

Vẽ hình khoa học TikZ - Asymptote

HỌC TIKZ
theo cách của BẠN

Bùi Quỹ - 2020

Mục lục

I Cơ bản vẽ hình bằng Tikz	4
I.1 Khai báo và môi trường vẽ	4
I.2 Các đối tượng hình học	4
I.2.1 Hệ trục tọa độ	4
I.2.2 Các lệnh cơ bản của Tikz	6
I.2.3 Khai báo điểm, tham số	6
I.2.4 Các hàm tính toán số liệu cơ bản	6
I.2.5 Các phép toán đối với tọa độ	7
I.2.6 Hiển thị văn bản trong Tikz	8
I.2.7 Vẽ Điểm	8
I.2.8 Đoạn thẳng, đường gấp khúc	9
I.2.9 Đường tròn, Ellipse. Cung tròn, cung Ellipse	11
I.2.10 Các dạng đường cong khác	13
I.2.11 Lấy một điểm trên đường cong, tiếp tuyến của đường cong	15
I.2.12 Giao điểm của các đường	19
I.2.13 Môi trường scope, clip, node và pic	20
I.3 Các tùy chọn thường dùng	25
I.3.1 Tùy chọn chung	25
I.3.2 Với node, pic	26
I.3.3 Vòng lặp foreach	26
I.3.4 Tạo macro	28
I.3.5 Vẽ hình không gian (giả 3d)	30
II Bảng xét dấu - Bảng biến thiên - Đồ thị hàm số	33
II.1 Bảng xét dấu	33
II.2 Bảng biến thiên	33
II.3 Kết hợp Tikz và tkz-tab vẽ bảng biến thiên tùy chỉnh	37
II.4 Vẽ bảng biến thiên bằng tikz thuần	40
II.5 Đồ thị hàm số	41
II.6 Đồ thị hàm số trong hệ tọa độ cực	43
III Tô miền đồ thị hàm số với Tikz	45
III.1 Vẽ đồ thị hàm số	45
III.2 Tô miền đồ thị hàm số	47
III.2.1 Khi đã biết hoành độ các điểm cận của miền cần tô	47
III.2.2 Tô miền đồ thị không biết hoành độ giao điểm	50
III.2.3 Sử dụng even odd rule	53
III.3 Các kiểu tô miền	55

IV Tư duy hình họa để vẽ hình	58
IV.1 Trình tự cơ bản	58
IV.2 Lựa chọn hệ tọa độ hợp lý	59
IV.3 Sử dụng Macro	59
IV.4 Sử dụng vòng lặp foreach	59
IV.5 Kỹ năng tìm kiếm	60
V Export hình ảnh	61
V.1 Cài đặt ImageMagick và Ghostscript	61
V.2 Làm ảnh động trực tiếp trên PDF	65
V.3 Làm ảnh động gif để upload lên web	67
VI Các thư viện thường dùng	70
VI.1 calc	70
VI.2 intersections	70
VI.3 angles	71
VI.4 patterns	71
VI.5 Decoration	71
VI.6 Shadings	72

I Cơ bản vẽ hình bằng Tikz

I.1 Khai báo và môi trường vẽ

Khai báo gói lệnh:

```
\usepackage{tikz}  
\usetikzlibrary{calc,angles}
```

Môi trường vẽ:

```
\begin{tikzpicture}  
.....  
\draw ....  
.....  
\end{tikzpicture}
```

Như vậy. Một file vẽ hình bằng **Tikz** có thể đầy đủ để bạn biên dịch:

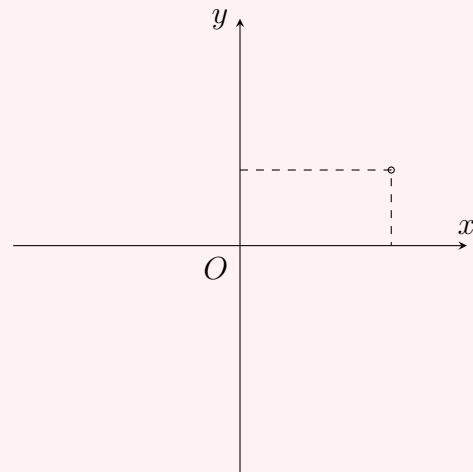
```
\documentclass{standalone}  
\usepackage{tikz}  
\usetikzlibrary{calc,angles}  
\begin{document}  
\begin{tikzpicture}  
.....  
\draw ....  
.....  
\end{tikzpicture}  
\end{document}
```

I.2 Các đối tượng hình học

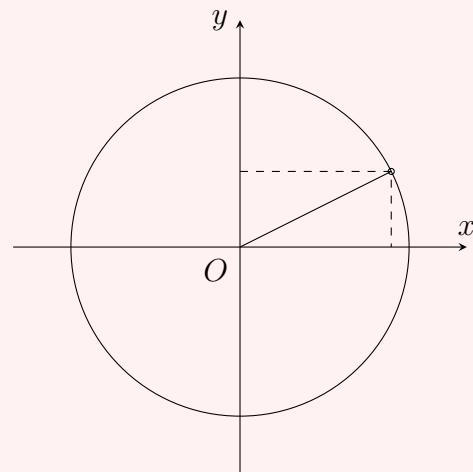
I.2.1 Hệ trục tọa độ

Các đối tượng hình học luôn gắn với hệ tọa độ. Chúng ta có thể sử dụng một trong hai hệ tọa độ để xác định các đối tượng cần vẽ: Hệ tọa độ vuông góc *Oxy* hoặc tọa độ cực.

```
\begin{tikzpicture}[>=stealth]
\draw[->] (-3,0)--(3,0);
\draw[->] (0,-3)--(0,3);
\draw[dashed] (0,1)--(2,1)--(2,0);
\draw (2,1) circle (0.04);
\draw (3,0) node[above]{$x$} (0,3)
    node[left]{$y$} (0,0) node[below left]{$O$};
\end{tikzpicture}
```



```
\begin{tikzpicture}[>=stealth]
\def\r{\sqrt{5}}
\draw[->] (-3,0)--(3,0);
\draw[->] (0,-3)--(0,3);
\draw[dashed] (0,1)--(2,1)--(2,0);
\draw (0,0) circle (\r);
\draw (0,0)--(2,1);
\draw (2,1) circle (0.04);
\draw (3,0) node[above]{$x$} (0,3)
    node[left]{$y$} (0,0) node[below left]{$O$};
\end{tikzpicture}
```

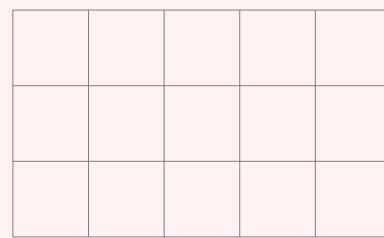


Để dễ dàng cho bạn khi bắt đầu tập vẽ, ta nên tạo lưới tọa độ vùng vẽ để khi xem kết quả ta dễ dàng hình dung ra cách thức và rút kinh nghiệm cho việc sử dụng lệnh bằng cú pháp:

`\draw[step=1,gray,thin] (lower left corner) grid (upper right corner);`
trong đó **lower left corner** là tọa độ điểm gốc trái phía dưới và **upper right corner** là tọa độ điểm gốc phải bên trên của vùng vẽ lưới

Chẳng hạn tạo một lưới tọa độ với vùng vẽ từ $(-2; -1)$ đến $(3; 2)$ ta dùng lệnh như sau:

```
\begin{tikzpicture}
\draw[step=1,gray,very thin]
(-2,-1) grid (3,2);
\end{tikzpicture}
```



I.2.2 Các lệnh cơ bản của Tikz

Với Tikz có các lệnh vẽ cơ bản sau:

1. **Lệnh vẽ đường:** `\draw[tùy chọn] Nội dung vẽ;`

2. **Lệnh đưa đầu bút vẽ đến vị trí nào đó:**

`\path (Tọa độ điểm) Nội dung làm việc ;`

3. **Lệnh tô màu:** `\fill[tùy chọn] miền tô màu;`

Tất nhiên trong Tikz còn một số lệnh khác, nhưng khi vẽ các hình thông thường ta chỉ cần sử dụng ba lệnh trên là đủ.

I.2.3 Khai báo điểm, tham số

Khi vẽ hình ta có những điểm cần sử dụng nhiều lần, các thông số cần sử dụng nhiều làm và các số liệu tính toán. Yêu cầu này buộc ta cần khai báo tên các điểm và giá trị của các tham số (hiểu đơn giản là một cách viết tắt cho những đoạn lệnh dài hơn).

- **Khai báo điểm có tọa độ cho trước:**

`\coordinate[tùy chọn] (Tên điểm) at (Tọa độ);`

`\coordinate[label=above:A] (A) at (1,2);` sẽ khai báo cho ta điểm $A(1; 2)$ và khi vẽ điểm A Tikz sẽ tự động thêm nhãn của điểm là A ở phía trên của điểm.

`\coordinate (A) at (30:2);` sẽ khai báo cho ta điểm $A(30 : 2)$ và không thêm nhãn của điểm.

- **Gán giá trị cho tham số:**

`\def\alpha{3}`: Gán cho tham số a giá trị 3 (hiểu là cho $a = 3$). Khi dùng đến giá trị này trong Tikz chỉ cần gọi `\alpha`.

`\pgfmathsetmacro\alpha{2*sqrt(2)}`: Gán cho tham số a giá trị $2\sqrt{2}$. Chú ý khi gán giá trị cho tham số, dùng `\def` khi gán một giá trị là một số cụ thể (không tính toán thông qua các phép tính) còn nếu buộc phải tính toán qua các phép tính thì dùng `\pgfmathsetmacro`.

I.2.4 Các hàm tính toán số liệu cơ bản

Một số hàm tính toán trong **Tikz**

- **$\sin(\alpha)$:** Trả về giá trị $\sin \alpha$.

- **$\cos(\alpha)$:** Trả về giá trị $\cos \alpha$.

- **$\tan(\alpha)$:** Trả về giá trị $\tan \alpha$.

- **asin (b)**: Trả về giá trị $\arcsin b$.
- **pow (α, a)**: Trả về giá trị a^α .
- **ln (a)**: Trả về giá trị $\ln a$.
- **exp (α)**: Trả về giá trị e^α .
- **log_a (b)**: Trả về giá trị $\log_a b$.
- **sqrt (a)**: Trả về giá trị \sqrt{a} .
- **abs(a)**: Trả về giá trị tuyệt đối của a .
- **int (a)**: Trả về phần nguyên của số a .

Ngoài ra, các phép tính toán thông thường: cộng, trừ, nhân, chia, lũy thừa dùng các ký hiệu tương ứng $+,-,*,/,\wedge$.

I.2.5 Các phép toán đối với tọa độ

- **Cộng tọa độ**: Nếu ta có điểm $A(x_A, y_A)$ và điểm $B(x_B, y_B)$. Khi đó ta có tọa độ điểm $C(x_A + x_B, y_A + y_B)$ bằng cách khai báo:

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

- **Điểm chia tỉ lệ đoạn thẳng (Phép vị tự)**: Ta có hai điểm A và B . Lấy điểm C sao cho $\frac{AC}{AB} = k$ ta khai báo:

`\coordinate (C) at ($(A)!k!(B)$);`

Như vậy, hoàn toàn ta có thể lấy trung điểm đoạn thẳng AB bằng cách

`\coordinate (M) at ($(A).5!(B)$);`

- Phép quay: Muốn lấy điểm B là ảnh của điểm A qua phép quay tâm I , góc quay φ ta có thể dùng:

`\coordinate (B) at ($(I)!1!góc quay:(A)$);`

Chẳng hạn lấy điểm A là ảnh của B qua phép quay tâm I , góc quay 30° ta dùng:

`\coordinate (A) at ($(I)!1!30:(B)$);`

Hơn thế nữa, ta có thể kết hợp phép quay và phép vị tự bằng lệnh:

`\coordinate (A) at ($(I)!k!30:(B)$)`. Khi đó ta được điểm A là ảnh của B qua việc thực hiện liên tiếp phép quay tâm I , góc quay 30° và phép vị tự tâm I tỉ số k .

- **Lấy chân đường vuông góc**: Lấy điểm chân đường vuông góc kẻ từ A đến cạnh BC

`\coordinate (H) at ($(B)!(A)!(C)$);`

- **Giao điểm của hai đường thẳng:** Giao điểm của hai đường thẳng đi qua AB và CD

```
\coordinate (M) at (intersection of A-B and C-D);
```

I.2.6 Hiển thị văn bản trong Tikz

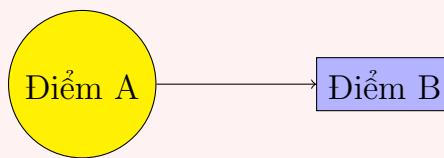
Hiển thị tên điểm khi chưa khai báo label:

```
\node[tùy chọn] (Tên điểm) at (Tọa độ điểm){Nội dung hiển thị};
```

Với lệnh trên, **Tên điểm** là tên khai báo của node (dùng để sau này ta vẽ thêm các đối tượng khác liên quan đến node). **Nội dung hiển thị** nội dung văn bản ta cần hiển thị trong hình vẽ.

Ví dụ:

```
\begin{tikzpicture}
\node[circle,fill=yellow,draw] (A)
  at (0,0){im A};
\node[rectangle,fill=blue!30,draw]
  (B) at (4,0){im B};
\draw[->] (A)--(B);
\end{tikzpicture}
```



Chú ý rằng **coordinate** và **node** có thể dùng bất cứ chỗ nào (trong các lệnh `\draw`, `\fill` hoặc `\path`).

I.2.7 Vẽ Điểm

Ta xác định một điểm trên hình vẽ bằng tọa độ. Trên hệ Oxy ta xác định điểm bằng tọa độ $(x; y)$. Còn trên hệ tọa độ cực ta xác định điểm bằng $(\alpha : r)$.

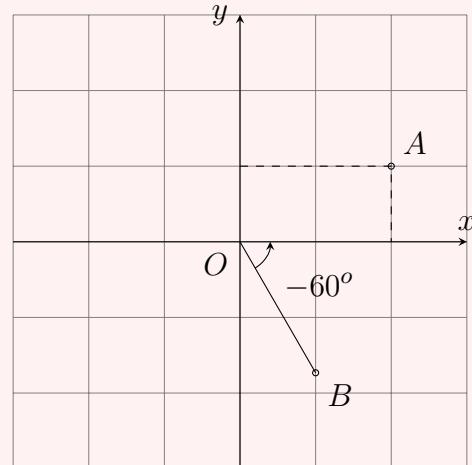
Ví dụ ta muốn vẽ điểm $A(3; -1)$ ta dùng lệnh sau: `\draw (3, -1) circle (1pt);`

Chú ý rằng tọa độ của điểm trong **Tikz** ngăn cách hoành độ và tung độ bởi dấu "phẩy" (,) chứ không phải dấu "chấm phẩy" (;) như ta thường viết trong văn bản. Kết thúc lệnh vẽ luôn bằng dấu "chấm phẩy" (;).

Ta cũng có thể vẽ điểm bằng lệnh `\draw (60:2) circle (1pt);`. Khi đó ta sẽ vẽ được điểm B sao cho $OB = 2$ và góc giữa OB và Ox là 60° .

Hãy xem ví dụ dưới đây:

```
\begin{tikzpicture} [->=stealth]
\draw [step=1,gray,very thin]
(-3,-3) grid (3,3);
\draw [->] (-3,0)--(3,0);
\draw [->] (0,-3)--(0,3);
\draw [dashed] (0,1)--(2,1)--(2,0);
\draw (2,1) circle (0.04);
\draw (3,0) node[above]{$x$} (0,3)
node[left]{$y$} (0,0) node[below left]{$0$};
\draw (-60:2) circle (0.04);
\draw (-60:2) -- (0,0);
\draw (2,1) node[above right]{$A$}
(-60:2) node[below right]{$B$};
\draw [->] (-60:4mm) arc (-60:0:4mm);
\draw (-30:5mm) node[below right]{$-60^\circ$};
\end{tikzpicture}
```



Chú ý rằng lệnh vẽ điểm là `\draw (point) circle (radius);` trong đó **point** là tọa độ điểm và **radius** là bán kính điểm (thường dùng là 1pt hoặc 2pt).

