SEMINAR & PSTricks LaTeX Stiles

for

Slides & PostScript

by

Timothy Van Zandt

`tvz@Princeton.EDU`

Introduzione a due stili LaTeX

di

Stefano ANSOLDI

email-address: `ansoldi@vstst0.ts.infn.it`

1 Introduzione.

SEMINAR.STY & **PSTRICKS** sono due pacchetti che aumentano considerevolmente le possibilità, peraltro tutt'altro che limitate, di LaTeX.
Il primo di essi è stato pensato esplicitamente per agevolare la preparazione di lucidi per conferenze o seminari, il secondo invece consente di utlizzare direttamente in LaTeX (e TeX!) le potenzialità proprie del linguaggio **PostScript** rendendo particolarmente agevole l'esecuzione di disegni e grafici da inserire nei propri lavori.
I pacchetti sono molto sofisticati, in grado di soddisfare praticamente tutte le esigenze, e vengono forniti con numerosi files di stile che consentono di sfruttarne appieno le potenzialità in maniera molto immediata.
È praticamente impossibile conoscere completamente tutte le molteplici funzioni di questi pacchetti, senza averli usati più e più volte. Per questo si cerherà di precisare soprattutto quello che *è possibile fare* piuttosto che *come farlo*. Poi ognuno potrà approfondire la conoscenza, tramite i manuali e la pratica, delle funzioni del pacchetto che gli risultano più utili, tralasciando completamente le altre ma ottenendo comunque risultati più che soddisfacenti.

Prima di presentare le caratteristiche salienti di questi pacchetti, si esporranno alcuni problemi che si possono avere utilizzandoli e che potrebbero scoraggiare un primo approccio verso di essi:

- i pacchetti sono molto "*vasti*" ed impararne tutte le caratteristiche in maniera sistematica può essere lungo e non molto divertente;
- non è possibile passare il file sorgente ad amici o colleghi che non possiedono i pacchetti.

Entrambe le difficoltà però possono essere aggirate. In primo luogo, come già detto, non è necessario imparare tutte le funzioni, ma solo quelle che interessano per il nostro scopo. In secondo luogo, l'installazione dei pacchetti non è particolarmente problematica e quindi si possono spedire a chiunque li voglia senza particolari problemi (ad esempio **Seminar** richiede semplicemente di avere alcuni files in una delle directory di input di LaTeX o TeX). Inoltre è sempre possibile con un minimo di tempo in più, soprattutto in relazione ai files in **PSTricks**, organizzare il sorgente in modo da poter processare separatamente le figure, e poterle poi inserire come si fa di solito, tramite macro come *psfig* o *epsfig*, spedendo poi questa seconda versione.

Questi lievi svantaggi sono compensati da vantaggi considerevoli. In primo luogo si ha un consumo di spazio su disco molto minore in quanto tutti i documenti vengono conservati come files di testo. Inoltre è più comodo apportare correzioni alle figure all'interno del file LaTeX ed inoltre si ha un notevole guadagno estetico perchè lo *stile* dei disegni è uguale a quello del testo. A queste "pignolerie" si aggiunge naturalmente il fatto che sfruttando le potenzialità offerte dai pacchetti è possibile ottenere risultati di molto superiori a quelli normalmente ottenibili in altri modi ed in molto meno tempo.

Per concludere è importante osservare che i programmi non sono *solo* stili LaTeX, ma possiedono una tale gamma di *style options*, che le necessità più comuni possono essere soddisfatte anche sapendo usare solo superficialmente i pacchetti, il che può essere ottenuto facilmente "giocando" con i files di esempio.

Tre semplici precisazioni:

- come si vedrà in seguito, certe opzioni di **Seminar** per funzionare richiedono che sia installato **PSTricks**;

- **PSTricks** definisce i nuovi comandi tramite istruzioni `\special` che di solito non sono riconosciute da *previewer per .dvi*, come *xdvi*; per vedere ciò che realmente verrà stampato è quindi necessario LaTeXare il file e poi processarlo anche con *DVIPS* osservando quindi il risultato con *GV* o equivalenti visori **PostScript**;

- se si è installato **Seminar**, ma non **PSTricks**, si possono ugualmente ottenere buoni risultati utilizzando le possibilità contenute in **Fancybox**, che consente di ottenere ugualmente il framing degli slides. Quello che purtroppo non si può fare è stampare contemporaneamente slides *landscape* e *portrait*, che quindi, pur potendo coesistere in uno stesso file, devono essre stampati in due blocchi separati (questo non altera comunque nè la corretta numerazione, nè eventuali *cross-reference*).

PSTricks: dove e cosa.

