sintassi:

```
\poligono[opzioni]{A,B,C,...};
```

```
\poligonale[opzioni]{A,B,C,...};
```

Il loro funzionamento è illustrato nel seguente esempio:

```
\pntl{A}{-90}{3,0};
\pntl{B}{180}{1,1};
\pntl{C}{90}{4,2};
\pntl{D}{0}{6,1.5};
\pntl{E}{-90}{7,0};
\pntl[red]{F}{0}{10,1};
\poligono[Green,fill=yellow]{A,B,C,D,E};
\poligonale[violet,thick]{A,C,E,F};
```




**Attenzione!** Come illustrato anche nell'esempio la chiave `fill` non è compatibile con punti definiti tramite `\punto` o simili. Per ovviare al problema serve definire i punti come `\coordinate` prima di usare `\poligono` e poi ripassare i punti in un secondo momento.

# 16 Circonferenze

La funzione

```
\circCR[keys]{C}{R};
```

disegna una circonferenza di centro $C$ e raggio $R$. Le `keys` sono:

- `col`, `width`, `pattern`, `name path`, `fill := false`, `col fill`, `opacity := 30`, `lbl`, `lbl style`, `lbl name`, `lbl sizee lbl col = col lbl` come descritte nella sezione 5.1.
- `lbl ang := -150` è l'angolo a cui viene posizionato il label (rispetto al centro della circonferenza).
- `lbl dist := 0.3` è la distanza del label dalla circonferenza.

Se invece vogliamo disegnare una circonferenza sapendo il centro $C$ e un suo punto $P$, possiamo usare le due funzioni (con le stesse `keys` di `\circCR`):

```
\circCP[keys]{C}{P};
```

```
\circCPr[keys]{C}{P}{r};
```

La differenza è che `\circCPr` registra in `r` la lunghezza del raggio della circonferenza disegnata.

Infine, possiamo disegnare una circonferenza passante per tre punti $A$, $B$ e $C$. Nel farlo possiamo registrare il centro in $O$ o il raggio in r usando i vari comandi qui definiti (hanno tutti le stesse `keys` di `\circCR`).

```
\circPPP[keys]{A}{B}{C};
```

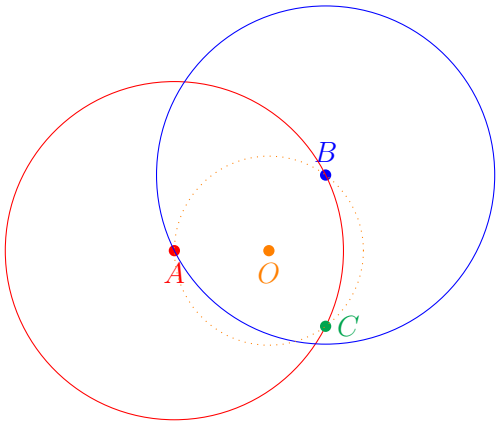
```
\circPPPc[keys]{A}{B}{C}{O};
```

```
\circPPPr[keys]{A}{B}{C}{c};
```

```
\circPPPcr[keys]{A}{B}{C}{O}{c};
```

Il seguente esempio illustra come usare i comandi per disegnare le circonferenze

```
\pntl[red]{A}{-90}{0,0};
\pntl[blue]{B}{90}{2,1};
\circCPr[col=red,width=thin]{A}{B}{\r};
\circCR[col=blue,width=thin]{B}{\r};
\pntl[Green]{C}{0}{2,-1};
\circPPPc[col=orange,pattern=dotted]{A}{B}{C}{O};
\pntl[orange]{O}{-90}{O};
```



## 16.1 Ottenere punti da circonferenze

È anche possibile trovare delle coordinate interessanti partendo da delle circonferenze.

Il comando

```
\interLC{P1}{P2}{A}{B}{O}{R};
```

trova i due punti di intersezione fra la retta $AB$ e la circonferenza di centro $O$ e raggio $R$ (si assume che $AB$ sia secante). Quindi registra questi due punti di intersezione in $P1$ e in $P2$.