I.2.8 Đoạn thẳng, đường gấp khúc

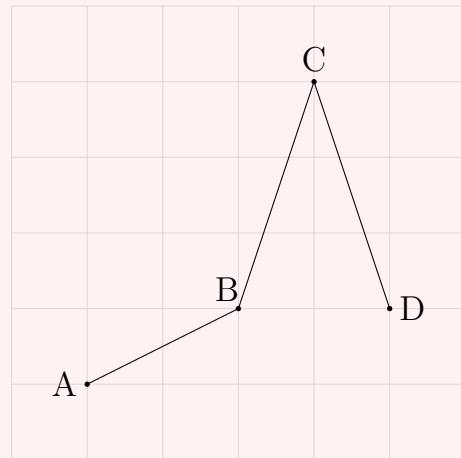
Chúng ta biết rằng để vẽ đoạn thẳng từ A đến B ta dùng lệnh `\draw (A)-(B);`. Tuy nhiên, nếu vẽ một đường gấp khúc từ A đến B rồi đến C tiếp tục đến D thì ta có thể dùng lệnh `\draw (A)-(B)-(C)-(D);` rồi rắc từng đoạn. Như vậy sẽ gây khó khăn vì phải gõ nhiều ký tự. Ta có một số cách sau đây để vẽ một đường gấp khúc:

- **Dùng liên tiếp -:** Ta có thể dùng

```
\draw (A)-(B)-(C)-(D);
```

Cách dùng này thường được dùng khi ta đã khai báo (hoặc biết tọa độ) tất cả 4 điểm A, B, C, D . Ví dụ:

```
\begin{tikzpicture}
\draw[gray!50,thin,opacity=.5]
(-1,-1) grid (5,5);
\path
(0:0) coordinate (A)
(2,1) coordinate (B)
(3,4) coordinate (C)
(4,1) coordinate (D)
;
\draw (A)--(B)--(C)--(D);
\foreach \x/\g in
{A/180,B/120,C/90,D/0}
\fill[black](\x) circle (1pt)
($(\x)+(\g:3mm)$) node{\x};
\end{tikzpicture}
```



- **Dùng $-++$:** Chẳng hạn ta mới có điểm A và B và biết rằng điểm C có tọa độ dạng $C(x_B + x_o; y_B + y_o)$, khi đó ta sẽ dùng lệnh cộng tọa độ:

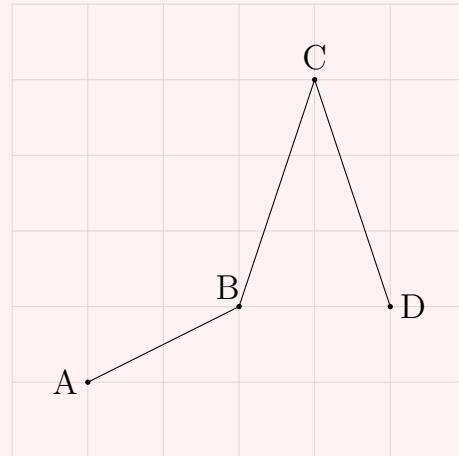
```
\draw (A)-(B)-++(xo,yo);
```

Cái hay của Tikz là việc cộng tọa độ này sẽ được cộng dồn. Sau khi vẽ đến C theo như lệnh trên ta hoàn toàn có thể tiếp tục vẽ thêm đoạn thẳng cộng tiếp tọa độ từ điểm C đến điểm D

Cũng như trên, ta thấy điểm $C(3,4) = (2+1; 1+3)$, như vậy $C(x_B + 1; y_B + 3)$ và tương tự $D(x_C + 1; y_C - 3)$. Ta sẽ dùng lệnh vẽ:

```
\draw (A)-(B)-++(1,3)-++(1,-3);
```

```
\begin{tikzpicture}
\draw[gray!50,thin,opacity=.5]
(-1,-1) grid (5,5);
\path
(0:0) coordinate (A)
(2,1) coordinate (B)
(3,4) coordinate (C)
(4,1) coordinate (D)
;
\draw (A)--(B)---+(1,3)---+(1,-3);
\foreach \x/\g in
{A/180,B/120,C/90,D/0}
\fill[black](\x) circle (1pt)
($(\x)+(\g:3mm)$) node{\x};
\end{tikzpicture}
```

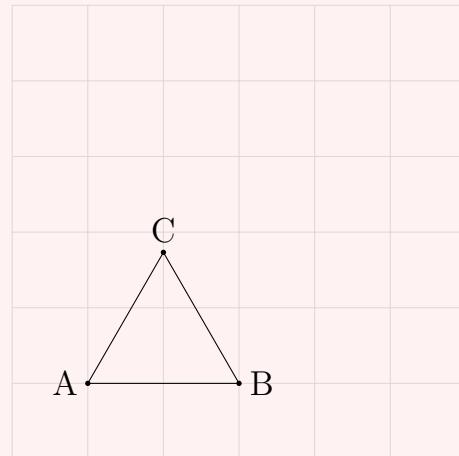


- **Dùng [turn]:** Khi ta vẽ đoạn thẳng từ điểm A đến điểm B rồi, ta muốn vẽ tiếp đến điểm C xác định bằng cách quay một góc φ và độ dài $BC = d$ nào đó ta dùng **[turn]** theo cú pháp:

```
\draw (A)-(B)-([turn]góc:độ dài);
```

Chẳng hạn trong ví dụ vẽ tam giác đều cạnh 2 ta có thể dùng **[turn]** như sau:

```
\begin{tikzpicture}
\draw[gray!50,thin,opacity=.5]
(-1,-1) grid (5,5);
\draw (0,0) coordinate (A)
--(2,0) coordinate(B)
--([turn]120:2) coordinate(C)
--cycle;
\foreach \x/\g in {A/180,B/0,C/90}
\fill[black](\x) circle (1pt)
($(\x)+(\g:3mm)$) node{\x};
\end{tikzpicture}
```



Như vậy, ta hoàn toàn có thể rút ngắn các lệnh vẽ bằng việc kết hợp nhiều cách vẽ đoạn thẳng để có một đường gấp khúc tùy ý.

I.2.9 Đường tròn, Ellipse. Cung tròn, cung Ellipse

Ta biết lệnh vẽ đường tròn là `\draw (Tâm) circle (bán kính);`. Tuy nhiên, để vẽ đường tròn khi biết tâm A và đi qua điểm B thì làm thế nào? Thực tế trong pgf

có lệnh lấy khoảng cách giữa hai điểm, tất nhiên khoảng cách này tính bằng đơn vị pt (điểm ảnh) nên khó trong việc tính toán. Vấn đề tạo lệnh lấy khoảng cách giữa hai điểm bằng thông số phù hợp với đơn vị của Tikz ta sẽ đề cập sau. Nhưng để vẽ đường tròn biết tâm và đi qua một điểm ta có thể dùng lệnh:

```
\draw (I) let \p1=($(I)-(A)$) in circle ({veclen(\x1,\y1)});
```

Khi đó ta có đường tròn tâm I và đi qua điểm A .

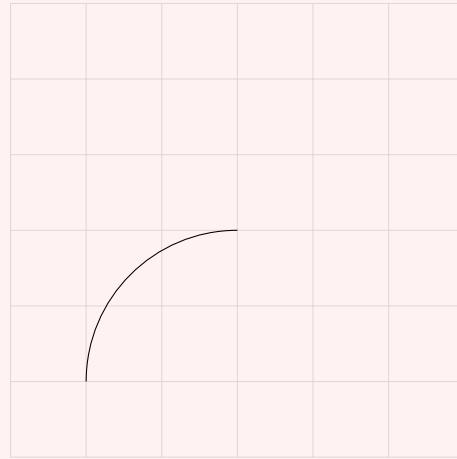
Như vậy ta cũng có lệnh tương tự để vẽ Ellipse có bán trục lớn a , bán trục nhỏ b và tâm $I(x_I; y_I)$ như sau:

```
\draw (I) ellipse ({a} and {b});
```

Ngoài ra, ta có thể vẽ các cung tròn, cung Ellipse nếu biết điểm xuất phát cung, điểm kết thúc cung (tính bằng góc) và các bán kính tương ứng của nó.

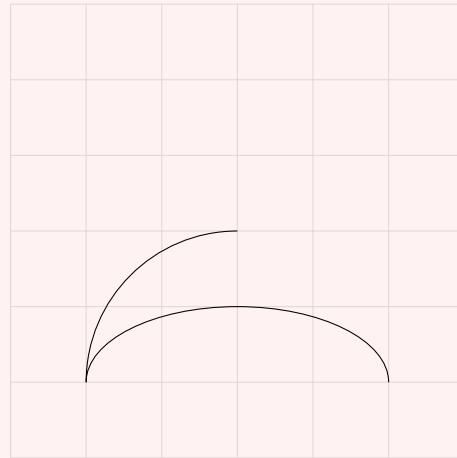
`\draw (A) arc (g1:g2:r);` sẽ vẽ Từ điểm A một cung tròn có góc xuất phát là g1, góc kết thúc là g2 và bán kính r. Ví dụ dưới đây vẽ từ gốc tọa độ một cung tròn xuất phát từ góc 180 độ đến góc kết thúc 90 độ với bán kính bằng 2.

```
\begin{tikzpicture}
\draw[gray!50,thin,opacity=.5]
(-1,-1) grid (5,5);
\draw (0,0) arc (180:90:2);
\end{tikzpicture}
```



Tương tự, nếu dùng lệnh `\draw (A) arc (180:0:{2} and {1});` sẽ được cung Ellipse từ điểm A với góc xuất phát là 180 độ, góc kết thúc 0 độ bán trục lớn là 2 và bán trục nhỏ là 1.

```
\begin{tikzpicture}
\draw[gray!50,thin,opacity=.5]
(-1,-1) grid (5,5);
\draw (0,0) arc (180:90:2);
\draw (0,0) arc (180:0:{2} and {1});
\end{tikzpicture}
```

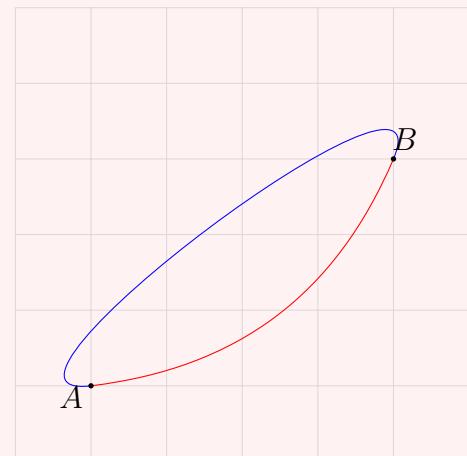


I.2.10 Các dạng đường cong khác

- **to[tùy chọn]**: Vẽ một đường cong từ điểm A đến điểm B với tùy chọn nào đó. Ta có các loại tùy chọn **bend left**, **bend right** hoặc **out**, **in**.

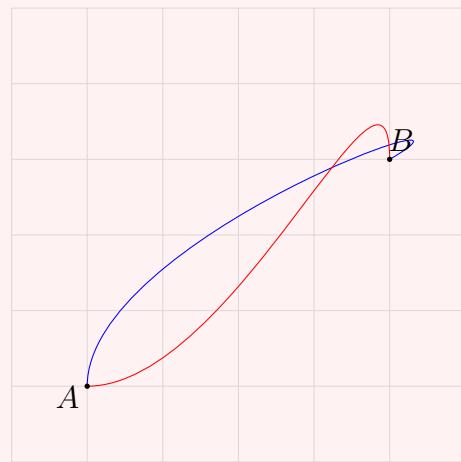
`\draw (A) to[bend left=30] (B);` cho ta một đường cong từ A đến B sao cho góc đi ra của đường cong từ điểm A là 30° và góc đi vào của đường cong tại điểm B là $180^\circ - 30^\circ = 150^\circ$. Đường cong này sẽ cân xứng cả hai phía đối với A và B . Chú ý rằng **bend right** cũng giống như vậy nhưng đường cong sẽ đảo ngược lại. Ta hãy xem ví dụ sau:

```
\begin{tikzpicture}
\draw[gray!50,thin,opacity=.5]
(-1,-1) grid (5,5);
\path (0,0) coordinate (A) (4,3)
coordinate (B);
\draw[blue] (A)to[bend left=150]
(B);
\draw[red] (A)to[bend right=30]
(B);
\foreach \x/\g in
{A/-150,B/60}\fill[black]
(\x) circle (1pt)
($(\x)+(\g:3mm)$)node{$\x$};
\end{tikzpicture}
```



Ta cũng có thể dùng tùy chọn **out**, **in** để xác định các góc ra tại điểm A và góc vào tại điểm B của đường cong. Cách dùng này cho ta thấy các góc ra, vào không cân xứng:

```
\begin{tikzpicture}
\draw[gray!50,thin,opacity=.5]
(-1,-1) grid (5,5);
\path (0,0) coordinate (A) (4,3)
coordinate (B);
\draw[blue] (A)to[out=90,in=30]
(B);
\draw[red] (A)to[out=0,in=90] (B);
\foreach \x/\g in
{A/-150,B/60}\fill[black]
(\x) circle (1pt)
($(\x)+(\g:3mm)$)node{$\x$};
\end{tikzpicture}
```

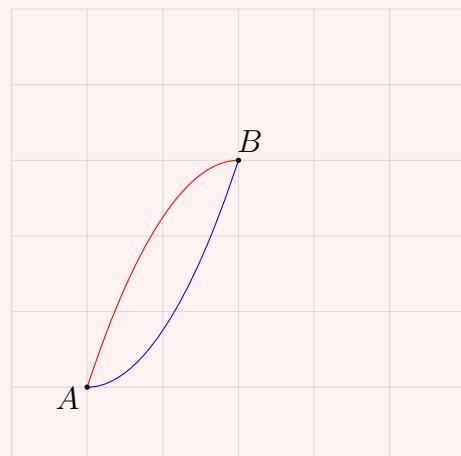


Như vậy, tùy thuộc vào tính chất của đường cong để ta có thể lựa chọn tùy chọn phù hợp cho hình vẽ của mình.

- **Parabola[tùy chọn]**: vẽ một đoạn Parabol từ điểm A đến điểm B . Tùy chọn **bend at end** hoặc không chọn gì. Hoặc ta cũng có thể dùng **parabola bend** với ba điểm A, B, C . Khi đó ta được đoạn Parabol xuất phát từ A , đỉnh B và kết thúc ở C

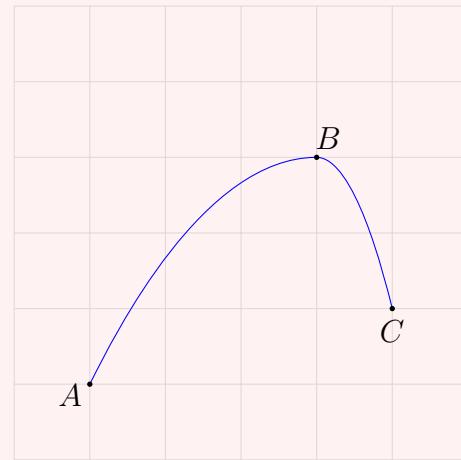
Ta xem ví dụ sau để hiểu rõ tùy chọn **bend at end**:

```
\begin{tikzpicture}
\draw[gray!50,thin,opacity=.5]
(-1,-1) grid (5,5);
\path (0,0) coordinate (A) (2,3)
coordinate (B);
\draw [blue](A) parabola (B);
\draw[red] (A) parabola[bend at
end] (B);
\foreach \x/\g in
{A/-150,B/60}\fill[black]
(\x) circle (1pt)
($(\x)+(\g:3mm)$)node{$\x$};
\end{tikzpicture}
```



Và ví dụ dưới đây để hiểu về **parabola bend**

```
\begin{tikzpicture}
\draw[gray!50,thin,opacity=.5]
(-1,-1) grid (5,5);
\path (0,0) coordinate (A) (3,3)
coordinate (B) (4,1)
coordinate (C);
\draw [blue](A) parabola bend (B)
(C);
\foreach \x/\g in
{A/-150,B/60,C/-90}\fill [black]
(\x) circle (1pt)
($(\x)+(\g:3mm)$)node{$\text{\tiny \x}$};
\end{tikzpicture}
```



- **plot coordinates:** Vẽ đường cong qua nhiều điểm (một đường gấp khúc qua các điểm một cách mềm mại). Ta có thể sử dụng cú pháp:

```
\draw[smooth] plot coordinate{ Tọa độ các điểm};
```

Ví dụ như đường cong sau đây:

```
\begin{tikzpicture}
\draw[gray!50,thin,opacity=.5]
(-1,-1) grid (5,5);
\draw[blue,smooth] plot
coordinates{(0,0) (1,0) (2,1)
(2,2) (3,1) (4,3)};
\end{tikzpicture}
```



- **controls:** Vẽ đường cong qua điểm *A* và *B* với các “lực đẩy”. Ta có cú pháp:

```
\draw (A).. controls (C) and (D) .. (B);
```

Trong cú pháp trên, *A* là điểm bắt đầu, *B* là điểm kết thúc của đường cong. Điểm *C* và *D* chỉ tham gia với tư cách là “lực đẩy” để kéo cho đường nối từ *A* đến *B* có độ cong mà không hiển thị trên hình vẽ.