Directory **TEX_ROOT:[PSTRICKS.DOC]**

- PST-USR%.PS
- PST-QUICK.PS
- SAMPLES.TEX (LaTeX)
- PST-TEST.TEX (TeX)
- FILETEST.TEX (TeX)
- CHARTEST.TEX (TeX)
- DENIS1.TEX (TeX)
- DENIS2.TEX (TeX)
- TP-TEST.TEX (LaTeX)

FILES DI OPZIONI:

1. **pstricks**
2. pst-plot
3. multido
4. pst-coil
5. pst-node
6. gradient
7. pst2eps
8. charpath
9. textpath
10. colortab

2 PSTricks.

PSTricks è uno stile molto più vasto di **Seminar** e consente di utilizzare istruzioni **PostScript** direttamente in LaTeX & TeX. Più precisamente definisce dei nuovi comandi, che consentono di usare le potenzialità del **PostScript** senza dover conoscere approfonditamente il linguaggio di programmazione.

La distribuzione contiene non solo lo stile **PSTricks**, ma anche altri mini stili aggiuntivi, che sono elencati nello slide (2). Nel complesso il pacchetto si presenta come una potente interfaccia grafica. Le potenzialità che si acquisiscono vanno ben al di la' della semplice possibilità di tracciare linee, cerchi, spezzate: un'ampia gamma di qualificatori consente infatti di gestire tramite semplici parametri, il colore del testo, lo spessore delle linee, l'aggiunta di speciali terminatori (frecce e simili), il tracciamento di assi cartesiani, la costruzione di diagrammi di flusso, ecc., ecc..

Per questo non è conveniente imparare in "massa" tutte le possibilità offerte dal pacchetto, ma, una volta apprese le conoscenza di base (cioè la filosofia del programma e la struttura fondamentale della sintassi) si può cercare di volta i volta come risolvere le necessità grafiche che si hanno.

Nel seguito vedremo alcuni esempi in cui verrà evidenziato lo spirito di questo pacchetto, in modo che chiunque voglia usarlo abbia già le idee di base necessarie per poter affrontare il manuale senza perdere tempo.

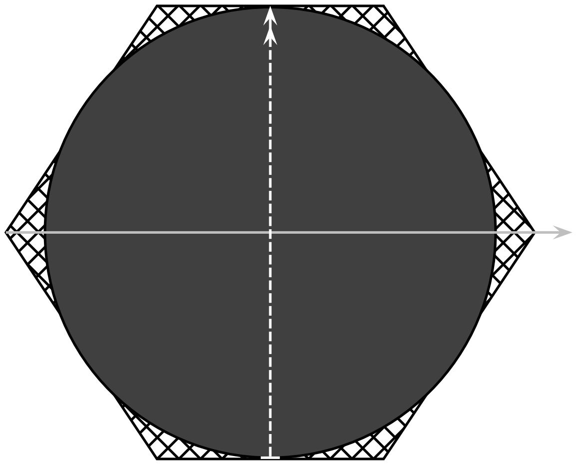
PSTricks

- **\command**[opt.arg.] $(x_1, y_1) \ldots (x_n, y_n)$ {arg}
- \black, \darkgray, \gray, \lightgray, \white,
  \red, \green, \blue, \cyan, \magenta, \yellow
- \psset ⇒ for Global (and Local) changes. Ad esempio:

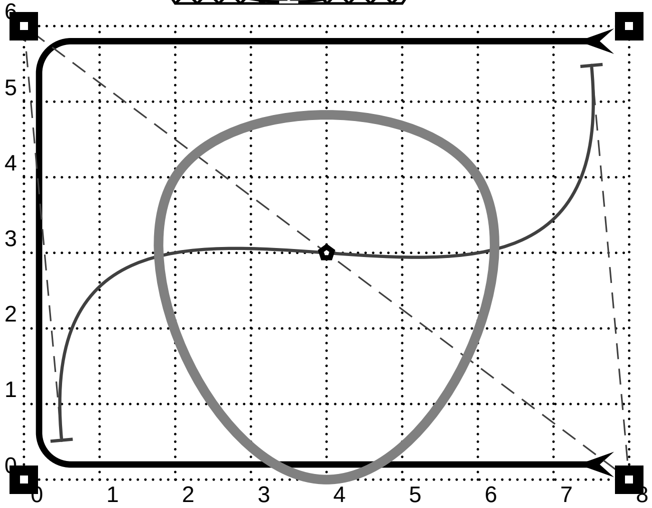
```
\psset{yunit=1pt}
\psset{xunit=1cm}
\psset{runit=1in}
\degrees[13]
```