Se $A$ è interno alla circonferenza, allora $P1$ sarà il punto di intersezione sulla semiretta $BA$, mentre $P2$ sarà il punto di intersezione sulla semiretta $AB$.

Se invece $A$ è esterno alla circonferenza, allora orientando la retta $AB$ (da $A$ verso $B$), $P1$ è la prima intersezione e $P2$ è la seconda intersezione.

Se invece vogliamo trovare i due punti di intersezione $P1$ e $P2$ fra due circonferenze di centri $C1$ e $C2$ e raggi $R1$ e $R2$ possiamo usare il comando

```
\interCC{P1}{P2}{O1}{R1}{O2}{R2};
```

Se $P$ è un punto esterno alla circonferenza di centro $O$ e raggio $R$, possiamo anche trovare i due punti di tangenza $T1$ e $T2$ uscenti da $P$. Per farlo ci basta usare il comando

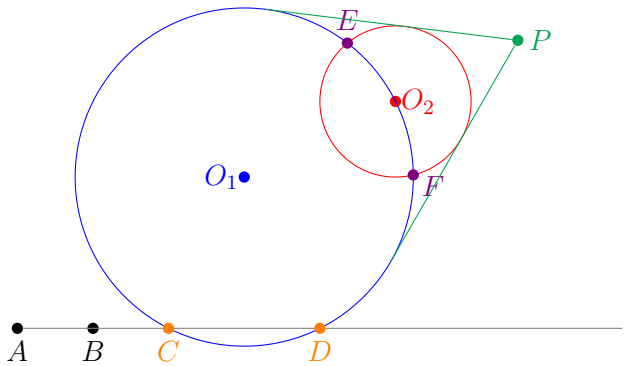
```
\tangentiPC{T1}{T2}{P}{O}{R};
```

Infine, date due circonferenze di centri $O1$ e $O2$ e raggi $R1$ e $R2$ (serve che $O1 \neq O2$ e $R1 \neq R2$) possiamo trovare il punto $P$ in cui si intersecano le due rette tangenti comuni alle due circonferenze. Per farlo usiamo il comando

```
\PtangenzaCC{P}{O1}{R1}{O2}{R2};
```

Vediamo un semplice esempio applicativo.

```
\pntlbl[blue]{Oa}{$O_1$}{180}{0,0};
\pntlbl[red]{Ob}{$O_2$}{0}{2,1};
\draw[red](Ob)circle(1);
\circCPr[col=blue]{Oa}{Ob}{\r};
\pntl{A}{-90}{-3,-2};
\pntl{B}{-90}{-2,-2};
\draw[thin,gray](A)--++(8,0);
\interLC{C}{D}{A}{B}{Oa}{\r};
\pntl[orange]{C}{-90}{C};
\pntl[orange]{D}{-90}{D};
\interCC{E}{F}{Oa}{\r}{Ob}{1};
\pntl[violet]{E}{90}{E};
\pntl[violet]{F}{-30}{F};
\PtangenzaCC{P}{Oa}{\r}{Ob}{1};
\pntl[Green]{P}{0}{P};
\tangentiPC{G}{H}{P}{Oa}{\r};
\draw[Green] (G)--(P)--(H);
```



## 16.2 Archi di circonferenza

A volte siamo anche interessati a disegnare degli archi di circonferenze. Questo può essere fatto usando semplicemente le opzioni della funzione `\draw` interna a `tikz`, come illustrato nella sezione 1.2.4. Il pacchetto `liLaTeXtikz` definisce però dei metodi alternativi per tracciare archi di circonferenze (ma non di ellissi).

Un primo comando ci permette di tracciare un arco di circonferenza di centro $O$ che parte dal punto $A$ e si ferma quando interseca la semiretta $OB$ (ruotando in senso antiorario). Per usarlo ci serviamo della sintassi:

```
\arcocirc[opzioni]{O}{A}{B};
```

In alternativa possiamo usare il comando

```
\arcocircang[opzioni]{O}{A}{ang};
```

che ci permette di disegnare un arco di centro $O$ che parte da $A$ e gira per un angolo complessivo `ang` (espresso in gradi). Se l'angolo è positivo si assume un senso di rotazione antiorario, se invece è positivo l'arco ruota in senso orario.