I.2.11 Lấy một điểm trên đường cong, tiếp tuyến của đường cong

Với một đường cong bất kỳ, ta có thể khai báo để lấy một điểm trên đường cong theo tỉ lệ với cú pháp:

```
\draw Dạng đường cong coordinate[pos=tỉ lệ](A);
```

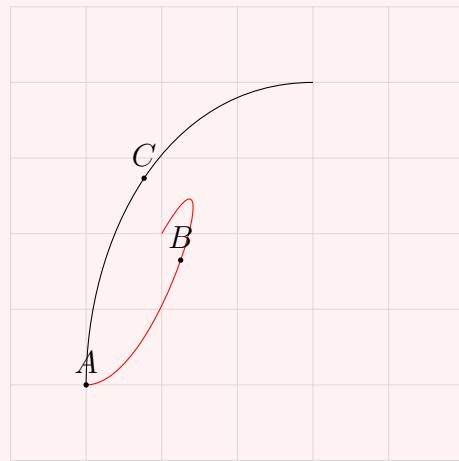
Chẳng hạn lấy một điểm A trên một đường controls như sau:

```
\draw (0,0)..controls +(0:1) and +(60:2)..(1,2) coordinate[pos=.5](A);
```

Khi đó ta lấy được điểm A là điểm chính giữa của đường cong này.

Ví dụ:

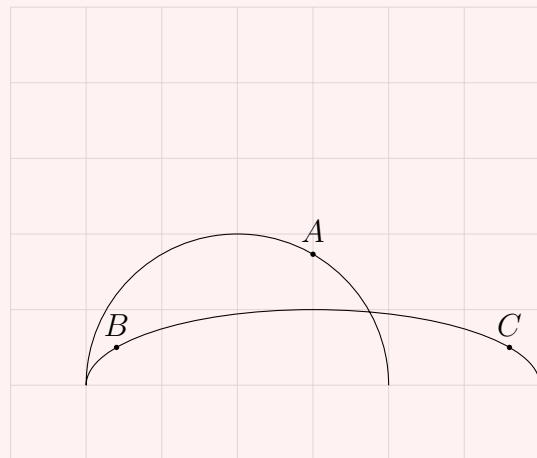
```
\begin{tikzpicture}
\draw[gray!50,thin,opacity=.5]
    (-1,-1) grid (5,5);
\draw[red] (0,0)..controls +(0:1)
    and +(60:2)..(1,2)
    coordinate[pos=.5](B);
\draw (0,0) to[out=90,in=180]
    coordinate[pos=.5](C) (3,4) ;
\foreach \x in {A,B,C}\fill[black]
    (\x) circle (1pt)
    ($(\x)+(90:3mm)$)node{$\x$};
\end{tikzpicture}
```



Tất nhiên lấy một điểm trên các đường có sẵn phương trình (hoặc định dạng quen thuộc như đường tròn hoặc ellipse) thì ta dùng cách khác sẽ tiện lợi hơn rất nhiều bằng việc dùng `\path ... coordinate ...` hoặc `\draw ... coordinate ...`

Chẳng hạn lấy điểm A trên đường tròn và điểm B trên ellipse như sau:

```
\begin{tikzpicture}
\draw[gray!50,thin,opacity=.5]
    (-1,-1) grid (6,5);
\path
    (0,0) arc (180:60:2) coordinate (A)
    (0,0) arc (180:150:{3} and {1})
        coordinate (B)
    (0,0) arc (180:30:{3} and {1})
        coordinate (C)
;
\draw
    (0,0) arc (180:0:2)
    (0,0) arc (180:0:{3} and {1})
;
\foreach \x in {A,B,C}\fill[black]
    (\x) circle (1pt)
    ($(\x)+(90:3mm)$)node{$\x$};
\end{tikzpicture}
```



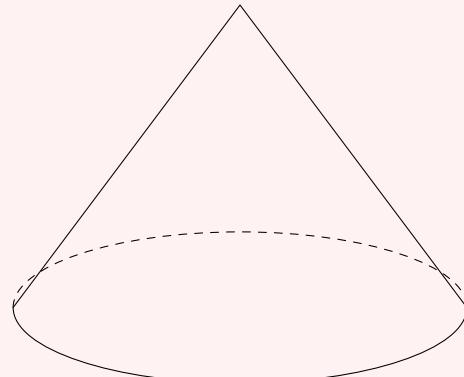
Việc vẽ các đường tiếp tuyến của đường tròn và Ellipse ta có thể căn cứ vào mối quan hệ hình học (và phương pháp tọa độ trong mặt phẳng) để hoạch định cách vẽ cho hợp lý. Tất nhiên Tikz cũng cung cấp thư viện **decoration.markings** để vẽ các đường cát tuyến và tiếp tuyến của đường cong tại một điểm bất kỳ. Nhưng việc sử dụng thư viện này chỉ nên dùng khi quá cần thiết (với dạng đường cong không quen thuộc mà hầu hết ta rất hiếm khi dùng trong các hình vẽ các bài toán, và việc sử dụng thư viện này khá phức tạp và khó nhớ) nên tôi không đề cập ở đây.

Trở lại với đường tròn và Ellipse, để vẽ tiếp tuyến của đường tròn tâm O bán kính r tại điểm M nào đó, ta luôn thấy tiếp tuyến vuông góc với bán kính OM . Như vậy chỉ cần quay tâm O quanh M một góc 90° hoặc -90° được điểm T thì ngay lập tức ta có MT là tiếp tuyến của đường tròn rồi.

Tương tự như vậy, với một Ellipse có bán trục lớn a , bán trục nhỏ b ta có ngay phương trình *Ellipse* dạng $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$ và khi đó tiếp tuyến tại $M(x_0, y_0)$ có dạng $\frac{x \cdot x_0}{a^2} + \frac{y \cdot y_0}{b^2} = 1$. Vậy nếu có hoành độ (hoặc tung độ tiếp điểm) thì ta có được phương trình tiếp tuyến của Ellipse, và như vậy việc vẽ tiếp tuyến chẳng qua chỉ là vẽ đồ thị một hàm số mà thôi.

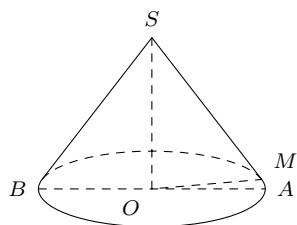
Ví dụ như hình vẽ một hình nón có đáy biểu diễn bởi một Ellipse có bán trục x là $a = 3$, bán trục nhỏ là $b = 1$ và chiều cao $h = 4$. Muốn vẽ chính xác được không đơn giản bởi nếu bình thường bạn sẽ vẽ

```
\begin{tikzpicture}[line
join=round, line cap=round]
\def\aa{3}
\def\bb{1}
\def\hh{4}
\draw[dashed] (180:\aa) arc
(180:0:{\aa} and {\bb});
\draw
(90:\hh)--(180:\aa)arc(180:360:{\aa}
and {\bb})--cycle;
\end{tikzpicture}
```



Nhìn hình vẽ trên ta thấy không được đẹp vì hai đường sinh hai bên cắt đường nét đứt. Hình sẽ không “thật” lắm. Muốn hình được “thật” hơn, bạn phải tính toán sao cho các đường sinh tiếp xúc với Ellipse. Ta sẽ đặt ra bài toán vẽ tiếp tuyến của Ellipse đi qua đỉnh của hình nón.

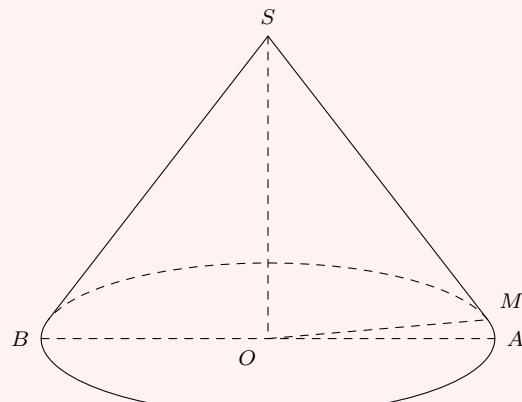
Bạn vẽ phác hình ra để phân tích và tìm cách vẽ:



Trong hình vẽ trên, ta thấy điểm $A(a; 0)$, $B(-a; 0)$, $S(0; h)$, $O(0; 0)$ và điểm $M(x_o; y_o)$. Vấn đề là xác định được tọa độ điểm M . Chú ý rằng trong hình học tọa độ phẳng nếu điểm, phương trình Ellipse là $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$ và phương trình tiếp tuyến của Ellipse tại $M(x_o; y_o)$ là $\frac{x \cdot x_o}{a^2} + \frac{y \cdot y_o}{b^2} = 1$. Vì tiếp tuyến đi qua điểm $S(0; h)$ nên ta thay tọa độ điểm S vào phương trình tiếp tuyến sẽ suy ra được $y_o = \frac{b^2}{h}$. Từ đó ta có thể tính được x_o .

Mặt khác, trong hệ tọa độ cực, tọa độ điểm $M(\varphi : r)$ với $\varphi = \widehat{AOM}$ thì hoành độ và tung độ điểm M được tính theo công thức:
$$\begin{cases} x_o = a \cos \varphi \\ y_o = b \sin \varphi \end{cases}$$
. Từ đây ta có thể tính góc $\varphi = \arcsin \frac{y_o}{b} = \arcsin \frac{b}{h}$ để lấy góc vẽ các cung Ellipse cần thiết. Như vậy ta có thể hoàn thiện code vẽ một cách linh hoạt và đơn giản:

```
\begin{tikzpicture}[line
join=round, line
cap=round, font=\scriptsize]
\def\aa{3}
\def\bb{1}
\def\hh{4}
\pgfmathsetmacro\g{asin(\bb/\hh)}
\pgfmathsetmacro\xo{\aa *cos(\g)}
\pgfmathsetmacro\yo{\bb *sin(\g)}
\draw[dashed]
(\xo,\yo) arc (\g:180-\g:\aa and
{\bb})
(180:\aa)node[left]{$B$} --(0:\aa)
node[right]{$A$} (90:\hh)
node[above]{$S$} --(0:0)
node[below left]{$O$}
--(\xo,\yo)node[above right]{$M$}
;
\draw (90:\hh)--(-\xo,\yo) arc
(180-\g:360+\g:\aa and
{\bb})--cycle;
\end{tikzpicture}
```



Với code vẽ trên, bạn có thể thay đổi tùy ý bán trục x, bán trục y và chiều cao h

của hình nón thì Tikz luôn tính được cho ta các điểm M một cách linh hoạt và tự động. Hình nón sẽ đẹp như “thật”. Ngược lại, nếu hình nón cho điểm tiếp xúc M (biết góc φ) thì ta thay đổi phần khai báo h bằng φ và lại tính h bằng công thức $h = \frac{b}{\sin \varphi}$ là xong.

I.2.12 Giao điểm của các đường

Trong vẽ hình khoa học, đặc biệt là các bài toán Hình học, việc sử dụng giao điểm của các đường là thường gặp. Với giao của hai đường thẳng (đoạn thẳng) tôi đã trình bày ở phần các phép toán tọa độ. Với hai đường bất kỳ ta cần sử dụng thư viện **intersections**.

Để sử dụng thư viện này, ta phải gọi thư viện (ngay sau khi khai báo gói **Tikz** và trước `\begin{document}`) bằng cú pháp:

```
\usetikzlibrary{intersections}
```

Mỗi đường sẽ được đặt một cái tên cho đường bằng

```
\path[name path=tên đường] Dạng đường ;
```

hoặc `\draw[name path=tên đường] Dạng đường ;`. Chú ý rằng nếu sử dụng `\path` thì **Tizk** chỉ đặt tên đường mà không vẽ ra, còn nếu sử dụng `\draw` thì đường được vẽ ra trên hình vẽ.

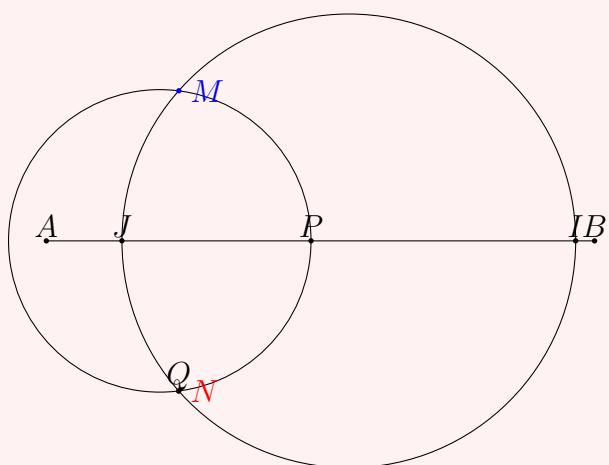
Khi đó nếu có hai đường đã được đặt tên là $c1$ và $c2$ bằng cú pháp:

```
\path[name intersections={of=c1 and c2}] (intersection-1) coordinate (A);
```

Với những cặp đường chỉ có một giao điểm thì như vậy, nếu cặp đường có nhiều giao điểm hơn thì ta sử dụng `intersection-1`, `intersection-2`, ... cho đến hết sẽ khai báo được tất cả các điểm.

Bạn có thể tham khảo code lấy giao điểm 2 đường tròn như dưới đây:

```
\begin{tikzpicture}
\draw[name path=ab] (-1.5,0)
    coordinate (A) --(5.75,0)
    coordinate (B);
\draw[name path=circ1] (0,0) circle
    (2);
\draw[name path=circ2] (2.5,0)
    circle (3);
\path[name intersections={of=circ1
    and circ2}]
(intersection-1) coordinate (M)
(intersection-2) coordinate (N)
;
\path[name intersections={of=circ1
    and ab}]
(intersection-1) coordinate (P)
(intersection-2) coordinate (Q)
;
\path[name intersections={of=circ2
    and ab}]
(intersection-1) coordinate (I)
(intersection-2) coordinate (J)
;
\fill[blue] (M)node[right]{$M$}
    circle (1pt);
\fill[red] (N)node[right]{$N$}
    circle (1pt);
\foreach \x in
{A,B,P,Q,I,J}\fill[black] (\x)
circle (1pt)+(90:2mm)node{$\x$};
\end{tikzpicture}
```



Bạn để ý thấy điểm Q không hiển thị đúng chỗ. Thật vậy. Vì đường tròn circ1 chỉ cắt đoạn AB tại một điểm nên điểm Q Tikz sẽ lấy ngẫu nhiên. Điều này rất quan trọng khi bạn cần tìm giao điểm của các đường bằng thư viện **intersections** bạn cần biết số giao điểm “thực” của hai đường đó. (giao điểm nhìn thấy).

I.2.13 Môi trường scope, clip, node và pic

- **Môi trường scope:**

Tikz cung cấp môi trường **scope** với cú pháp

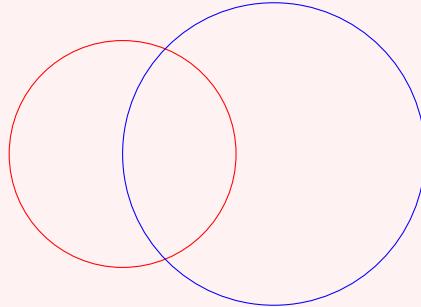
```
\begin{scope}
...
\draw ...
...
\end{scope}
```

Trong môi trường **scope** Mọi lệnh vẽ, lệnh khai báo chỉ có tác dụng trong môi trường này, khi thoát ra khỏi **scope** thì các điểm đã khai báo, các giá trị đã gán, tên đường đã gán ... trong **scope** không còn tác dụng. Vì vậy cần sử dụng linh hoạt môi trường này.