- Graphics commands:
  1. \psline, \pspolygon ,\psframe, \pscircle, \pswedge, \psellipse, \psarc
  2. \psbezier, \parabola, \pscurve, \psccurve, \psdot
  3. \grid

Seguono alcuni esempi.



```
\pspolygon[fillcolor=black,fillstyle=crosshatch]
           (0,3)(2,6)(5,6)(7,3)(5,0)(2,0)
 \pscircle[fillcolor=darkgray,fillstyle=solid](3.5,3){3}
 \psline[linecolor=lightgray]{>}(0,3)(8.5,3)
 \psline[linestyle=dashed,dash=3pt 1pt,linecolor=white]
        {|->>}(3.5,0)(3.5,6)
```




```
\psgrid[subgriddiv=1,griddots=10,gridlabels=7pt](8,6)
  \psline[linewidth=2pt,linearc=10pt]{>-<}%
         (7.8,5.8)(0.2,5.8)(0.2,0.2)(7.8,0.2)
  \psbezier[linewidth=1pt,showpoints=true,%
            linecolor=darkgray]{|-|}%
           (0.5,0.5)(0,6)(8,0)(7.5,5.5)
  \psccurve[linewidth=3pt,linecolor=gray]%
           (4,0)(2,4)(6,4)
  \psdots[dotscale=8,dotstyle=square](0,0)(0,6)(8,0)(8,6)
  \psdots[dotscale=4,dotstyle=pentagon](4,3)
```

2.1 PSTRICKS.STY

Il primo stile di cui si parlerà è il principale, cioè `PSTRICKS`. Questo stile aggiunge la gran parte dei comandi grafici come mostrato nello slide (3). Quelli riportati non sono tutti i comandi, ma solo quelli più significativi, divisi in categorie. La sintassi quasi generale è

\command`[opt.arg.]`$(x_1, y_1) \ldots (x_n, y_n)$ `{arg}`

in cui tra parentesi quadre vanno gli argomenti opzionali. Le parentesi tonde delimitano sempre una coppia di coordinate (solo in un paio di casi queste ultime sono considerate parametri e vanno tra parentesi graffe), mentre le parentesi graffe sono riservate ad eventuali argomenti (cioè non sono necessariamente presenti in ogni comando), ad esempio nei comandi che ruotano qualcosa, ecc.

Importante è il comando

`\psset{arg}`

che consente di settare vari parametri PostScript per le istruzioni che seguono. È comunque sempre possibile settare questi parametri utilizzando la possibilità di introdurre gli argomenti opzionali nel comando grafico.

Un altro comando che può essere utile è

`\psgrid[opt.]`$(x_1,y_1)(x_2,y_2)(x_3,y_3)$

che traccia una griglia con angoli opposti (x_2,y_2) e (x_3,y_3) ed origine (se si vogliono le label) in (x_0,y_0). Numero delle suddivisioni, spessore delle linee principali e secondarie, tipo e posizione delle labels, colore, possono essere controllati tramite vari parametri.

Ecco la sintassi di alcuni comandi:

- `\psline*[par]{arr}(coor1)...(coorn)`
- `\pspolygon*[par](coor1)...(coorn)`
- `\psframe*[par](coor1)(coor2)`
- `\pscircle*[par](centro){raggio}`

- `\pswedge*[par](centro){raggio}{ang1}{ang2}`
- `\pscurve*[par]{arr}(coor1)...(coorn)`
- `\psdot*[par](coor1)...(coorn)`

I parametri grafici (`par` nelle righe precedenti) non influenzano solo lo spessore e il colore delle linee, ma anche il tipo (ad esempio si può usare `[linestyle=solid]` o `dotted` o `dashed`) personalizzando ad esempio il pattern (provare `[linestyle=dashed,dash=15pt 5pt]`. È addirittura possibile scegliere se avere gli spigoli arrotondati o vivi (parametro `linearc`) e così via.

Le frecce agli oggetti possono essere aggiunte, ove permesso, sostituendo ad `arr` vari simboli, $\{>\}$ $\{<\}$ $\{>>\}$ $\{|->\}$ ad esempio, nelle più svariate combinazioni.

Degno di nota è il parametro `[showpoints=true]` che pone dei punti nelle coordinate che si assegnano, in modo da poter correggere più facilmente il disegno.

L'asterisco in tutti i comandi è **opzionale** e consente di eseguire un **fill** automatico del **path** del colore `fillcolor` e nello stile `fillstyle` entrambi settabili tramite `\psset`.

Degli esempi di utilizzo di questi comandi sono nello slide (4).