Abbiamo infine il comando

```
\semicirc[opzioni]{A}{B};
```

che disegna una semicirconferenza di diametro $AB$.

**Attenzione!** Se nelle opzioni vengono specificate delle frecce come `->`, gli archi sono sempre da intendersi tracciati in senso antiorario.

# 17 Plot generici

È possibile disegnare una funzione generica usando il comando

```
\plot[opzioni]{x_1 : x_2}{x}{y};
```

Questo disegna una curva parametrizzata (con parametro `\x` che varia nel range $[x_1, x_2]$). Al posto di $x$ e $y$ dobbiamo scrivere la dipendenza di $x$ e $y$ da questo parametro `\x`.

Ad esempio il seguente codice disegna la funzione $f(x) = \frac{\sin(x)}{x}$ per un intervallo $[-6 : +6]$:

```
\plot{-6 : +6}{\x}{sin(\x*180/pi)/(\x)};    % le funzioni sin e cos prendono l'input in gradi.
```

## 17.1 Seno e Coseno

Possiamo anche plottare le funzioni seno e coseno (con ampiezza $A$ per un intervallo $[x_1, x_2]$):

```
\plotsin[opzioni]{x_1 : x_2}{A};
```

```
\plotcos[opzioni]{x_1 : x_2}{A};
```

Volendo è anche possibile plottare le funzioni seno e coseno traslate di un vettore $V$, usando i comandi

```
\plotsinshift[opzioni]{x_1 : x_2}{A}{V};
```

```
\plotcosshift[opzioni]{x_1 : x_2}{A}{V};
```

Infine, è possibile plottare le funzioni seno e coseno dilatate lungo l'asse $x$ in modo che per ogni unità lungo $x$ vengano completate $N$ oscillazioni. Per farlo usiamo i comandi:

```
\plotsinfr[opzioni]{x_1 : x_2}{A}{N};
```

```
\plotcosfr[opzioni]{x_1 : x_2}{A}{N};
```

# 18 Coniche

Possiamo disegnare una conica di equazione implicita $ax^2 + by^2 + cx + dy + e = 0$ per valori di $x \in [x_1, x_2]$ utilizzando il comando

```
\conicaeq[keys]{x_1 : x_2}{a}{b}{c}{d}{e};
```

È anche possibile disegnare la versione più generale di equazione $ax^2 + by^2 + cxy + dx + ey + f = 0$ usando il comando

```
\conicageneq[keys]{x_1 : x_2}{a}{b}{c}{d}{e}{f};
```

## 18.1 Disegnare delle ellissi

Il pacchetto `liLaTeXtikz` mette a disposizione dell'utente diverse funzioni per disegnare delle ellissi.

La prima può essere usata per disegnare un'ellisse di centro $O$ e semiassi $a$ e $b$:

```
\ellisse[opzioni]{C}{a}{b};
```

È anche possibile ruotare l'ellisse di un angolo `ang` (in senso antiorario attorno a $C$). Per farlo ci serviamo del comando

```
\ellisserot[opzioni]{C}{a}{b}{ang};
```

Infine, abbiamo un comando che ci permette di disegnare un'ellisse partendo dalla sua equazione implicita $ax^2 + by^2 + cx + dy + e = 0$:

```
\ellisseeq[opzioni]{a}{b}{c}{d}{e};
```

## 18.2 Disegnare degli archi di parabole

Una prima possibilità consiste nel disegnare un arco di parabola da un punto $P_1$ a un punto $P_2$ e di vertice $V$. Per farlo serve però che $V$ sia compreso fra $P_1$ e $P_2$. In tal caso, possiamo usare il comando

```
\parabolaVPP[opzioni]{V}{P_1}{P_2};
```