- **Lệnh clip**

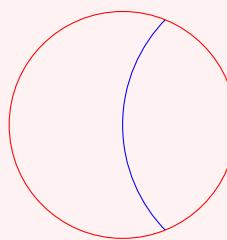
Đi đôi với **scope** là lệnh **\clip** **Dạng đường kín;**. **clip** dùng để cắt các phần thừa khi vẽ hình để hình vẽ đúng theo ý vẽ. Chẳng hạn cần vẽ nửa một cung tròn của đường tròn màu xanh nằm trong đường màu đỏ như sau:

```
\begin{tikzpicture}
\draw[red] (0,0) circle (1.5);
\draw[blue] (2,0) circle (2);
\end{tikzpicture}
```



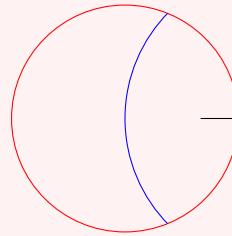
Ta có thể clip như sau:

```
\begin{tikzpicture}
\draw[red] (0,0) circle (1.5);
\clip (0,0) circle (1.5);
\draw[blue] (2,0) circle (2);
\end{tikzpicture}
```



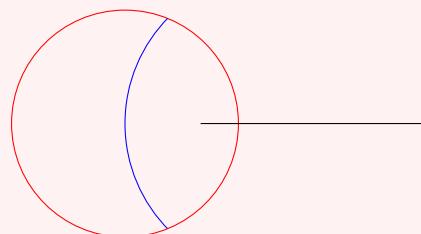
Để ý rằng lệnh **clip** sẽ có tác dụng từ vị trí đặt nó cho đến **\end{tikzpicture}** nên ta cần kết hợp **scope**. Chẳng hạn trong hình vẽ trên ta lại muốn vẽ tiếp đoạn thẳng nằm bên ngoài đường tròn màu đỏ nhưng nếu để bình thường **clip** ta sẽ không thấy hiển thị.

```
\begin{tikzpicture}
\draw[red] (0,0) circle (1.5);
\clip (0,0) circle (1.5);
\draw[blue] (2,0) circle (2);
\draw (1,0)--(4,0);
\end{tikzpicture}
```



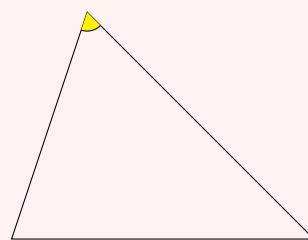
Vậy thì ta phải đưa việc vẽ cung tròn đó vào trong môi trường **scope** để có thể tiếp tục vẽ thêm

```
\begin{tikzpicture}
\draw[red] (0,0) circle (1.5);
\begin{scope}
\clip (0,0) circle (1.5);
\draw[blue] (2,0) circle (2);
\end{scope}
\draw (1,0)--(4,0);
\end{tikzpicture}
```



Kỹ thuật này rất hữu dụng trong việc vẽ các ký hiệu góc trong bài toán Hình học. Chẳng hạn muốn thể hiện \widehat{ABC} trong tam giác ABC ta có thể làm như sau:

```
\begin{tikzpicture}
\draw (0,0) coordinate (B)
    --(4,0) coordinate (C) --
    (1,3) coordinate (A) --cycle;
\begin{scope}
\clip (B)--(A)--(C);
\draw[fill=yellow] (A) circle (.25);
\end{scope}
\end{tikzpicture}
```



- **node**

Như tôi đã trình bày, lệnh `\node ... at ... {Đối tượng hiển thị};` hoặc dùng kép kiểu như

`\draw ... node{đối tượng hiển thị};` được dùng khá nhiều khi vẽ hình. Ở đây ta tìm hiểu thêm về **node** để sử dụng cho thật sự linh hoạt.

Đối tượng hiển thị trong các lệnh trên (gọi tắt là node) được Tikz khá thoái mái. Nó có thể là bất cứ lệnh nào trong L^AT_EX chứ không riêng gì bfseries Tikz.

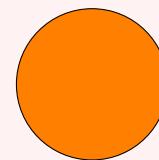
Chẳng hạn bạn muốn vẽ một hình bài toán thực tế có hình cây dừa, mà việc vẽ cây dừa tốn quá nhiều thời gian, bạn có thể include hình png cây dừa đó vào trong hình vẽ Tikz.

```
\begin{tikzpicture}
\draw (-3,0) -- (3,0);
\path (0,0) node[above]
{\includegraphics[scale=.3]
{caydua}};
\end{tikzpicture}
```



Hay nhất là trong **node** có thể đưa vào bất cứ thứ gì mà L^AT_EX chấp nhận được, kể cả môi trường **tikzpicture**

```
\begin{tikzpicture}
\draw (-3,0) -- (3,0);
\path (0,3) node{
\begin{tikzpicture}
\draw[fill=orange] (0,0) circle
(1);
\end{tikzpicture}
};
\end{tikzpicture}
```



• pic

Tương tự như **node**, **pic** cũng là một trong những dạng có sẵn của **Tikz**. Sử dụng cú pháp:

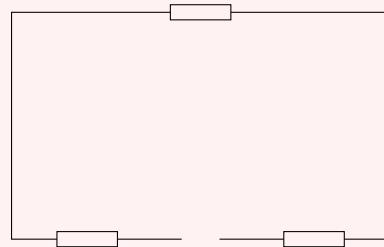
```
\draw (tọa độ) pic[tên pic];
```

Khi đó ta sẽ có hình ảnh của **pic** vào đúng tọa độ đã chỉ ra. Vấn đề là các pic này ở đâu? tên pic là gì? Ta không cần quan tâm quá nhiều đến các pic có sẵn của Tikz vì chủ tâm của người tạo ra Tikz là để người dùng tự tạo pic khi cần thiết. Chẳng hạn bạn muốn vẽ một mạch điện, mỗi điện trở là một hình chữ nhật. Trong khi mạch điện của ta có nhiều điện trở, mỗi lần vẽ điện trở lại phải vẽ một hình chữ nhật thì thật vất vả. Ta sẽ tạo pic bằng cú pháp:

```
\tikzset{dientro/.pic={\draw(-.4,-.1)rectangle (.4,.1);}}
```

Sau đó ta sử dụng pic này thoải mái như sau:

```
\begin{tikzpicture}
\tikzset{dientro/.pic={
    \draw(-.4,-.1) rectangle (.4,.1);}}
\draw
(0,0) pic[local bounding box=R1]
{dientro}
(3,0) pic[local bounding box=R2]
{dientro}
(1.5,3) pic[local bounding
box=R3] {dientro}
;
\draw (1.75,0) --(R2) --(4,0)
--(4,3) --(R3) --(-1,3)
--(-1,0) --(R1) --(1.25,0);
\end{tikzpicture}
```

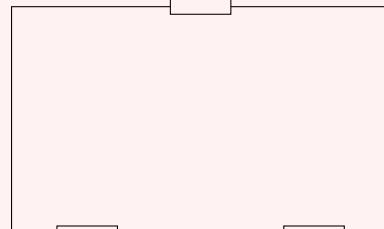


Nhân đây, để vẽ mạch điện tôi cũng đề cập đến một cách vẽ hai cạnh liên tiếp của một hình chữ nhật xác định bởi hai đỉnh đối nhau (thường dùng khi giống tọa độ trong đồ thị hàm số và vẽ mạch điện) như sau:

```
\draw (A)-|(B); \draw (A)|-(B);
```

Chẳng hạn để rút ngắn lệnh vẽ mạch điện trên, tôi sẽ dùng:

```
\begin{tikzpicture}
\tikzset{dientro/.pic={
    \draw(-.4,-.1) rectangle (.4,.1);}}
\draw
(0,0) pic[local bounding box=R1]
{dientro}
(3,0) pic[local bounding box=R2]
{dientro}
(1.5,3) pic[local bounding
box=R3] {dientro}
;
\draw (1.75,0) -|(R2) --(4,0)
|-(R3) -|(R1) --(1.25,0);
\end{tikzpicture}
```



I.3 Các tùy chọn thường dùng

I.3.1 Tùy chọn chung

Tùy chọn của **Tikz** là các tham số đầu vào nằm trong cặp [...] ở ngay sau mỗi môi trường, lệnh vẽ. Với tất cả các lệnh vẽ, môi trường ta có một số tùy chọn chung như sau:

- **color=**: Lựa chọn màu cho đối tượng. Với tùy chọn này thường thì ta không cần nhập **color= tên màu** mà ta chỉ cần nhập **tên màu** là Tikz cũng tự hiểu được. Tên màu có thể là **yellow**, **blue**, **red**, **orange**, **pink**, ... Hơn nữa, việc xác định màu có thêm tùy chọn **red!50** ta hiểu là độ đỏ 50%, **blue!20** là độ xanh 20% ... Ngoài ra có có một số tùy chọn **left color=**, **right color=**, **bottom color=**, **top color=**, **ball color=** ...
- **rotate=**: Quay đối tượng quanh gốc tọa độ một góc nào đó, đơn vị góc ở đây là “độ” chứ không dùng **radian**.
- **shift=**: Tịnh tiến đối tượng theo một vec tơ nào đó.

Cụ thể hơn ta có **xshift=** và **yshift=** là cách tịnh tiến đối tượng theo chiều ngang hoặc chiều dọc. Chú ý khi dùng **xshift** hoặc **yshift** thì các thông số đưa vào phải có đơn vị độ dài (pt, mm, cm, ...)

- **scale=**: Tỉ lệ hiển thị đối tượng.

Tương tự như trên ta cũng có **xscale=** lấy đối xứng đối tượng qua trực tung và **yscale** lấy đối xứng đối tượng qua trực hoành.

- **font=**: Chọn cỡ chữ hiển thị trong hình vẽ (tùy chọn này thường dùng cho các môi trường **tikzpicture** và **scope** để cỡ chữ thống nhất trong cùng một hình vẽ)
- **line join**, **line cap**: Tùy chọn này thường dùng cho môi trường **tikzpicture**. **line join=round** để các đoạn gấp khúc mềm mại hơn, **line cap=round** để các đầu mút đoạn thẳng được bo tròn.
- **line width=**: Độ dày nét vẽ, với tùy chọn này nếu đơn vị là điểm ảnh (pt) thì thông số đầu vào không cần đơn vị. Nếu ta muốn dùng đơn vị độ dài khác thì cần nhập thông số đi kèm đơn vị (mm, cm, ...)
- **smooth**: Tạo độ trơn cho nét vẽ, thường dùng với vẽ đồ thị và đi kèm lệnh **\draw** hoặc **\fill**
- **samples=**: Ta hiểu rằng vẽ đường cong trong **Tikz** hoặc bất cứ phần mềm nào bản chất vẫn là chia nhỏ đường cong thành các đoạn thẳng để vẽ liên tiếp các đoạn thẳng này. **samples** quy định điều đó. **samples** càng lớn thì số điểm chia càng nhiều, khi đó đường cong càng đẹp hơn. Tuy nhiên đi kèm với nó là việc tính

toán của **Tikz** sẽ tăng lên làm chậm quá trình biên dịch. Khuyên dùng với thông số **samples=100** là vừa phải (1 đơn vị được chia thành 100 điểm nhỏ).

- **->**: Dùng vẽ các đoạn thẳng (đoạn cong) có định hướng (mũi tên ở cuối đoạn). Các kiểu mũi tên thường dùng là **stealth**, **latex**,
- **opacity=**: Tùy chọn này chỉ định độ hòa trộn giữa các lớp đối tượng (layer) trong một hình vẽ. Giống như các phần mềm làm việc với hình ảnh khác, **Tikz** vẽ hình theo các lớp, lớp sau đè lớp trước, **opacity** sẽ hòa trộn các đối tượng vào với nhau. Thường dùng với **\fill**
- **double**: Tùy chọn vẽ nét đôi trong lệnh vẽ **\draw**. Ta hoàn toàn có thể tùy chỉnh khoảng cách giữa các nét đôi bằng tùy chọn **double distance**
- **rounded corners=**: Tùy chọn làm mềm độ gấp khúc của lệnh **\draw** và **\fill**

Ta hoàn toàn có thể dùng cùng lúc các tùy chọn này, Tikz sẽ thực hiện lần lượt từng tùy chọn cho đến hết khi vẽ hình. Chú ý với các tùy chọn về các phép quay, tịnh tiến, vị tự (**rotate**, **shift**, **scale**) khi dùng trong lệnh **\draw** hoặc **\fill** sẽ không thực hiện nếu các điểm đã được định nghĩa từ trước.

I.3.2 Với node, pic

Với **pic** và **node** ngoài những tùy chọn chung còn có một số tùy chọn riêng.

pic[local bounding box=A] ... là tùy chọn để khai báo vị trí đặt **pic** là *A*, dùng để vẽ các đường nối các điểm một cách hợp lý.

sloped: Dùng với **node** khi ta node theo một đường nào đó (có thể cong hoặc thẳng). Khi đó văn bản trong **node** sẽ nghiêng theo độ nghiêng của đường.

midway: Dùng với **node** như trên, vị trí đặt của node sẽ ở giữa đường đó. Thường dùng khi đặt các khoảng cách giữa hai điểm.

Ngoài ra, còn rất nhiều tùy chọn khác trong các lệnh của **Tikz** mà tôi chưa liệt kê ở đây, bởi nó ít được dùng. Bạn hoàn toàn có thể tra cứu pgfmanual.pdf có đầy trên internet.

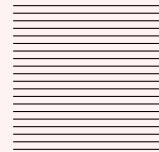
I.3.3 Vòng lặp foreach

Vòng lặp **foreach** thực chất là lệnh của **LATEX**. Nhưng nó rất hữu dụng khi vẽ hình bằng **Tikz**. Ta có thể hiểu vòng lặp này như sau:

\foreach tên biến in {tập hợp biến} {lệnh thực thi;}

Việc sử dụng **foreach** tiết kiệm cho ta việc gõ quá nhiều dòng lệnh giống nhau, chỉ thay đổi một vài thông số. Chẳng hạn bạn muốn vẽ 20 đoạn thẳng song song và cùng có độ dài bằng 2 từ các điểm trên trực tung, có tung độ tăng dần .1 đơn vị. Nếu bạn vẽ lần lượt thì phải gõ đúng 20 dòng lệnh **\draw (...) -+(0:2);** Nhưng với foreach bạn chỉ cần một dòng lệnh, Tikz (thực chất là **LATEX**) sẽ thay bạn làm việc đó:

```
\begin{tikzpicture}
\foreach \i in {1,...,20}\draw
(0,.1*\i)---+(0:2);
\end{tikzpicture}
```



Ở đây, ta thấy hoành độ luôn là 0, tung độ sẽ tăng dần, với $i = 1$ thì tung độ là .1, với $i=2$ thì tung độ là .2, ... Bạn thấy cú pháp chõ **lệnh thực thi** ở trên được đặt trong ngoặc nhọn còn trong ví dụ thì không? Nếu như chỉ cần làm một việc duy nhất (một lệnh duy nhất) thì không cần cặp ngoặc nhọn đó, nhưng nếu từ hai lệnh trở lên bạn cần phải có ngoặc nhọn, nếu không **foreach** chỉ thực thi cho bạn một lệnh đầu tiên liền sát nó mà thôi.

foreach còn hơn thế, ta có thể dùng vòng lặp này lồng nhau để vẽ (hiển thị một bảng). Chẳng hạn:

```
\begin{tikzpicture}
\foreach \i in {1,...,7}{
\foreach \j in {1,...,5}
\path (\i,\j) node{$a_{\i \j}$};
}
\end{tikzpicture}
```

a_{15}	a_{25}	a_{35}	a_{45}	a_{55}	a_{65}	a_{75}
a_{14}	a_{24}	a_{34}	a_{44}	a_{54}	a_{64}	a_{74}
a_{13}	a_{23}	a_{33}	a_{43}	a_{53}	a_{63}	a_{73}
a_{12}	a_{22}	a_{32}	a_{42}	a_{52}	a_{62}	a_{72}
a_{11}	a_{21}	a_{31}	a_{41}	a_{51}	a_{61}	a_{71}

Quả là tiện lợi, nếu không dùng **foreach** bạn sẽ phải dùng 35 node với 35 tọa độ và 35 văn bản. Tất nhiên, khi dùng **foreach** bạn cần chú ý đến những công thức mang tính tổng quát. Chẳng hạn, cũng bảng trên nhưng tôi muốn hiển thị $a_{ij} = \beta + j$ thì tôi sẽ làm:

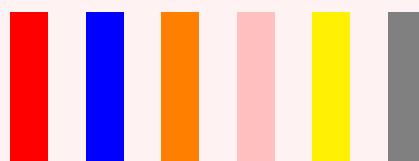
```
\begin{tikzpicture}
\foreach \i in {1,...,7}{
\foreach \j in {1,...,5}
\pgfmathsetmacro{\ht}{int(\i+\j)}
\path (\i,\j) node{\ht};
}
\end{tikzpicture}
```

6	7	8	9	10	11	12
5	6	7	8	9	10	11
4	5	6	7	8	9	10
3	4	5	6	7	8	9
2	3	4	5	6	7	8

Tiếp tục nghiên cứu về **foreach**. Trong cái **tập hợp biến** ta có thể có nhiều cách đưa vào. Chẳng hạn với giá trị biến là các con số có quy luật (chẳng hạn các số cách đều nhau như ở ví dụ trên) thì ta nhập vào dạng **.1,.3,...,5** sẽ được danh sách là .1, .3, .5, .7 ... 5 sao cho các biến cách nhau .2. Còn nếu là các số nguyên thì chỉ cần nhập vào số đầu tiên và số cuối cùng như trong ví dụ. Nếu các giá trị biến không có quy luật hoặc định dạng không phải các con số thì sao? Ta buộc phải liệt kê các giá trị biến trong đó.