Altri comandi:

- `\begin{pspicture}[bas](coor1)(coor2)`
  `...`
  `\end{pspicture}`

  `bas` frazione dell'altezza da allineare con TEX *currentpoint*

  `(coor1)(coor2)` coordinate nella forma *lower-left–upper-right*

  All'interno dell'*environment*:

  `\rput*[ref]{rot}`(x,y)`{stuff}`

- `\pscustom[opt.arg.]{graphic commands}`
- `\psclip ...graphic commands ...  \endpsclip`

Esempio:

```
\begin{pspicture}(0,0)(10,10)
\psframe[...](10,10)
\psclip{%
     \pscustom[linestyle=none]{%
               \parabola(1,7)(5,1)
               \rlineto(0,3) \rlineto(-8,0)
               \closepath}
     \pscustom[linestyle=none]{%
               \psarc(5,5){4}{0}{180}
               \rlineto(0,-5) \rlineto(8,0)
               \closepath}}
    \rput[lt]{90}(1,1){\parbox[t]{8cm}{%
                   Questo \`e un esempio...
                   ...
                   ...altri effetti speciali.}}
\endpsclip
\end{pspicture}
```

Un comando che si presta a svariati utilizzi è il comando

`\begin{pspicture}[bas](coor1)(coor2)...\end{pspicture}`

simile al comando LaTeX

`\begin{picture}(coor1)(coor2)...\end{picture}`

Il significato dei vari parametri è spiegato nello slide (5). Se si vuole che la *reference line* della figura sia al centro si può usare `[]`. In tale *environment* si possono utilizzare tutti i comandi grafici già descritti, più il comando

`\rput*[ref]{rot}`(x, y)`{stuff}`

che consente di posizionare il *LR-box* `stuff` nella posizione (x, y), ruotato di un angolo `rot`; con l'eventuale settaggio `[ref]` si può al solito determinare il *reference point* di `stuff`, che per default è il centro. È possibile così generare semplici (e complessi) disegni, sfruttando i comandi grafici già visti, oppure personalizzare grafici e disegni ottenuti con altri programmi (ad esempio lo `stuff` di un `\rput` può tranquillamente essere un file **PS**).

Nell'esempio che si vede nello slide (6) si coglie l'occasione per descrivere due interessanti comandi che possono essere usati all'interno di un `pspicture`–*environment*, cioè

`\pscustom[opt.arg.]{graphic commands}`

e

`\psclip ... \endpsclip`

dei quali il primo serve a definire dei propri oggetti grafici ed il secondo ad eseguire dei *clipping*. All'interno di `\pscustom` si possono usare i più svariati comandi grafici (fare riferimento al manuale per conoscerli tutti).

Ma ecco l'esempio (slide (6)), in cui si mostra un utilizzo di `\psclip` insieme a `\pscustom`. Ciascun comando `\pscustom` definisce un percorso chiuso, il primo costituito da un arco di parabola e tre segmenti, il secondo da un'arco di circonferenza e tre segmenti (notare l'utilizzo di `\closepath`). Il parametro `[linestyle=none]` garantisce che non venga eseguito uno *stroke* del *path*. Quindi il comando `\psclip`, che ha per argomento proprio il pscustom, "clippa" quella parte di oggetti grafici, che si trovano al suo interno (cioè l'`\rput{90}`... con questa minipage ruotata di 90 gradi; notare il parametro `l`*eft*`t`*op*) secondo il percorso definito dai due `\pscustom`. Una caratteristica importante di `\psclip` è che se si inseriscono, come argomenti più percorsi chiusi, il "clipping" avviene nell'intersezione, che è proprio il caso di questo esempio. Questi comandi possono essere utlizzati ad esempio per ombreggiare delle regioni dalla forma strana o per altri effetti speciali.

m. Ciascun
ennisce un percorso chiu
ostituito da un arco di parab
re segmentiil secondoe da un'arco
rconferenza e tre segmenti (notare l'u-
tilizzo di `\closepath`). Il parametro
`[linestyle=none]` garantisce che non venga
eguito uno *stroke* del *path*. Quindi il co
o `\psclip`, che ha per argomento pr
custom, “clippa” quella part
he si trovano al suo i
con q