Volendo esiste anche il comando `\parabolaVPPy`, identico a `\parabolaVPP`. Se invece vogliamo disegnare un arco di parabola con l'asse orizzontale (sempre fra due punti $P_1$ e $P_2$ e sempre con un vertice $V$ compreso fra $P_1$ e $P_2$) allora possiamo usare il comando

```
\parabolaVPPx[opzioni]{V}{P_1}{P_2};
```

Volendo possiamo anche disegnare una parabola partendo dalla sua equazione canonica $y = ax^2 + bx + c$. Per farlo, dobbiamo anche specificare il range $x \in [x_1, x_2]$ per cui vogliamo disegnare la parabola:

```
\parabolaeq[opzioni]{x_1 : x_2}{a}{b}{c};
```

Esiste anche il comando `\parabolaeqy`, identico a `\parabolaeq`. Se invece desideriamo disegnare un arco di parabola di equazione $x = ay^2 + by + c$ (questa volta per un intervallo $y \in [y_1, y_2]$ utilizziamo il comando

```
\parabolaeqx[opzioni]{y_1 : y_2}{a}{b}{c};
```

## 18.3 Disegnare una funzione di proporzionalità inversa

Possiamo anche disegnare un tratto di funzione di proporzionalità inversa (un'iperbole equilatera ruotata di 45°). Il seguente comando ci permette di disegnare un tratto di funzione $xy = k$:

```
\propinv[opzioni]{x_1 : x_2}{y_1 : y_2}{k};
```

Così facendo si disegno i punti $(x, y)$ tali per cui $xy = k$ e $x \in [x_1, x_2]$ e $y \in [y_1, y_2]$.

Volendo è anche possibile disegnare una funzione di proporzionalità inversa traslata di un vettore $V$. Per farlo usiamo il comando

```
\propinvshift[opzioni]{x_1 : x_2}{y_1 : y_2}{k}{V};
```

# 19 Segmenti, archi e angoli marcati

Spesso siamo interessati a disegnare oggetti marcandoli con dei segni. Le opzioni per i segni sono quelle descritte nella sezione 5.1, relativamente alla key `mark`.

Per disegnare un segmento $AB$ segnato utilizziamo il comando

```
\segmento[keys]{A}{B};
```

In questo caso le `keys` disponibili sono:

- `col`, `width`, `pattern`, `arrow`, `name path`, `lbl`, `lbl style`, `lbl name`, `lbl dist := 0.2`, `lbl size`, `lbl col = col lbl`, `mark := none`, `mark size := 0.2` e `col mark := black` come descritte nella sezione 5.1.

Può aver senso usare il comando `\segmento` anche se non siamo interessati a mettere un `mark`, dal momento che aggiunge comunque un label ben posizionato (in prossimità del punto medio del segmento).

Se vogliamo mettere un `mark` su un arco di circonferenza $\overset{\frown}{AB}$ (si suppone che la circonferenza abbia centro $O$) si usa il comando

```
\markarco[keys]{mark}{O}{A}{B};
```

Per le opzioni per mark si rimanda alle opzioni per la chiave `mark` descritte nella sezione 5.1. `\markarco` usa le seguenti `keys`:

- `col := black` che rappresenta il colore del mark.
- `size := 0.2` che rappresenta una misura della dimensione del mark.

**Attenzione!** `\segmento` disegna il segmento, a cui aggiunge un `mark`. Invece `\markarco` non disegna l'arco, ma si limita ad aggiungere il `mark`.