Vậy còn tên biến? Bạn có thể đặt tên biến một cách thoải mái, hơn nữa ta có thể dùng nhiều biến cùng lúc như ví dụ dưới đây:

```
\begin{tikzpicture}
\foreach \x/\col in {1/red, 2/blue,
3/orange, 4/pink, 5/yellow,
6/gray} \fill[\col] (\x,0)
rectangle +(.5,2);
\end{tikzpicture}
```



I.3.4 Tạo macro

Macro là gì? Nó là một lệnh do người dùng tạo ra, bản chất là để thực hiện một tập hợp các lệnh của **LATEX** theo kiểu “gõ tắt” trên word bằng Unikey mà thôi. Cú pháp tạo lệnh:

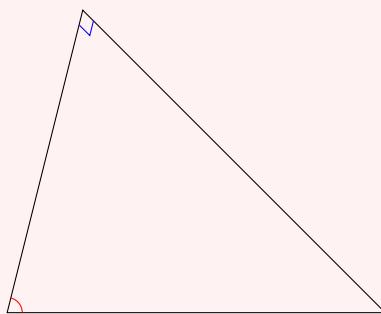
```
\def \name{Các lệnh thực thi}
```

Như vậy, bạn có thể đặt **name** thoải mái, miễn sao đừng trùng với các lệnh có sẵn của **LATEX** và **Tikz** để khỏi gây lỗi.

Phần **lệnh thực thi** bạn có thể đưa vào bất cứ lệnh vẽ nào của **Tikz**. Và khi nào sử dụng đến tập hợp các lệnh đó bạn chỉ việc `\name` là xong.

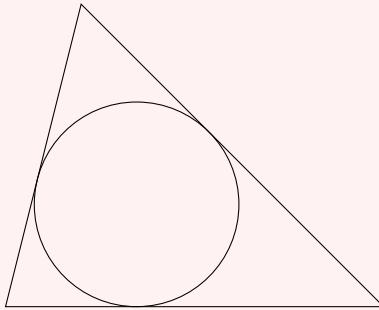
Tất nhiên nếu chỉ có thể thì tạo macro cũng không tiện lợi mấy. Macro còn có thể đưa thông số đầu vào và thông số đầu ra. Chẳng hạn với macro tạo lệnh vẽ ký hiệu góc vuông và ký hiệu góc thường mà tôi đã từng tạo:

```
\begin{tikzpicture}
\def\khvuong(#1,#2,#3){
\path
($(#2)!2mm!(#1)$) coordinate (#2#1)
($(#2)!2mm!(#3)$) coordinate (#2#3)
;
\draw[blue] (#2#1)
--($(#2#1)+(#2#3)-(#2)$)--(#2#3);
}
\def\khgoc(#1,#2,#3){
\begin{scope}
\clip (#1)--(#2)--(#3);
\draw[red] (#2) circle (2mm);
\end{scope}
}
\draw (0,0) coordinate(B)
--(5,0) coordinate (C)
--(1,4) coordinate (A)
--cycle;
\khvuong(B,A,C)
\khgoc(A,B,C)
\end{tikzpicture}
```



Hơn nữa, **macro** còn lo cho ta cả thông số đầu ra. Chẳng hạn ta muốn tìm tâm nội tiếp của một tam giác, ta cũng có thể xây dựng một macro như sau:

```
\begin{tikzpicture}
\def\tamnoitiep(#1,#2,#3)(#4){
\path
($($(#2)!2mm!(#1$))!0.5!($(#2)!2mm!(#3$)$)
coordinate (#2a)
($($(#1)!2mm!(#2$))!0.5!($(#1)!2mm!(#3$)$)
coordinate (#1a)
(intersection of #2-- #2a and
#1--#1a) coordinate (#4);
}
\draw (0,0) coordinate(B)
--(5,0) coordinate (C)
--(1,4) coordinate (A)
--cycle;
\tamnoitiep (A,B,C)(I)
\path ($(A)!(I)!(B)$) coordinate
(M);
\draw (I) let \p1=($(I)-(M)$) in
circle ({veclen(\x1,\y1)});
\end{tikzpicture}
```



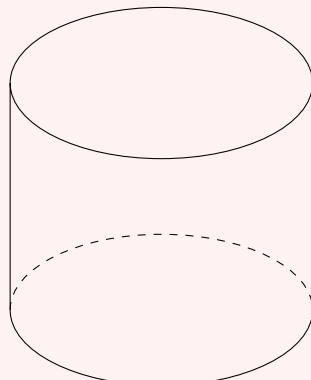
Như vậy, bạn có thể tự tạo ra các macro để vẽ theo ý bạn và mạch tư duy hình học của bạn để vẽ mà không mất quá nhiều thời gian. Về bản chất, gói lệnh **tkz-euclide** được xây dựng với các ý tưởng như thế này (người ta gọi đó là các gói thứ cấp của Tikz). Với bộ môn Vật lý hoặc Hóa học cũng có những gói thứ cấp của Tikz như vậy.

I.3.5 Vẽ hình không gian (giả 3d)

Khi vẽ hình không gian (bản chất là vẽ 2d hình chiếu song song của một hình không gian trên mặt phẳng), ta thường có một tham số đầu vào là góc nghiêng (góc nhìn hình). Bạn cần nhắc lại về các quy tắc biểu diễn của một hình không gian trên mặt phẳng. Bạn hãy nhớ đến các quy tắc về tính song song, tỉ lệ các đoạn thẳng song song được giữ nguyên.Thêm nữa, bạn cần nhớ đến quy tắc nét liền, nét đứt.

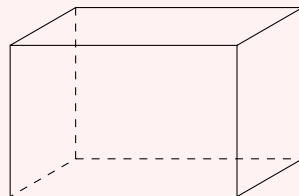
Chẳng hạn bạn muốn biểu diễn một hình tròn, bạn sẽ vẽ một Ellipse. Vậy thì bạn hoàn toàn có thể biểu diễn một hình trụ đơn giản như sau:

```
\begin{tikzpicture}
\def\aa{2}
\def\bb{1}
\def\hh{3}
\draw[dashed] (180:\aa) arc
(180:0:{\aa} and {\bb});
\draw (-\aa,\hh)--(-\aa,0) arc
(180:360:{\aa} and {\bb})--(\aa,\hh)
(90:\hh) ellipse ({\aa} and {\bb});
\end{tikzpicture}
```



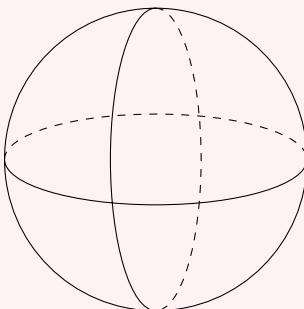
Bạn muốn một hình hộp chữ nhật?

```
\begin{tikzpicture}
\def\aa{3}
\def\bb{1}
\def\gg{30}
\def\hh{2}
\path
(0:0) coordinate (A)
--+(\gg:\bb) coordinate (B)
--+(0:\aa) coordinate (C)
--+(\gg-180:\bb) coordinate (D)
\foreach \x in {A,B,C,D}{
    ($(\x)+(90:\hh)$) coordinate (\x')
};
\draw[dashed] (B')--(B)--(A)
(B)--(C);
\draw
(A)--(D)--(D')--(A')--cycle
(A')--(B')--(C')--(D')
(D)--(C)--(C')
;
\end{tikzpicture}
```



Bạn muốn một hình cầu:

```
\begin{tikzpicture}
\def\r{2}
\draw[dashed]
(180:\r) arc (180:0:{\r} and
{.3*\r})
(90:\r) arc (90:-90:{.3*\r} and
{\r})
;
\draw
(0:0) circle (\r)
(180:\r) arc (180:360:{\r} and
{.3*\r})
(90:\r) arc (90:270:{.3*\r} and
{\r})
;
\end{tikzpicture}
```



II Bảng xét dấu - Bảng biến thiên - Đồ thị hàm số

II.1 Bảng xét dấu

Ta hãy hình dung bảng xét dấu gồm 2 dòng, một dòng có giá trị của biến x , một dòng có giá trị là dấu của $f(x)$ (hoặc $f'(x)$) trong bài khảo sát và vẽ đồ thị hàm số. Ta có thể dùng **foreach** để vẽ bảng này một cách rất đơn giản. Chẳng hạn một bảng như sau:

```
\begin{tikzpicture} [scale=.75]
\draw[thin,opacity=.5] (-1,-2) grid
(9,1);
\foreach \i/\x in
{0/x,2/-\infty,4/-1,6/2,8/+\infty}
\path (\i,.5) node{$\x$};
\foreach \i/\x in
{0/f(x),3/-,4/0,5/+,6/0,7/-}
\path (\i,-.5) node{$\x$};
\draw (-.5,0)--(8.5,0)
(.75,1)--(.75,-1);
\end{tikzpicture}
```

x	$-\infty$	-1	2	$+\infty$
$f(x)$	+	0	0	+

Tất nhiên trong code trên bạn đã hình dung trong đầu các vị trí các điểm cần đặt giá trị. Công việc này bạn hoàn toàn có thể làm đúng như việc bạn vẽ bằng bút trên giấy vậy. Một cách khác là bạn có thể sử dụng gói **tkz-tab** để vẽ mà tôi sẽ trình bày ở phần sau đây.

II.2 Bảng biến thiên

Cấu trúc của bảng biến thiên gồm ba dòng. Dòng giá trị x , dòng giá trị đạo hàm $f'(x)$ và dòng $f(x)$. Như vậy ta cần đến gói **Tkz-tab** để làm việc này.

Khai báo gói bằng lệnh sau: `\usepackage{tkz-tab}`

Sau khi khai báo gói rồi, ta bắt đầu vẽ bảng biến thiên nhé:

```
\begin{tikzpicture}
\tkzTabInit[nocadre,lgt=1.5,espcl=2,deltacl=.5]
{$x$/0.7, $f'(x)$/0.7, $f(x)$/2}
{$-\infty$,$-1$,$1$,$2$,$+\infty$}
\end{tikzpicture}
```

x	$-\infty$	-1	1	2	$+\infty$
$f'(x)$					
$f(x)$					

Như vậy ta đã có khung của một bảng biến thiên. Nếu bạn có ý định thay đổi thì cứ thử thay đổi từng thông số một để kiểm tra sự thay đổi của hình vẽ. Như vậy bạn sẽ hiểu thêm được ý nghĩa của các thông số.

\tkzTabInit[tùy chọn]{Đối số dòng}{Đối số cột}

- Tùy chọn **[Tùy chọn]** gồm các thông số đầu vào như sau:

Tùy chọn	Mặc định	Ý nghĩa
espcl	2 cm	Độ rộng cột
lgt	2 cm	Độ rộng cột thứ nhất
deltacl	0.5 cm	Độ rộng cột một và cột cuối cùng có đường kẻ cuối
lw	0.4 pt	Nét kẻ bảng
nocadre	false	Không có đường kẻ quanh ngoài bảng
color	false	Màu bảng có hay không
colorC	white	Màu cột thứ nhất
colorL	white	Màu hàng thứ nhất
colorT	white	Màu bên trong bảng
colorV	white	Màu của các biến trong bảng

- {Đối số dòng}**: Quy định bảng của mình có bao nhiêu dòng. Trong ví dụ trên bạn thấy ta nhập vào là **[\$x\$/0.7, \$f'(x)\$/0.7, \$f(x)\$/2]**. Điều đó có nghĩa bảng có 3 dòng gồm dòng x với độ cao là 0.7, dòng $f'(x)$ với độ cao là 0.7 và dòng $f(x)$ với độ cao là 2. Bạn có thể thay đổi các độ cao và giá trị này để có một bảng như ý.
- {Đối số cột}**: Quy định bảng có bao nhiêu cột. Trong ví dụ trên ta có **[\$-\infty\$,\$-1\$,\$1\$,\$2\$,\$+\infty\$]**. Như vậy bảng của ta gồm các thông số (ở dòng x) lần lượt từ trái qua phải là $-\infty$, -1 , 1 và $+\infty$.

\tkzTabLine{Giá trị dòng}: Để nhập thông số vào các dòng trong bảng biến thiên, ta có thể sử dụng lệnh này, trong đó **Giá trị dòng** gồm các số liệu cần nhập vào. Các số liệu nhập vào ngăn cách nhau bởi dấu **,** như trong ví dụ trên:

\tkzTabLine{,-,z,+d,-,z,+}. Các số liệu này có một số quy tắc sau:

Nhập vào	Hiển thị
z	cột đứng xuyên qua số 0
t	Dường đứng đứt đoạn
d	Kẻ hai đường đứng
h	Tô đường kẻ ô
+	ô mang dấu +
- ô	mang dấu -

Bạn xem bảng dưới đây:

```
\begin{tikzpicture}
\TkzTabInit[nocadre, lgt=1.5, espcl=2, deltacl=.5]
{$x$/0.7, $f'(x)$/0.7, $f(x)$/2}
{$-\infty$, $-1$, $1$, $2$, $+\infty$}
\TkzTabLine{,-,z,+d,-,z,+}
\end{tikzpicture}
```

x	$-\infty$	-1	1	2	$+\infty$
$f'(x)$	-	0	+	-	0
$f(x)$					

\tkzTabVar[Tùy chọn]{Các đối số}: Dòng lệnh vẽ mũi tên thể hiện chiều biến thiên. Mỗi đối số có dạng **s/e** trong đó s quy định chiều mũi tên, e là giá trị số hoặc biểu thức (của hàm số).