PST-PLOT

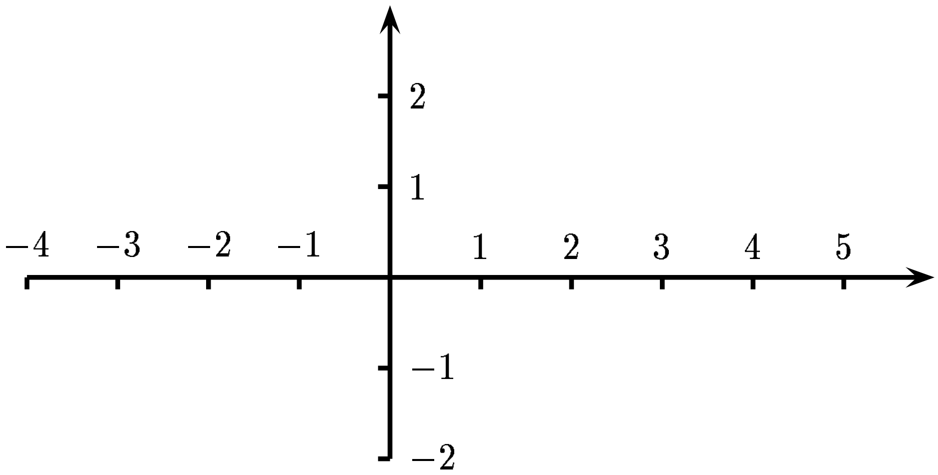
TRE INTERESSANTI COMANDI INTRODOTTI:

1. `\fileplot[par]{file}`
2. `\psplot[par]{min}{max}{ps-func}`
3. `\parametricplot[par]{min}{max}{ps-func}`

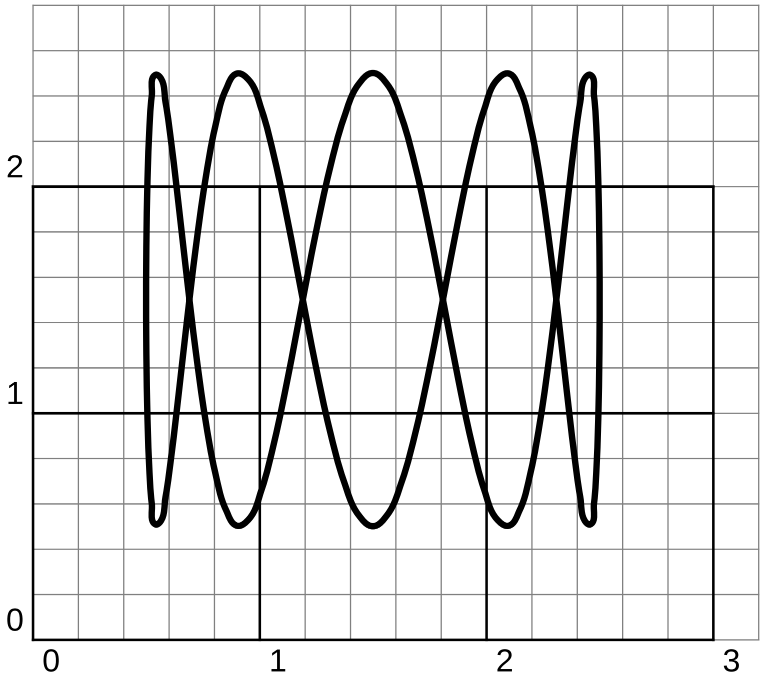
Parametro importante per `\psplot` e `\parametricplot` è **plotpoints**, per tutte **plotstyle**.

È possibile ottenere degli assi cartesiani tramite

`\psaxes[param]{arrows}(coor1)(coor2)(coor3)`




```
\psaxes[linewidth=1.5pt,
        tickstyle=top,
        labels=all]{>}(4,2)(0,0)(10,5)
```




```
\begin{pspicture}(3.25,3)
    \psgrid
    \rput{0}(1.5,1.5){\parametricplot%
                      [plotstyle=ccurve,%
                       plotpoints=99,%
                       linewidth=2pt]%
                      {0}{360}{t sin 5 t mul cos}
                     }
\end{pspicture}
```

2.2 PST-PLOT.STY

Uno degli ulteriori stili aggiunti dal pacchetto è `PST-PLOT` che è pensato per consentire di graficare semplici funzioni direttamente sul proprio lavoro, con un minimo di conoscenza del linguaggio PostScript. Due comandi particolarmente interessanti introdotti da questa *style-option* sono

\psplot*[par]{x_{min}}{x_{max}}{ps-func}

\parametricplot*[par]{t_{min}}{t_{max}}{ps-func}

dove x_{min}, x_{max} (risp. t_{min}, t_{max}) sono gli estremi della variabile (risp. parametro) e `ps-func` è il codice **PostScript** del grafico da tracciare, in cui x e t rispettivamente devono comparire come variabile. Nel primo caso si grafica una funzione di x e nel secondo un grafico parametrico in funzione del parametro t. `par` consente di selezionare il "tratto" di tracciatura, a linea, punti, ecc (cioè `curve`, `dots`, `ccurve`...). Importante è anche il parametro `plotpoints=`*num* che consente di scegliere il numero di punti da tracciare. Nello slide (8) c'è un esempio.