Per disegnare degli angoli $A\widehat{B}C$ (eventualmente marcati) abbiamo invece i seguenti comandi (il comando `\angoloretto` è specifico per gli angoli retti):

```
\angolo[keys]{A}{B}{C};
```

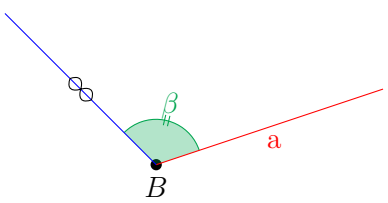
```
\angoloretto[keys]{A}{B}{C};
```

con le seguenti keys:

- `col`, `col draw`, `width`, `pattern`, `arrow`, `fill := false`, `col fill`, `opacity := 30` come descritti in sezione 5.1.
- `r` ha valori di default e significati diversi per i due comandi:
  - `r := 0.6` rappresenta il raggio dell'angolo per `\angolo`;
  - `r := 0.4` rappresenta il lato del quadratino per `\angoloretto`.
- `lbl`, `lbl dist := 0.2`, `lbl size`, `lbl col = col lbl`, `mark := none` come descritti in sezione 5.1 sono disponibili solo per `\angolo` (non per `\angoloretto`).

Vediamo un semplice esempio di utilizzo di questi comandi:

```
\coordinate (A) at (4,1);
\pntl{B}{-90}{1,0};
\coordinate (C) at (-1,2);
\angolo[lbl=$\beta$,col=Green,fill=true,mark=||]{A}{B}{C};
\segmento[col=red,lbl=a,lbl style=rm]{B}{A};
\segmento[col=blue,mark=oo]{B}{C};
```



# 20 Oggetti per la fisica

Nel pacchetto `liLaTeXtikz` ci sono alcuni comandi specifici per disegnare oggetti particolarmente ricorrenti nei problemi di fisica.

Questi oggetti vengono elencati in questa sezione.

## 20.1 La sfera

Il pacchetto `liLaTeXtikz` definisce una semplice funzione per disegnare una sfera tridimensionale:

```
\sfera[colore := teal]{centro}{raggio};
```

## 20.2 La molla

Il pacchetto `liLaTeXtikz` fornisce inoltre un comando specifico per disegnare una molla da un punto $A$ a un punto $B$:

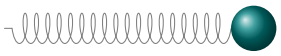
```
\molla[keys]{A}{B};
```

con le seguenti `keys`:

- `col := gray`, `width`, `s = scale := 1` con i significati spiegati nella sezione 5.1.
- `r := 0.2` rappresenta il raggio del cerchio della molla (in realtà il raggio è uguale a `s` · `r`).
- `step := 0.15` misura la distanza fra due cerchi consecutivi nella molla.
- `aspect := 0.3` ha a che fare con la forma della molla in prossimità dei punti $A$ e $B$. Conviene non toccare questo parametro.

Vediamo un semplice esempio di utilizzo:

```
\molla{0,0}{3,0};
\sfera{3.3,0}{0.3};
```



## 20.3 La carrucola

Nel problemi di statica del corpo rigido e di dinamica rotazionale dobbiamo spesso utilizzare le carrucole. Il pacchetto `liLaTeXtikz` definisce quindi il seguente comando per disegnare una carrucola di centro $C$ e raggio R:

```
\carrucola[keys]{C}{R};
```

con le seguenti `keys`:

- `col1 := gray!70`, `col2 := gray` e `col3 := black` sono i colori dei tre cerchi che compongono la carrucola (`col1` è il colore più esterno, mentre `col3` è quello più interno).
- `r2 := 2/3` e `r3 := 1/10` sono i raggi dei due cerchi interni (nello specifico, il cerchio più interno avrà raggio `r3` · R).

## 20.4 La cassa per il piano inclinato

Siamo spesso interessati a disegnare delle casse, soprattutto sui piani inclinati (di un angolo `ang`). Per farlo, possiamo usare il seguente comando

```
\cassa[keys]{punto iniziale}{x}{y}{ang};
```

dove $x$ e $y$ indicano le lunghezze della cassa lungo le due direzioni (possono eventualmente essere negativi). Le `keys` in questo caso sono:

- `col`, `width`, `pattern`, `col draw`, `col fill`, `opacity := 20`, `fill := true`, `lbl`, `lbl style := bf`, `lbl size`, `lbl col = col lbl := black` come descritti nella sezione 5.1.
- `centro := cassacentro` è il nome che verrà dato al centro della cassa (dove viene scritto l'eventuale label).