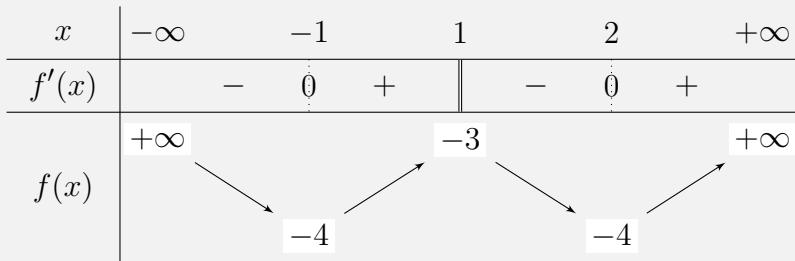
Nhập s	
+/e	Hiển thị giá trị e ở phía trên.
-/e	Hiển thị giá trị e ở phía dưới

Và khi đó, với hai thông số liên tiếp nhau dạng **s/e** ngăn cách nhau bởi dấu **,** thì tkz-tab tự hiểu rằng chiều mũi tên (chiều biến thiên) đi như thế nào. Chẳng hạn như dưới đây:

\tkzTabVar{+/\$+\$\infty\$,-\$/\$-4\$,+\$/\$-3\$, -\$/\$-4\$,+\$/\$+\$\infty\$}

Bạn sẽ có mã hoàn chỉnh của bảng biến thiên như sau:

```
\begin{tikzpicture}
\TkzTabInit[nocadre,lgt=1.5,espcl=2,deltacl=.5]
{$x$/0.7, $f'(x)$/0.7, $f(x)$/2}
{$-\infty$,$-1$,$1$,$2$,$+\infty$}
\TkzTabLine{-,-,z,+,-,z,+,-}
\TkzTabVar{+/$+$+\infty$ ,-$-4$ ,+$-3$ ,-$-4$ ,+$+\infty$}
\end{tikzpicture}
```

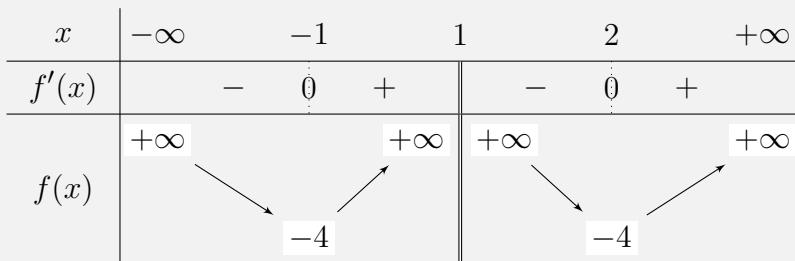


Hoặc nếu dòng $f(x)$, tại $x = 1$ cũng không xác định thì bạn có thể thay bằng dòng lệnh:

```
\TkzTabVar{+/$+$+\infty$ ,-$-4$ ,+D+$+\infty$ , -$-4$ ,+$+\infty$}
```

Bạn sẽ có ngay kết quả:

```
\begin{tikzpicture}
\TkzTabInit[nocadre,lgt=1.5,espcl=2,deltacl=.5]
{$x$/0.7, $f'(x)$/0.7, $f(x)$/2}
{$-\infty$,$-1$,$1$,$2$,$+\infty$}
\TkzTabLine{-,-,z,+,-,z,+,-}
\TkzTabVar{+/$+$+\infty$ ,-$-4$ ,+D+$+\infty$ , -$-4$ ,+$+\infty$}
\end{tikzpicture}
```



Hoặc sử dụng lệnh:

```
\TkzTabVar{+/$+$+\infty$ ,-$-4$ ,+D-$+\infty$/$-\infty$ , +$-4$ ,-$-\infty$}
```

Ta sẽ được:

```
\begin{tikzpicture}
\tkzTabInit[nocadre, lgt=1.5, espcl=2, deltacl=.5]
{$x$/0.7, $f'(x)$/0.7, $f(x)$/2}
{$-\infty, -1, 1, 2, +\infty$}
\tkzTabLine{,-,z,+,-,z,+}
\tkzTabVar{+/$+\infty$ , -/$-4$ , +D/$-\infty$ , +/$-4$ , -/$-\infty$}
\end{tikzpicture}
```

x	$-\infty$	-1	1	2	$+\infty$
$f'(x)$	-	0	+	-	0
$f(x)$	$+\infty$		$+\infty$		-4

Hơn thế nữa, nếu bạn muốn một khoảng nào đó $f'(x)$ hoặc $f(x)$ không xác định bằng cách gạch chéo khoảng thuộc dòng đó, bạn có thể thay $+D$ bằng $+H$ hoặc $-H$,...

```
\begin{tikzpicture}
\tkzTabInit[nocadre, lgt=1.5, espcl=2, deltacl=.5]
{$x$/0.7, $f'(x)$/0.7, $f(x)$/2}
{$-\infty, -1, 1, 2, +\infty$}
\tkzTabLine{,-,z,h,d,-,z,+}
\tkzTabVar{+/$+\infty$ , -H/$-4$ , D/$-\infty$ , +/$-4$ , -/$-\infty$}
\end{tikzpicture}
```

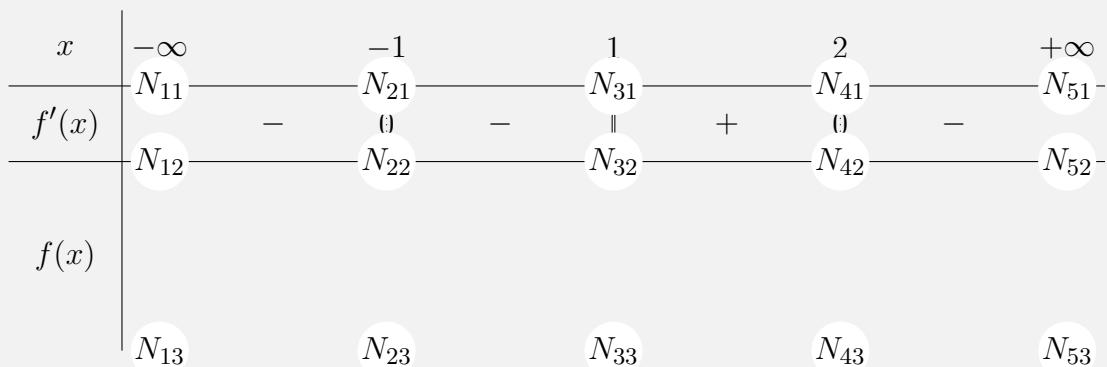
x	$-\infty$	-1	1	2	$+\infty$
$f'(x)$	-	0		-	0
$f(x)$	$+\infty$		-4		$-\infty$

II.3 Kết hợp Tikz và tkz-tab vẽ bảng biến thiên tùy chỉnh

Khi vẽ bảng biến thiên của các hàm số thực tế. Trong nhiều trường hợp quý vị cần tùy chỉnh bảng biến thiên theo ý mình. Tất nhiên trong bảng biến thiên đó thì hai dòng x và $f'(x)$ hầu như không có gì cần tùy chỉnh. Chỉ cần tùy chỉnh dòng $f(x)$ mà thôi.

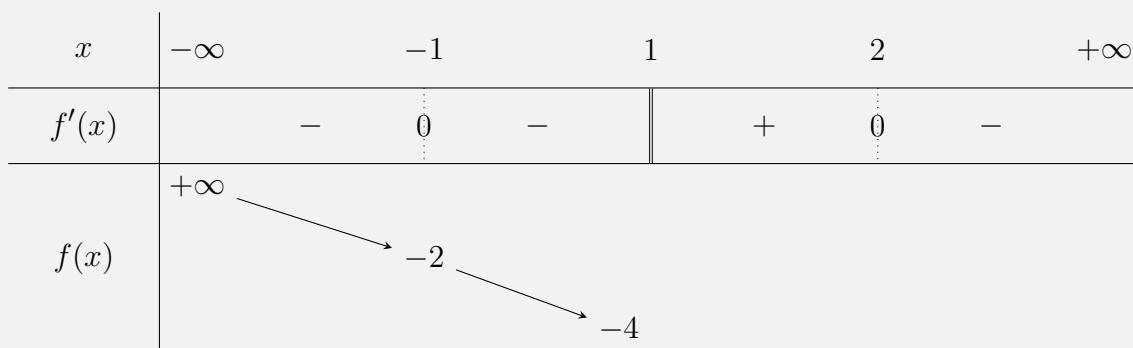
Để tùy chỉnh dòng $f(x)$, trước hết ta tìm hiểu về cách định nghĩa các điểm trong bảng biến thiên. Gói **tkz-tab** đã định nghĩa các điểm cơ bản của bảng theo vị trí như hình sau:

```
\begin{tikzpicture}
\tkzTabInit[nocadre,lgt=1.5,espcl=3,deltacl=0.5]
{$x$/1, $f'(x)$/1, $f(x)$/2.5}
{$-\infty$,$-1$,$1$,$2$,$+\infty$}
\tkzTabLine{,-,z,-,d,+,-,-}
\foreach \i in {1,...,5}{%
\foreach \j in {1,...,3}{%
\draw (N\i\j) node[circle,fill=white,inner sep=0.1]{$N_{\i\j}$};}}
}
\end{tikzpicture}
```



Như vậy Quý vị hoàn toàn có thể vẽ tùy ý dòng $f(x)$ theo ý mình dựa vào các điểm đã được định nghĩa N_{ij} . Số điểm này phụ thuộc bảng của quý vị dài hay ngắn (có nhiều giá trị x hay không). Chẳng hạn như ở bảng trên, khi ta vẽ chiều biến thiên từ $-\infty$ đến 1 là nghịch biến. Nếu bình thường vẽ bằng lệnh `\tkzTabVar` thì sẽ vẽ thẳng một mũi tên từ điểm N_{12} đến điểm N_{33} . Tại hạ lại muốn vẽ kiểu mũi tên từ $+\infty$ đến $f(-1) = -2$ rồi tiếp tục lại vẽ mũi tên thứ hai từ $f(-1)$ đến $f(1) = -4$. Như vậy ta sẽ vẽ bằng tikz thuận tiện dựa vào các điểm có sẵn:

```
\begin{tikzpicture}
\tkzTabInit[nocadre, lgt=2, espcl=3, deltacl=0.5]
{$x$/1, $f'(x)$/1, $f(x)$/2.5}
{$-\infty$, $-1$, $1$, $2$, $+\infty$}
\tkzTabLine{,-,z,-,d,+,z,-,}
\draw (N12) node[below] (A){$+\infty$} ($(N22)!0.5!(N23)$) node (B){$-2$} (N33)
node[above left] (C){$-4$};
\draw[-stealth] (A)--(B);
\draw[-stealth] (B)--(C);
\end{tikzpicture}
```



Tiếp theo, quý vị muốn vẽ đường thẳng đứng biểu diễn hàm số không xác định tại $x = 1$. Ta hãy vẽ đoạn thẳng từ N_{32} đến N_{33} với tham số **double** như sau:

```
\draw [double] ([yshift=-1mm]N32)-(N33);
```

Quý vị nên chú ý **[yshift=-1mm]** ở điểm N_{32} chính là đến đường thẳng đứng cách vạch kẻ ngang 1mm. Nếu quý vị không muốn tách rời ra thì cứ việc bỏ nó đi là xong.

Phía bên phải của bảng biến thiên, Quý vị vẫn vẽ bình thường hai mũi tên biểu diễn đồng biến từ $x = 1$ đến $x = 2$ và nghịch biến từ $x = 2$ đến $+\infty$. Tuy nhiên giá trị $f(2) = -3$ lại nhỏ hơn -2 nên quý vị muốn nó thấp hơn cả $f(-1) = -2$. Vậy thì quý vị hãy xác định điểm đặt $f(2) = -3$ bằng công thức **$$(N42)!0.75!(N43)$$** như sau:

```
\draw (N33) node[above right] (D){$-\infty$} ($(N42)!0.55!(N43)$)
node(E){$-3$} (N53) node[above] (F){$-\infty$};
```

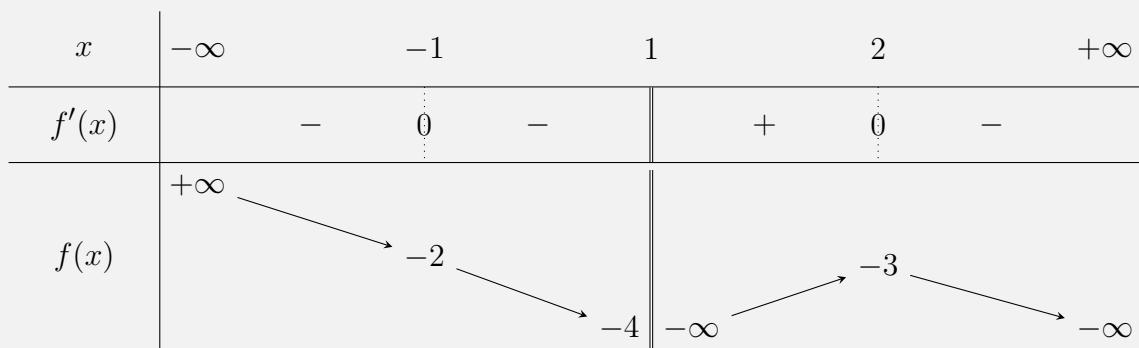
Chú ý rằng con số 0.55 ở trên quý vị có thể tùy ý chọn sao cho phù hợp. Hơn thế nữa là thông số của **$|\f(x)|/2.5$** ở **\tkzTabInit** quý vị cũng nên chọn thay 2.5 bằng các thông số khác để nhìn bảng hài hòa hơn (độ cao của dòng $f(x)$).

Cuối cùng thì quý vị vẽ nốt hai mũi tên còn lại để hoàn thành bảng biến thiên:

```
\draw[-stealth] (D)--(E);
\draw[-stealth] (E)--(F);
```

Kết quả cuối cùng, quý vị hoàn thành bảng biến thiên theo tùy chỉnh của mình:

```
\begin{tikzpicture}
\tkzTabInit[nocadre,lgt=2,espcl=3,deltacl=0.5]
{$x$/1, $f'(x)$/1, $f(x)$/2.5}
{${}-\infty$, ${}-1$, ${}1$, ${}2$, ${}+\infty$}
\tkzTabLine{,-,z,-,d,+,z,-,}
\draw (N12)node[below](A){$+\infty$} ($(N22)!0.5!(N23)$) node (B){$-2$} (N33)
node[above left](C){$-4$};
\draw[-stealth] (A)--(B);
\draw[-stealth] (B)--(C);
\draw[double] ([yshift=-1mm]N32)--(N33);
\draw (N33) node[above right](D){$-\infty$} ($(N42)!0.55!(N43)$)
node(E){$-3$} (N53) node[above](F){$-\infty$};
\draw[-stealth] (D)--(E);
\draw[-stealth] (E)--(F);
\end{tikzpicture}
```



II.4 Vẽ bảng biến thiên bằng tikz thuần

Việc vẽ bảng xét dấu và bảng biến thiên bằng tkz-tab quả thực cũng không quá khó nếu bạn nhớ đầy đủ các lệnh và quy tắc của nó. Tất nhiên nhiều khi bạn không thể nhớ được các lệnh, tùy chọn và quy cách của tkz-tab thì bạn cũng hoàn toàn vẽ được bảng biến thiên bằng tikz thuần một cách đơn giản. Bạn hãy hình dung ra tọa độ của các điểm đặt các thông số trong bảng biến thiên và sử dụng khéo léo **\foreach** là bạn cũng có ngay một bảng biến thiên như ý (đặc biệt với những bảng biến thiên có sự khác biệt thông thường). Chẳng hạn như bảng biến thiên sau:

```
\begin{tikzpicture}
\foreach \x/\texn in {0/x,
  2/-\infty,4/1,6/+infinity} \path
  (\x,3.5)node{$\texn$};
\foreach \x/\texn in
  {0/f'(x),3/-,4/0,5/+} \path
  (\x,2.5)node{$\texn$};
\foreach \x/\y/\texn in {0/1/f(x),
  2/1.5/+infinity,4/.5/-1,6/1.5/+infinity}
  \path (\x,\y) node(\x){$\texn$};
\draw[-stealth] (2)--(4);
\draw[-stealth] (4)--(6);
\draw
  (-.5,3)--(6.5,3) (1,4)--(1,0)
  (-.5,2)--(6.5,2)
  (-.5,4) rectangle (6.5,0)
;
\end{tikzpicture}
```

x	$-\infty$	1	$+\infty$
$f'(x)$	-	0	+
$f(x)$	$+\infty$	-1	$+\infty$

Vấn đề là bạn xác định tọa độ các điểm sao cho khéo léo để bảng đẹp hơn mà thôi.

II.5 Đồ thị hàm số

Để vẽ đồ thị một hàm số trong **Tikz**. Ta đã có lệnh `\draw[...] plot ()`; Cụ thể lệnh như sau:

```
\draw[smooth,samples...] plot[domain=xmin:xmax] (\x ,{f(x)});
```

trong đó **xmin** và **xmax** là các hoành độ xuất phát và kết thúc khi vẽ đồ thị

Ta cần chú ý một số tham số sau trong lệnh vẽ đồ thị:

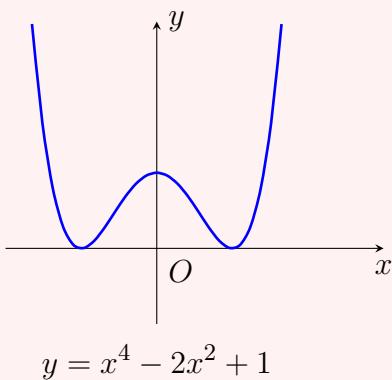
- **domain= a:b** Tham số này quy định cho ta vùng vẽ đồ thị. Với giá trị cụ thể của a và b thì **Tikz** vẽ cho ta phần đồ thị của hàm số với $a \leq x \leq b$. Việc xác định a , b cho phù hợp hình vẽ là điều rất cần thiết để hình vẽ không quá lớn hoặc tràn ra ngoài vùng cần vẽ.
- **smooth** Nếu ta không có tham số này, đồ thị của ta nhìn như các đường gấp khúc nối lại, vẽ rất xấu và nhìn không ra hệ thống cống rãnh gì. Nhất là với các đồ thị là đường cong.
- **variable=\x** hoặc **variable=\y** Định dạng cho ta vẽ đồ thị theo biến x hay biến y . Với các đồ thị thông thường thì ta thường bỏ qua tham số này hoặc sử dụng **variable=\x** là chính.

- Ngoài ra, một số tham số màu (**color**), độ đậm nét của đường (**line width**) hoặc độ hòa trộn (**opacity**) ... cũng có thể tùy biến để được một đồ thị ưng ý.