Tracciare funzioni è utile ma spesso servono anche degli assi cartesiani per completare il grafico. Si può ottenere rapidamente ciò tramite il comando

`\psaxes[par]{arr}(coor1)(coor2)(coor3)`

`(coor2)` e `(coor3)` indicano la posizione degli angoli inferiore sinistro e superiore destro del rettangolo che contiene gli assi, mentre `(coor1)` designa la posizione dell'origine (esempio nello slide (7)). Anche per questo comando vi sono numerosi parametri; ad esempio

1. `labels=none/x/y/all` consente di scegliere se mettere i numeri;
2. `ticks=none/x/y/all` consente di scegliere se mettere le tacche;

3. `tickstyle=full/top/bottom` consente di scegliere se le tacche devono estendersi dal lato delle label, dal lato opposto o da entrambi;

4. ...

MULTIDO

Uno stile per eseguire un **DO-LOOP**!

Il comando è

```
\multido{variables}{repetitions}{stuff}
```

variables è una control-sequence in LaTeXche inizi per **d** o **n** o **i** o **r**, nella forma *valore iniziale* + *incremento*:

1. **d** stà per distanza e i parametri devono essere delle *lenght* per LaTeX; in fase di output si ottiene un valore in `sp` (output significa "*stampa*");
2. **n** stà per numero e i parametri devono essere numeri con le parti decimali lunghe uguali;
3. **i** stà per intero ed i parametri devono essere numeri interi;
4. **r** stà per reale ed i parametri devono essere numeri reali;

repetitions è il numero di ripetizioni da eseguire;

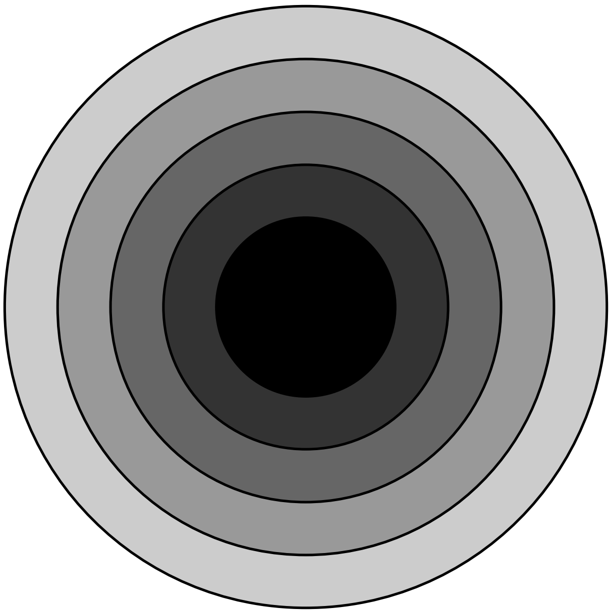
stuff è la sequenza LaTeX da ripetere, che può contenere le variabili utilizzate in *variables*

Ecco alcuni esempi

```
\multido{\int=1+2}{10}{\int,\ }
```

1, 3, 5, 7, 9, 11, 13, 15, 17, 19,

```
\begin{pspicture}(-4.5,-4.5)(4.5,4.5)
    \newgray{mygray}{0}
    \psset{fillstyle=solid,fillcolor=mygray}
    \multido{\nm=1.0+-0.15,\dim=4cm+-.7cm}{5}
            {\newgray{mygray}{\nm}
             \rput{0}(0,0){\pscircle(0,0){\dim}}}
\end{pspicture}
```



2.3 MULTIDO.STY

Una *style option* particolarmente interessante è anche `multido`. Tramite questa opzione è possibile creare dei loop, inserendo dei comandi grafici in cui le variabili del **DO LOOP** compaiono come parametri; si può quindi stampare più volte un disegno, magari cambiando solo il colore o la posizione, o stampare dei *mark* su degli assi, e cose simili. Due esempi sono riportati negli slides (9-10). Nello slide (9) è riportata la sintassi e ci si soffermerà quindi brevemente sul secondo esempio (slide (10)).