## 20.5 La parentesi graffa

La graffa viene spesso usata per specificare delle lunghezze nell'immagine. Per farlo, utilizziamo la sintassi seguente:

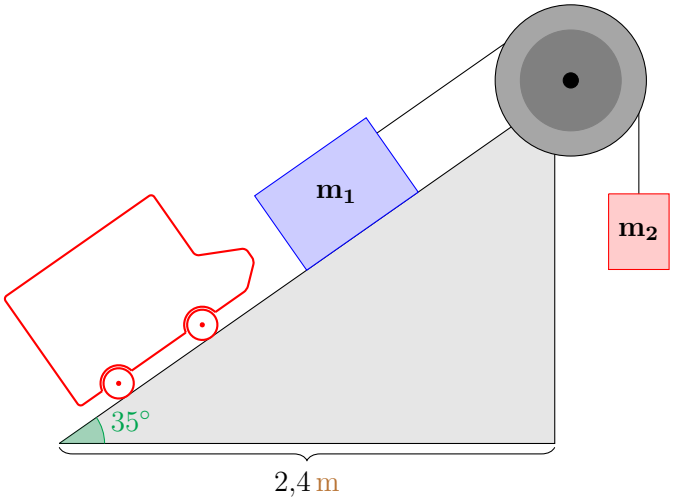
```
\graffa[keys]{A}{B};
```

dove la parentesi graffa ha gli estremi in $A$ e $B$. Il comando usa queste `keys`:

- `col`, `width`, `pattern`, `lbl`, `lbl style`, `lbl name`, `lbl dist := 0.5` e `lbl col = col lbl` come spiegati nella sezione 5.1.
- `mirror := false`, se posto uguale a `true` ribalta la graffa. Questo in realtà è equivalente a scambiare $A$ e $B$ nell'input.
- `amplitude := 5` è l'ampiezza della graffa. Tendenzialmente è un parametro da non toccare.
- `shape := brace` dice che dobbiamo disegnare una graffa (e non un altro tipo di parentesi). Anche questo è un parametro da non toccare.
- `dist := 0.05` la distanza dei punti della graffa dai punti $A$ e $B$.

**Attenzione!** Spesso vogliamo usare delle unità di misura all'interno del label. Questo è incompatibile con la key `lbl`. Per ottenere un label di questo tipo è quindi necessario utilizzare la key `lbl name`, come illustrato nelle ultime righe del seguente esempio.

```
\tikzdef\ang{35}
\coordinate (O) at(0,0);
\coordinate (NE) at (\ang:8);
\coordinate (E) at (O-|NE);
\poligono[fill=gray!20]{O,E,NE};
\angolo[col=Green,fill=true,lbl=$\ang^\circ$,lbl dist=0.4]{E}{O}{NE};
\coordinate (C) at ($(NE)+(45:0.3)$);
\draw ($(C)+(0.9,0)$)--++(0,-1.5);
\draw ($(C)+(\ang+90:0.9)$)--++(\ang:-2.5)node(top){};
\cassa[col=red,lbl=m_2]{$(C)+(0.9-0.4,-1.5)$}{0.8}{-1}{0};
\proj{H}{top}{O}{NE};
\cassa[col=blue,lbl=m_1]{H}{-1.8}{1.2}{\ang};
\carrucola{C}{1};
\graffa[lbl name=base]{O}{E};
\node (baselbl) at (base) {$2{,}4\,\SI{m}$};
\coordinate (A) at (\ang:2);
\begin{rotazione}[A]{\ang}
  \auto[col=red]{A};
\end{rotazione}
```



## 20.6 Il raggio di luce

In ottica siamo spesso interessati a disegnare un raggio di luce che segue una linea spezzata. Per farlo possiamo usare il comando

```
\raggioluce[keys]{P1}{P2}{P3}...;
```

dove $P_1$, $P_2$, $P_3$, ... sono i vertici della linea spezzata. Le `keys` sono:

- `col := orange`, `width`, `pattern`, `arrow := ->` come illustrati nella sezione 5.1.
- `arrow dist := 0.8` un valore nell'intervallo $[0, 1]$ che indica dove collocare la freccia specificata con `arrow` per ogni segmento della linea spezzata. Se il segmento è $AB$, ponendo `arrow dist := 0` la freccia verrà sovrapposta ad $A$, mentre ponendo `arrow dist := 1` la freccia verrà sovrapposta ad $B$.