Hơn nữa, quý vị cũng cần chú ý đến cách viết công thức hàm số để **Tikz** có thể hiểu được. Ví dụ như khi cần vẽ đồ thị hàm số $y = \frac{2}{3}x^3 - 3x + 1$ thì ta không viết như khi viết văn bản là `\frac{2}{3}x^3-3x+1` mà phải viết `(2/3)*(x)^3-3*x+1`. Như vậy các dấu phép tính viết bình thường, nhưng phân số khi tính toán ta phải dùng phép chia (dấu `/`), phép nhân phải dùng dấu nhân (dấu `*`) và phép lũy thừa thì vẫn dùng dấu `^` nhưng chú ý rằng các đối tượng này nằm trong cặp ngoặc đơn () chứ không phải cặp `{ }`.

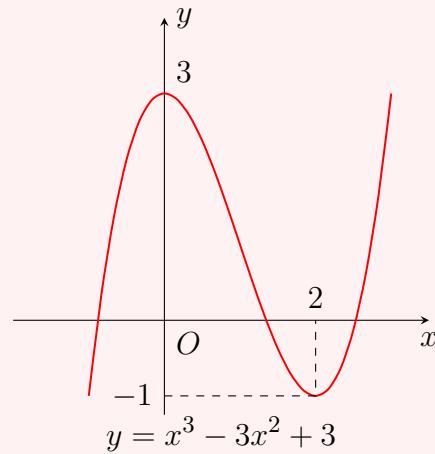
Ta hãy bắt đầu vẽ một vài đồ thị hàm số quen thuộc. Một đoạn lệnh đơn giản như dưới đây đã vẽ được đồ thị hàm số $y = x^4 - 2x^2 + 1$:

```
\begin{tikzpicture}[>=stealth]
\draw[->] (-2,0) --(0,0) node[below right]{$0$} -- (3,0)
node[below]{$x$};
\draw[->](0,-1)--(0,3)
node[right]{$y$};
\draw[smooth,blue,line width=1]
plot [domain=-1.65:1.65]
(\x,{(\x)^4-2*(\x)^2+1});
\draw (0,-1.5) node
{$y=x^4-2x^2+1$};
\end{tikzpicture}
```



Quý vị cũng có thể thử vẽ đồ thị hàm bậc ba $y = x^3 - 3x^2 + 3$ với màu đỏ:

```
\begin{tikzpicture} [->=stealth]
\draw[->] (-2,0) --(0,0) node[below right]{$0$} -- (3.5,0)
node[below]{$x$};
\draw[->] (0,-1.25)--(0,4)
node[right]{$y$};
\draw[smooth,red,line width=0.75]
plot [domain=-1:3]
(\x,{(\x)^3-3*(\x)^2+3});
\draw[dashed] (0,-1)
node[left]{$-1$} -- (2,-1) --
(2,0) node[above]{$2$};
\draw (0,3) node[above right]{$3$};
\draw (0.75,-1.5) node
{$y=x^3-3x^2+3$};
\end{tikzpicture}
```



Tương tự như vậy bạn hoàn toàn có thể vẽ đồ thị hàm số bất kỳ nào.

II.6 Đồ thị hàm số trong hệ tọa độ cực

Cấu trúc lệnh vẽ đồ thị hàm số trong hệ tọa độ cực hoàn toàn đơn giản như sau:

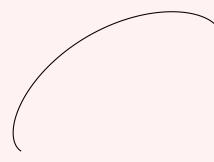
```
\draw[smooth] plot [domain=0:6.3] (canvas polar cs:angle=\x r,radius=
r=f(x));
```

Trong đó ta chú ý đến tham số **domain=0: 6.3** chính là góc ϕ trong hệ tọa độ cực với đơn vị là **rad**. Biến **\x** chính là góc θ và hàm số $r = f(x)$ chính là công thức tính r theo góc θ . Chẳng hạn ta muốn vẽ đồ thị hàm số $r = \cos(3\theta)$ với $0 \leq \theta \leq \pi$ thì ta dùng lệnh:

```
\draw[smooth] plot [domain=0:6.3] (canvas polar cs:angle=\x r,radius=
{\cos(3*\x r)});
```

Khi đó ta sẽ được:

```
\begin{tikzpicture} [->=stealth,scale=5]
\draw[domain=0:pi/3,samples=200,smooth]
plot (canvas polar cs:angle=\x
r,radius= {10*cos(3*\x r)});
\end{tikzpicture}
```



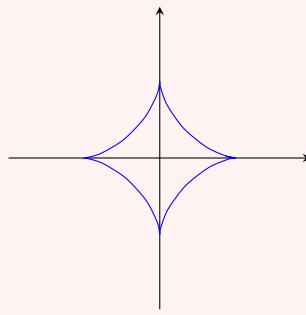
Ngoài ra, ta cũng có thể vẽ đồ thị hàm số với hệ tọa độ để các và phương trình tham số $\begin{cases} x = u(t) \\ y = v(t) \end{cases}$ bởi lệnh vẽ đồ thị như sau:

```
\draw [smooth,variable=\t] plot [domain=mint:maxt] ({u(\t)},{v(\t)});
```

Trong đó **mint** và **maxt** là giá trị nhỏ nhất, giá trị lớn nhất của tham số t cho vùng cần vẽ. Biểu thức $u(t)$ và $v(t)$ chính là hai biểu thức hoành độ và tung độ theo t .

Chẳng hạn ta vẽ đồ thị hàm số $\begin{cases} x = \cos^3 t \\ y = \sin^3 t \end{cases}$ trên vùng $0 \leq t \leq 120\pi$ như sau:

```
\begin{tikzpicture}[>=stealth]
\draw[->] (-2,0)--(2,0);
\draw[->] (0,-2)--(0,2);
\draw [blue,smooth,variable=\t]
plot [domain=0:120*pi ]
({(\cos(\t))^3},{(\sin(\t))^3});
\end{tikzpicture}
```



III Tô miền đồ thị hàm số với Tikz

Qua quá trình trao đổi cùng các bạn vẽ hình trong và ngoài nhóm **TikZ - Asymptote** tôi thấy khá nhiều vướng mắc khi các bạn vẽ phần tô miền của đồ thị hàm số trong các bài toán tính diện tích hình phẳng và thể tích khối tròn xoay. Chính vì vậy tôi viết tạm bài viết ngắn này mong các bạn dễ dàng tháo gỡ khó khăn khi vẽ hình.

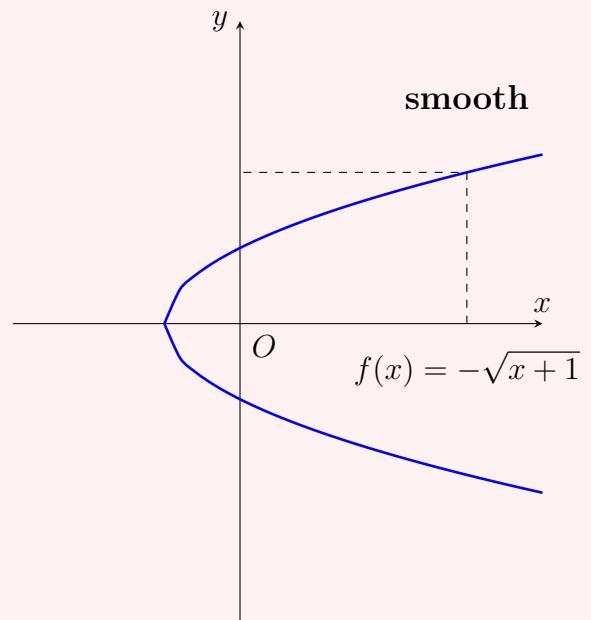
III.1 Vẽ đồ thị hàm số

Thông thường khi vẽ đồ thị hàm số, các bạn vẫn dùng lệnh:

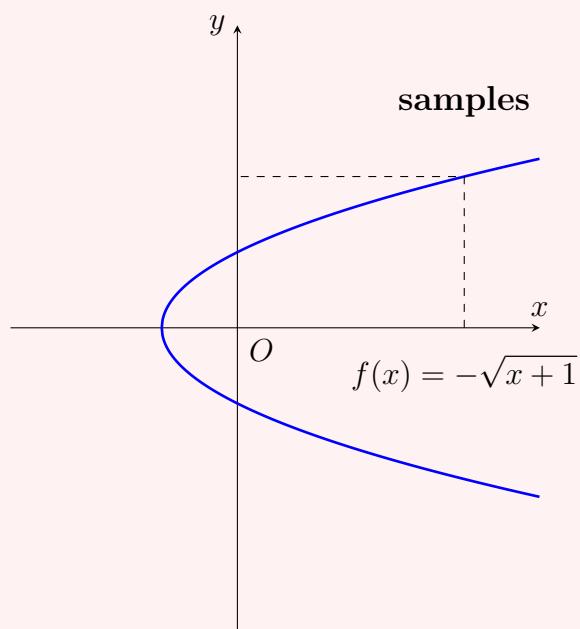
```
\draw [samples=100,smooth] plot [domain=0:120*pi] (\x,{\f(\x)});
```

Trong lệnh trên đây thì **smooth** quyết định độ trơn (nhẵn) của đồ thị hàm số, còn **samples** quyết định số điểm chia để vẽ. Do đó việc sử dụng tham số nào cũng nên lưu ý. Thông thường những hàm đa thức thường gấp ta chỉ cần dùng **smooth** là đủ, một số hàm đặc biệt có độ gấp khúc cao thì ta cần sử dụng **samples** và tăng giá trị này để tạo độ cong hợp lý khi zoom hình. Ví dụ sau đây cho các bạn thấy sự khác biệt ở hai hình để giúp bạn lựa chọn cho phù hợp, bởi giá trị **samples** càng cao thì tốc độ biên dịch sẽ tăng thêm đáng kể:

```
\begin{tikzpicture} [>=stealth]
\def\f(#1){sqrt(#1+1)}
\draw [->] (-3,0)--(0,0)node [below right]{$x$}--(4,0)
node [above]{$x$};
\draw [->]
(0,-4)--(0,4)node [left]{$y$};
\draw [smooth,blue, line width=1]
plot [domain=-1:4] (\x,{\f(\x)})
plot [domain=-1:4] (\x,{-\f(\x)});
\draw (3,-1)
node [above]{$f(x) = -\sqrt{x+1}$};
\draw [dashed] (3,0)--(3,2)--(0,2);
\draw (3,3) node{\bf smooth};
\end{tikzpicture}
```



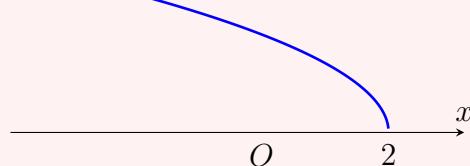
```
\begin{tikzpicture}[>=stealth]
\def\f(#1){sqrt(#1+1)}
\draw[->] (-3,0)--(0,0)node[below right]{$0$}--(4,0)
node[above]{$x$};
\draw[->]
(0,-4)--(0,4)node[left]{$y$};
\draw[samples=500,blue, line width=1]
plot [domain=-1:4] (\x,{\f(\x)})
plot [domain=-1:4] (\x,{-\f(\x)});
\draw (3,-1)
node[above]{$f(x) = -\sqrt{x+1}$};
\draw[dashed] (3,0)--(3,2)--(0,2);
\draw (3,3)node{\bf samples};
\end{tikzpicture}
```



Một điều khác nữa tôi lưu ý các bạn, thông thường các bạn hay để **domain** ở ngay tham số sau lệnh `\draw`. Điều này gây khó khăn khi ta vẽ hai đồ thị liên tiếp để tô miền (mà tôi sẽ nói ở phần sau). Thông thường ta nên để **domain** ở sau `plot [domain=]` thì sẽ chủ động hơn trong việc quyết định khoảng vẽ đồ thị giống như ví dụ trên.

Qua kinh nghiệm vẽ đồ thị hàm số. Việc lựa chọn **domain** cũng cần sự tinh tế nhất định. Chẳng hạn với hàm số $y = \sqrt{2-x}$. Bạn thông thường vẽ đồ thị hàm số với **domain** $=-2:2$ tức là điểm xuất phát là bên trái, điểm kết thúc là bên phải. Nhưng bạn hãy thử xem hình dưới đây để thấy sự lựa chọn **domain** cần sự tinh tế như thế nào:

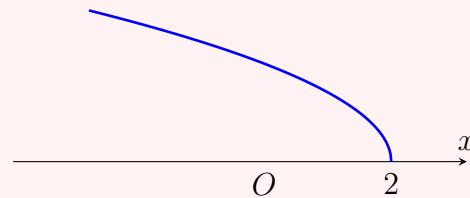
```
\begin{tikzpicture}[>=stealth]
\draw[->] (-3,0)--(0,0)node[below right]{$0$}--(3,0)
node[above]{$x$};
\draw[samples=500,blue, line width=1]
plot [domain=-2:2] (\x,{sqrt(2-\x)});
\draw (2,0)node[below]{$2$};
\end{tikzpicture}
```



Nếu zoom lên, bạn sẽ thấy tại điểm $x = 2$ đồ thị bị cách trực hoành một khoảng nhỏ mặc dù trên phương diện lý thuyết thì hàm số vẫn xác định tại $x = 2$. Lý do chính là cách vẽ của **Tikz** chia thành các khoảng nhỏ để vẽ các đoạn thẳng thông qua tham số **samples** nên ở một lân cận nào đó của điểm $x = 2$ sẽ kết thúc việc vẽ dẫn đến bị đứt đoạn.

Để giải quyết vấn đề này bạn chỉ cần đổi chiều **domain**. Tại sao lại vậy? Vì khi đổi chiều **domain** thì **Tikz** sẽ vẽ từ điểm bắt đầu là $x = 2$ nên đồ thị sẽ liền nét từ đó, phần cách đoạn là phần cuối cùng tại một lân cận của $x = -2$ nên bạn sẽ hầu như thấy không thay đổi tại $x = -2$ (vì không vẽ tiếp đồ thị nữa). Bạn hãy so sánh code dưới đây và code phía trên cùng hai hình vẽ để có kinh nghiệm cho bản thân:

```
\begin{tikzpicture} [->=stealth]
\draw [->] (-3,0)--(0,0)node [below right]{$0$}--(3,0)node [above]{$x$};
\draw [samples=500,blue, line width=1]
plot [domain=2:-2] (\x,{sqrt(2-\x)});
\draw (2,0)node [below]{$2$};
\end{tikzpicture}
```



III.2 Tô miền đồ thị hàm số

Thông thường trong các bài toán, ta có một số dạng tô miền đồ thị như sau:

- Tô miền đồ thị hàm số $y = f(x)$, $x = a$, $x = b$ và trực hoành.
- Tô miền đồ thị hàm số giao bởi hai đồ thị hàm số $y = f(x)$ và $y = g(x)$

Với cả hai dạng trên, ta quy về việc có sẵn các hoành độ để tô miền hay không. Với mục đích code càng ngắn gọn, đơn giản và đặc biệt là độ chính xác càng cao thì càng tốt. Mỗi trường hợp ta có cách giải quyết cụ thể khác nhau.

III.2.1 Khi đã biết hoành độ các điểm cận của miền cần tô

Khi đã biết hoành độ các điểm cận của miền cần tô, ta dùng lệnh ngắn gọn như sau:

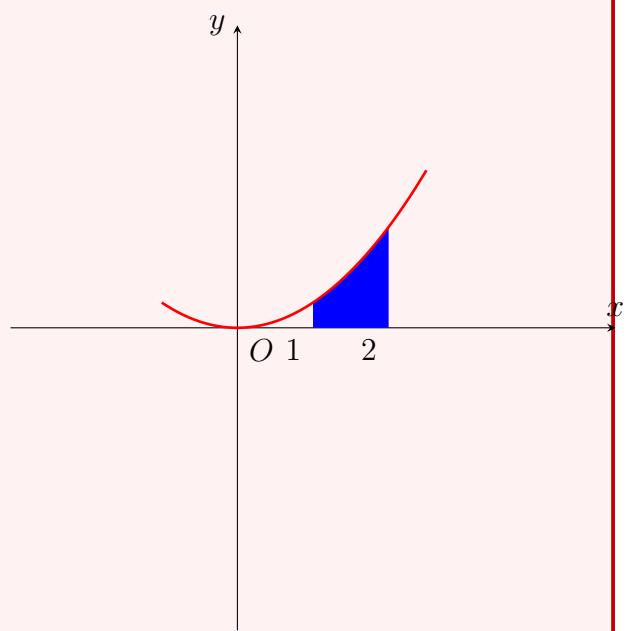
Chẳng hạn cần tô miền giới hạn bởi đồ thị hàm số $y = f(x)$, $y = g(x)$, $x = a$ và $x = b$, khi đó ta sử dụng lệnh:

```
\draw [smooth] plot [domain=a:b] (\x,{f(\x)})--plot [domain=b:a] (\x,{g(\x)})--cycle;
```

Ở đây ta để ý đầu bút vẽ. Hiểu đơn giản sẽ là bút vẽ đồ thị hàm số $f(x)$ từ $x = a$ đến điểm $x = b$ và sau đó vẽ tiếp đồ thị hàm số $g(x)$ từ điểm $x = b$ đến $x = a$. Và như vậy bút vẽ sẽ đi một miền kín để tô màu ưng ý.