Osserviamo dapprima che un'istruzione **multido** può anche apparire all'interno di una figura, come nel caso in esame. La prima riga dopo il `\begin{pspicture}`, cioè

```
\newgray{mygray}{0}
```

serve solo a definire il colore `mygray` in modo da non ottenere un errore alla successiva istruzione,

```
\psset{fillstyle=solid,fillcolor=mygray}
```

che lo richiama e serve a fissare i parametri **PostScript** per le successive istruzioni: in particolare si fissa lo stile per il *filling* ed il relativo colore. Si arriva così al `multido`:

> Questo ripete per 5 volte il disegno di un cerchio di raggio `\dim` che risulterà *fillato* grazie alla precedente istruzione `\psset` del colore `\mygray`, che viene ridefinito prima di ogni tracciatura dal noto comando `\newgray`. Ad ogni ciclo le variabili `\nm` e `\dim` vengono aumentate del rispettivo incremento (-0.15 e $-.7$ cm rispettivamente): si noti che poichè il secondo parametro del comando `\newgray` è un numero, tale è il *tipo* di `\nm`, mentre `\dim` è una distanza, come deve essere il raggio del cerchio.

Si vede così come questa istruzione consenta di semplificare tutte le operazioni ripetitive.

PST-NODE & PST-COIL

I comandi aggiunti da **PST-NODE** si dividono in 3 categorie:

definizione di un nodo: consistono nell'assegnare un nome e nel posizionare un certo oggetto grafico:

1. `\rnode`, `\Rnode`
2. `\pnode`
3. `\cnode`, `\circlenode`, `\cnodeput`
4. `\ovalnode`

connessioni tra nodi: consentono di connettere tra loro i nodi richiamandoli con il loro nome:

1. `\ncLine`, `\ncbar`, `\nc`
2. `\nccurve`, `\nccircle`, `\ncloop`
3. `\ncangle`, `\ncangles`, `\ncdiag` `\ncdiagg`
4. `\nccoil`, `\nczigzag`

"labels" di nodi: attribuiscono delle labels alle connessioni:

1. `\put`, `\aput`, `\bput`
2. `\mput`, `\Aput`, `\Bput`

Comandi di **PST-COIL**:

- `\pscoil`
- `\psCoil`
- `\pszigzag`

$$\begin{tikzcd}
X \arrow[r, two heads, "\psi"] \arrow[d, two heads, "\phi"'] & Y \arrow[r, leftrightarrow, dotted, "\tau", "\sim"'] \arrow[dl, leftrightarrow, dashed, "\chi"', "\sim"] & \tilde{Y} \arrow[dr, leftrightarrow, dashed, "\tilde{\chi}"', "\sim"] & \tilde{X} \arrow[l, two heads, "\tilde{\psi}"'] \arrow[d, two heads, "\tilde{\phi}"] \\
X/ker(\psi) \arrow[rrr, leftrightarrow, dotted, "\sigma", "\sim"'] & & & \tilde{X}/ker(\tilde{\psi})
\end{tikzcd}$$

```
$$
\begin{array}{cccccc}
    \Rnode{a}{X} & \Rnode{b}{Y} & \QQ &
    \Rnode{c}{\tilde{Y}} & \QQ & \Rnode{d}{\tilde{X}}
    \\[3\semcm]
    \Rnode{aa}{X/ker(\psi)} & & & & &
    \Rnode{dd}{\tilde{x}/ker(\tilde{\psi})}
    \\[2\semcm]
\end{array}
\psset{nodesep=5pt} \everypsbox{\scriptstyle}
\ncLine{->>}{a}{b}\Aput{\psi}
\ncLine{<<-}{c}{d}\Aput{\tilde{\psi}}
\ncLine{->>}{a}{aa}\Bput{\phi}
\ncLine{->>}{d}{dd}\Aput{\tilde{\phi}}
\ncLine[linestyle=dashed,dash=2pt 2pt]%
{<->}{aa}{b}\Aput{\chi}\Bput{\sim}
\ncLine[linestyle=dashed,dash=2pt 2pt]%
       {<->}{dd}{c}\Aput{\tilde{\chi}}\Bput{\sim}
\ncLine[linestyle=dotted]%
       {<->}{aa}{dd}\Aput{\sigma}\Bput{\sim}
\ncLine[linestyle=dotted]{<->}{b}{c}\Aput{\tau}\Bput{\sim}
$$
```

2.4 PST-NODE.STY & PST-COIL.STY.

PST-COIL.STY è semplicemente un file che consente di tracciare particolari linee a forma di *molla* e *zig–zag*; come al solito un certo numero di parametri consente di dimensionare nella maniera preferita gli oggetti.

Più interessante, **PST-NODE.STY** semplifica tramite i comandi illustrati nello slide (11) la stesura di diagrammi di flusso e di tutti i diagrammi in cui si vogliono connettere tramite frecce vari *item* (ad esempio nei diagrammi che spiegano la dipendenza tra i vari capitoli di un libro o anche nei diagrammi matematici, ecc.).