Vediamo un semplice esempio di utilizzo:

```
\griglia0{8}{1};
\raggioluce{1,0}{4,1}{5,0}{7,1}{5,1};
```

## 20.7 La casetta

Il comando `\casetta` ci permette di disegnare una casetta:

```
\casetta[keys]{B};
```

dove $B$ è il centro della base della casetta. Le `keys` per questo comando sono:

- `col`, `width`, `pattern`, `s = scale := 1` come spiegate nella sezione 5.1.
- `y = h := 1.0` indica l'altezza della casa, mentre `x = w := 0.6` indica la sua larghezza (in realtà l'altezza e la larghezza sono rispettivamente `s` · `y` e `s` · `x`).

## 20.8 Omini stilizzati (`\uomo` e `\donna`)

Il pacchetto `liLaTeXtikz` definisce anche due comandi per disegnare degli omini stilizzati:

```
\uomo[keys]{B};
```

```
\donna[keys]{B};
```

dove $B$ indica il centro della base dell'immagine e le `keys` (uguali per i due comandi) sono:

- `col`, `width := thick`, `side := ->`, `s = scale := 1` come illustrate nella sezione 5.1.
- `h := 1` è l'altezza dell'omino (il realtà l'altezza sarà `s` · `h`).
- `lung braccio := 0.3` la lunghezza delle braccia (in realtà la lunghezza sarà `lung braccio` 1 · `s` · `h`).
- `ang braccio = ang braccio 1 := 0` l'angolo del primo braccio (ricorda che se `side := <-` viene comunque ribaltato).
- `ang braccio 2` l'angolo del secondo braccio. Se non viene specificato non viene disegnato.
- `ang gamba = ang gamba 1 := -60` l'angolo con cui viene disegnata la prima gamba.
- `ang gamba 2 := -90` l'angolo della seconda gamba.
- `mano 1 := ManoDx`, `mano 2 := ManoSx`, `piede 1 := PiedeDx`e `piede 2 := PiedeSx` sono i nomi da dare a mani e piedi (possono essere utili per disegnare altri oggetti successivi).

Vediamo un semplice esempio:

```
\griglia0[step=0.5]{6}{1.5};
\casetta[s=1.5]{2,0};
\uomo[col=blue]{4,0};
\donna[side=<-,col=red]{5,0};
```

## 20.9 I veicoli (\auto, \camion, \moto e \bici)

Il pacchetto liLaTeXtikz definisce anche comandi per disegnare diversi tipi di veicoli:

```
\auto[keys]{B};
```

```
\camion[keys]{B};
```

```
\moto[keys]{B};
```

```
\bici[keys]{B};
```

dove $B$ indica il centro della base del veicolo. I quattro comandi per i veicoli hanno tutti le stesse keys:

- col, width := thick, side := -> e s = scale := 1 come illustrate nella sezione 5.1.
- ang se specificato ruota l'immagine attorno a $B$ dell'angolo specificato.
- y = h e x = w indicano rispettivamente l'altezza e la larghezza del veicolo (in realtà l'altezza e la larghezza misurano rispettivamente s · y e s · x. Il valori di default cambiano in base al veicolo specificato, come qui illustrato:
  - per \auto valgono x := 3 e y := 2;
  - per \camion valgono x := 6 e y := 3;
  - per \moto valgono x := 2 e y := 1.5;
  - per \bici valgono x := 1.6 e y := 2.

La seguente immagine mostra il funzionamento dei veicoli:

```
\grigliaO{15}{3};
\camion{3,0};
\auto[col=red,side=<-]{8.5,0};
\moto[col=blue]{12,0};
\bici[col=Green,side=<-]{14,0};
```