Come evidenziato nello slide le istruzioni introdotte si dividono in tre tipi:

1. istruzioni di assegnazione, che consentono di stampare un *box* di grafica e/o testo e di associargli un nome, con il quale in seguito potrà essere richiamato; queste istruzioni si distinguono perchè incorniciano diversamente (o nient'affatto) il *box* o perchè fissano dei diversi *punti di riferimento* per le successive connessioni;

2. istruzioni di connessione, che consentono di connettere tra loro due nodi. Hanno la sintassi

   `\nc...{arr}{node1}{node2}`

   ed ovviamente connettono il nodo `node1` con quello `node2` in vari modi (curve spezzate, ecc.); al solito vi sono dei parametri che consentono di personalizzare le varie caratteristiche, spessore, lunghezza dei vari tratti curvatura, ecc.;

3. istruzioni di *labeling*, per assegnare dei nomi alle connessioni; in generale una di queste istruzioni si presenta come

   `\...put[ref]{rot}(pos){lab}`

   con il seguente significato dei simboli:

   `ref` è il punto di riferimento della label che va posto alle coordinate indicate;

   `rot` è l'angolo di rotazione;

`pos` è la posizione in coordinate;

`lab` è la label;

le istruzioni che iniziano con lettera maiuscola usano il default per i primi tre parametri, così come `\mput`.

Nello slide (12) c'è un esempio di diagramma di commutazione, ottenuto in poche righe. Si osservi come i *boxes* che rappresentano i nodi possano essere inseriti in normali istruzioni LaTeX!!!

GRADIENT.STY

```
\psframe[fillstyle=gradient,%
         gradbegin=white,%
         gradend=black,%
         gradmidpoint=.333,%
         gradangle=45](10,10)
```

2.5 GRADIENT.STY

`gradient` è un'opzione per aggiungere un nuovo tipo di *filling*, *sfumato*. Utilizzando questa opzione si può usare, in tutti i comandi grafici che ammettono *filling*, lo stile

`fillstyle=gradient`

Parametri ulteriori introdotti sono:

`gradbegin` colore con cui deve iniziare e terminare la sfumatura;

`gradend` colore al *midpoint*;

`gradmidpoint` è la frazione dell'altezza a cui si pone il *midpoint*;

`gradangle` è l'angolo di *filling*.

Un semplice esempio è nello slide (13)

PST2EPS.STY

Come creare files `.EPS` da comandi grafici **PSTricks**:

`\PSTtoEPS[par]{file}{graph}`

Parametri (`par`):

- `bbllx=...`, `bblly=...`, `bburx=...`, `bbury=...` consentono di fissare il **BoundingBox** che altrimenti per default è $(-1,-1)$ $(1,1)$;
- `headerfile` fissa eventuali files di *header* particolari.
- `headers=none/all/user` seleziona i files di *header* da inserire.

CHARPATH.STY, TEXTPATH.STY, COLORTAB.STY

Fillare un testo, scrivere un testo lungo un percorso, ottenere tabelle a colori.

2.6 PST2EPS.STY

Interessante è il comando

`\PSTtoEPS[par]{file}{graph}`

la cui sintassi è descritta nell'ultimo slide; questo comando consente infatti di creare un file **.EPS** dai comandi presenti nel suo argomento `graph`. Tale file verrà chiamato con il nome `file.eps` e potrà essere inserito nello stesso documento, anche a partire dalla riga successiva.

Questo comando è particolarmente interessante perchè consente di spedire un nostro lavoro anche a chi non possiede **PSTricks**, generando prima i files **EPS** e poi modificando il programma in modo che nel programma che spediremo, vengano eseguiti degli input, grazie a macro come *psfig* o *epsfig*, dei files creati, al posto dell'esecuzione del codice in comandi **PSTricks**.

È importante ricordarsi di aggiungere il **BoundingBox** tra i parametri, come descritto nell'ultimo slide e di usare il parametro

`headers=all`

in modo che i files **EPS** contengano tutte le intestazioni necessarie per non generare errori in fase di stampa.

2.7 CHARPATH.STY, TEXTPATH.STY, COLORTAB.STY

Questi stili consentono rispettivamente di:

- fillare un certo testo con il *pattern* scelto tramite i parametri `fillstyle`, `fillcolor`;
- scrivere un testo lungo una curva o qualsiasi altro percorso definito da comandi grafici di **PSTricks**;
- ottenere tabelle a colori.

Non si descrivono ulteriormente, perchè dalla stessa guida di **PSTricks** sono definiti meno **robusti**, cioè non completamente affidabili!