Ví dụ cụ thể, chẳng hạn tô miền đồ thị hàm số $y = \frac{x^3}{3}$, $x = 1$, $x = 2$ và trực hoành:

```
\begin{tikzpicture}[>=stealth]
\def\f(#1){((#1)^2)/3}
\draw[->] (-3,0)--(0,0) node[below right]{$0$}--(5,0)
            node[above]{$x$};
\draw[->] (0,-4)--(0,4)
            node[left]{$y$};
\fill[blue,smooth] (1,0) --
    plot[domain=1:2] (\x,{\f(\x)})
    -- (2,0) -- cycle;
\draw[smooth,red, line width=1]
    plot[domain=-1:2.5]
    (\x,{\f(\x)});
\draw (1,0) node[below left]{$1$}
      (2,0) node[below left]{$2$};
\end{tikzpicture}
```

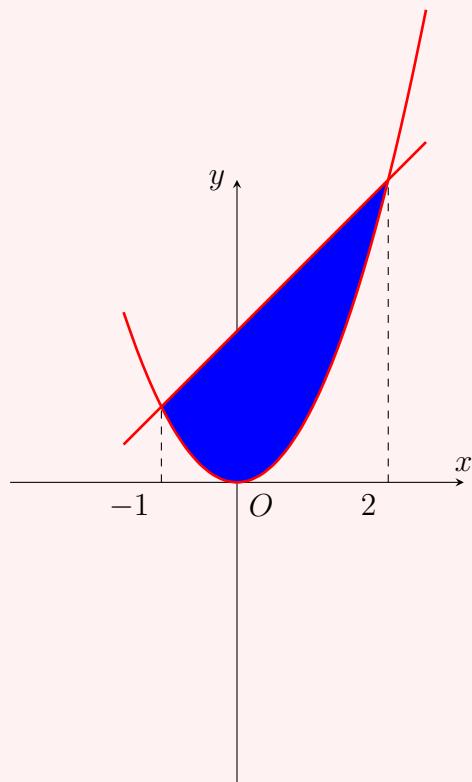


Như vậy việc còn lại bạn chỉ cần vẽ hai đường thẳng $x = 1$ và $x = 2$ nữa là việc tô miền đã hoàn thành.

Một ví dụ khác là tô miền phần giao hai đồ thị mà ta có thể dễ dàng tìm thấy các hoành độ giao điểm bằng cách giải phương trình $f(x) = g(x)$. Chẳng hạn tô miền phần giới hạn bởi hai đồ thị hàm số $y = x + 2$ và $y = x^2$

Ta dễ dàng tìm được hoành độ giao điểm của hai đồ thị này là nghiệm phương trình $x^2 = x + 2 \Rightarrow x = -1$ hoặc $x = 2$. Khi đó ta thực hiện vẽ đồ thị $y = x^2$ từ $x = -1$ đến $x = 2$ rồi tiếp tục vẽ $y = x + 2$ từ $x = 2$ đến $x = -1$:

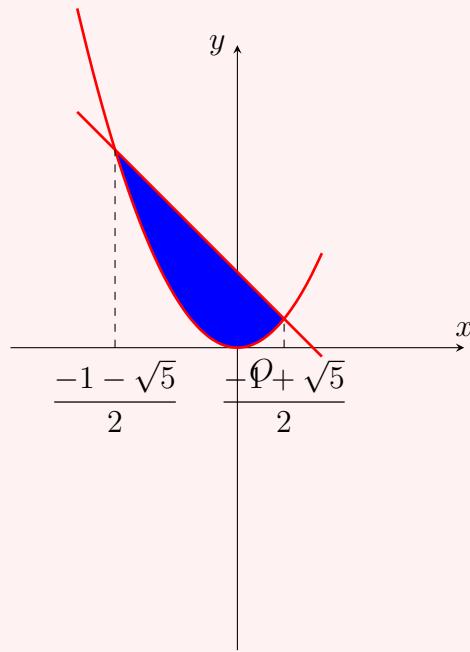
```
\begin{tikzpicture} [->=stealth]
\draw [->] (-3,0)--(0,0) node[below right]{$0$} -- (3,0)
node[above]{$x$};
\draw [->] (0,-4)--(0,4)
node[left]{$y$};
\fill[blue,smooth]
plot [domain=-1:2] (\x,{(\x)^2})
-- plot [domain=2:-1] (\x,{\x+2})
-- cycle;
\draw [smooth,red, line width=1]
plot [domain=-1.5:2.5]
(\x,{(\x)^2});
\draw [smooth,red, line width=1]
plot [domain=-1.5:2.5] (\x,{\x+2});
\draw[dashed] (-1,0)--(-1,1)
(2,0)--(2,4);
\draw (-1,0) node[below left]{$-1$}
(2,0) node[below left]{$2$};
\end{tikzpicture}
```



Chú ý rằng các bài toán có thể tìm hoành độ giao điểm nhưng hoành độ đó là số vô tỉ hoặc các số thập phân vô hạn. Ta không nên lấy các giá trị xấp xỉ của nó mà hãy gán cho giá trị đó vào một biến. Khi đó ta hoàn toàn có thể dùng biến đó đưa vào **domain** để chính xác đồ thị.

Chẳng hạn như hoành độ giao điểm của hai đồ thị $f(x) = x^2$ và $g(x) = -x + 1$. Khi giải phương trình $f(x) = g(x)$ ta có hai nghiệm lần lượt là $x_1 = \frac{-1 - \sqrt{5}}{2}$ và $x_2 = \frac{-1 + \sqrt{5}}{2}$. Nếu bạn bấm máy tính để lấy xấp xỉ các con số để đưa vào **domain** dẫn đến đồ thị không hoàn toàn chính xác và đồng thời việc nhớ các con số dài ngoằng để nhập vào cũng làm code của bạn bớt đi tính thẩm mĩ. Giải pháp là bạn sẽ gán hai biến $a = x_1$ và $b = x_2$ sau đó đưa a, b vào **domain**

```
\begin{tikzpicture}[>=stealth]
\pgfmathsetmacro{\a}{(-1-sqrt(5))/2}
\pgfmathsetmacro{\b}{(-1+sqrt(5))/2}
\draw[->] (-3,0)--(3,0) node[below right]{$0$} -- (3,0)
node[above]{$x$};
\draw[->] (0,-4)--(0,4)
node[left]{$y$};
\fill[blue,smooth]
plot [domain=\a:\b](\x,{(\x)^2})
-- plot [domain=\b:\a]
(\x,{-\x+1}) -- cycle;
\draw[smooth,red, line width=1]
plot [domain=\a-0.5:\b+0.5]
(\x,{(\x)^2});
\draw[smooth,red, line width=1]
plot [domain=\a-0.5:\b+0.5]
(\x,{-\x +1});
\draw[dashed] (\a,0) -- (\a,-\a+1)
(\b,0) -- (\b,-\b+1);
\draw (\a,0) node[below]{$\frac{-1-\sqrt{5}}{2}$}
(\b,0) node[below]{$\frac{-1+\sqrt{5}}{2}$};
\end{tikzpicture}
```



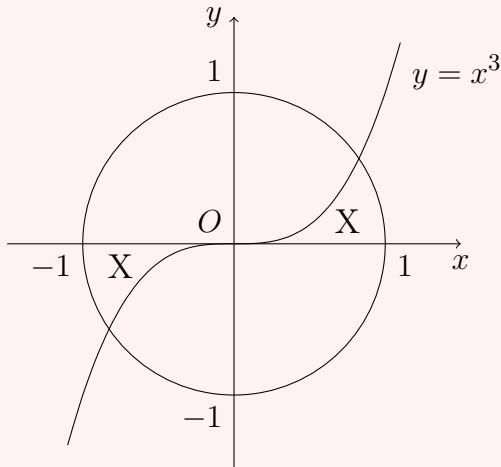
Nếu cả hai hàm số đều có công thức phức tạp mà không phải là bậc nhất như trên, bạn hãy gán thêm hai biến fa và fb là các tung độ giao điểm tương ứng với lệnh **pgfmathsetmacro** như khi gán biến a và b rồi đưa các tung độ tương ứng vào điểm cần vẽ.

III.2.2 Tô miền đồ thị không biết hoành độ giao điểm

Một vấn đề đặt ra là nhiều khi ta có hàm số $f(x)$, hàm số $g(x)$ nhưng việc giải phương trình $f(x) = g(x)$ gặp khó khăn. Khi đó đến cả công thức nghiệm ta cũng chẳng có thì làm thế nào? Bạn hãy dùng môi trường **scope** với lệnh **clip**

Một ví dụ đơn giản là tô miền phần giới hạn của đồ thị hàm số $f(x) = x^3$ và đường tròn có tâm là gốc tọa độ, bán kính bằng 1 được đánh dấu X như hình vẽ sau:

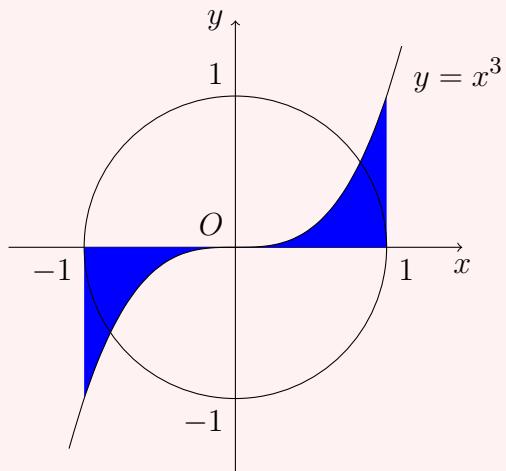
```
\begin{tikzpicture}[x=2cm,y=2cm]
\draw[->] (-1.5,0)--(1.5,0)
  node[below]{$x$};
\draw[->] (0,-1.5) --(0,1.5)
  node[left]{$y$};
\draw plot [domain=-1.1:1.1,smooth]
  (\x,{(\x)^3})
node[below right]{$y=x^3$};
\draw (0,0) circle(1);
\path
(0,0) node[above left]{$O$}
(1,0) node[below right]{$1$}
(-1,0) node[below left]{$-1$}
(0,1) node[above left]{$1$}
(0,-1) node[below left]{$-1$};
\draw (0.75,0.15) node{X}
(-0.75,-0.15)node{X};
\end{tikzpicture}
```



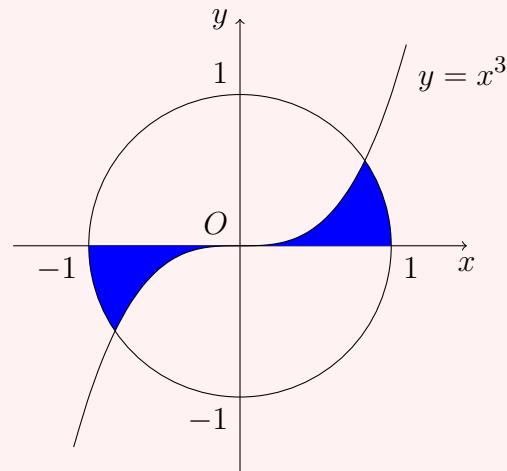
Bình thường bạn sẽ nghĩ đến việc dùng phương trình đường tròn $x^2 + y^2 = 1$ để suy ra hàm số $y = \sqrt{1 - x^2}$. Tuy nhiên việc giải phương trình $x^3 = \sqrt{1 - x^2}$ đã quá sức bạn. Hoặc có thể bạn giải được nhưng mất rất nhiều thời gian.

Bạn hãy phân tích phần được tô miền sẽ nằm trong đường tròn và nằm ở phần giới hạn bởi đồ thị $f(x) = x^3$ với trực hoành và đường thẳng $x = 1$, $x = -1$. Khi đó bạn hãy tô toàn bộ phần giới hạn bởi đồ thị hàm số $f(x) = x^3$ với các đường $x = 1$, $x = -1$ và trực hoành. Sau đó cắt đi chỉ lấy phần nằm bên trong đường tròn:

```
\begin{tikzpicture}[x=2cm,y=2cm]
\draw[->] (-1.5,0)--(1.5,0)
  node[below]{$x$};
\draw[->] (0,-1.5) --(0,1.5)
  node[left]{$y$};
\fill[blue,smooth] (-1,0) --
  plot[domain=-1:1] (\x,{(\x)^3})
  -- (1,0) -- cycle;
\draw plot[domain=-1.1:1.1,smooth]
  (\x,{(\x)^3}) node[below right]
  {$y=x^3$};
\draw (0,0) circle(1);
\path
(0,0) node[above left]{$0$}
(1,0) node[below right]{$1$}
(-1,0) node[below left]{$-1$}
(0,1) node[above left]{$1$}
(0,-1) node[below left]{$-1$};
\end{tikzpicture}
```



```
\begin{tikzpicture}[x=2cm,y=2cm]
\draw[->] (-1.5,0)--(1.5,0)
  node[below]{$x$};
\draw[->] (0,-1.5) --(0,1.5)
  node[left]{$y$};
\begin{scope}
\clip (0,0) circle (1);
\fill[blue,smooth] (-1,0) --
  plot [domain=-1:1] (\x,{(\x)^3})
  -- (1,0) -- cycle;
\end{scope}
\draw plot [domain=-1.1:1.1,smooth]
  (\x,{(\x)^3}) node[below right]
  {$y=x^3$};
\draw (0,0) circle(1);
\path
(0,0) node[above left]{$O$}
(1,0) node[below right]{$1$}
(-1,0) node[below left]{$-1$}
(0,1) node[above left]{$1$}
(0,-1) node[below left]{$-1$};
\end{tikzpicture}
```



Và bây giờ bạn đã thấy thành quả. Bạn chỉ việc xem hai code trên bạn đã hiểu việc dùng **scope** và **clip** hiệu quả như thế nào.

Bạn hãy thử thực hành với các hình vẽ khác xem sao.

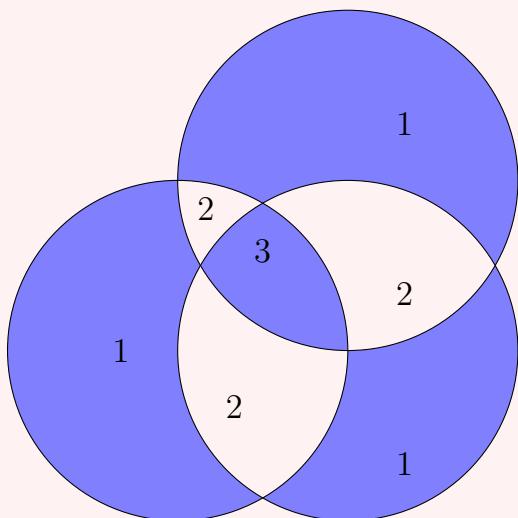
III.2.3 Sử dụng even odd rule

Một cách khác để tô một miền kín là sử dụng tham số **even odd rule**. Hiểu đơn giản **even** là chẵn, **odd** là lẻ. Và tham số **even odd rule** chính là lựa chọn miền tô không giao nhau của các miền kín.

Ví dụ ta có ba miền kín coi như ba tập hợp A , B và C . Khi biểu diễn ba tập hợp này sẽ có các miền giao nhau và không giao nhau. Miền nào thuộc số chẵn trong số ba tập hợp A, B, C thì sẽ không tô và chỉ tô miền lẻ.

Các bạn hãy xem hình dưới đây để hiểu thế nào là miền **chẵn** (even) và **lẻ** (odd):

```
\begin{tikzpicture}[scale=0.75]
\fill[blue!50, even odd rule] (0,0)
    circle (3) (3,0) circle (3)
    (3,3) circle (3);
\draw (0,0) circle (3);
\draw (3,0) circle (3);
\draw (3,3) circle (3);
\draw (1,-1)node{2} (4,-2)node{1}
(4,1)node{2} (4,4)node{1}
(1.5,1.75)node{3}
(0.5,2.5)node{2} (-1,0)node{1};
\end{tikzpicture}
```



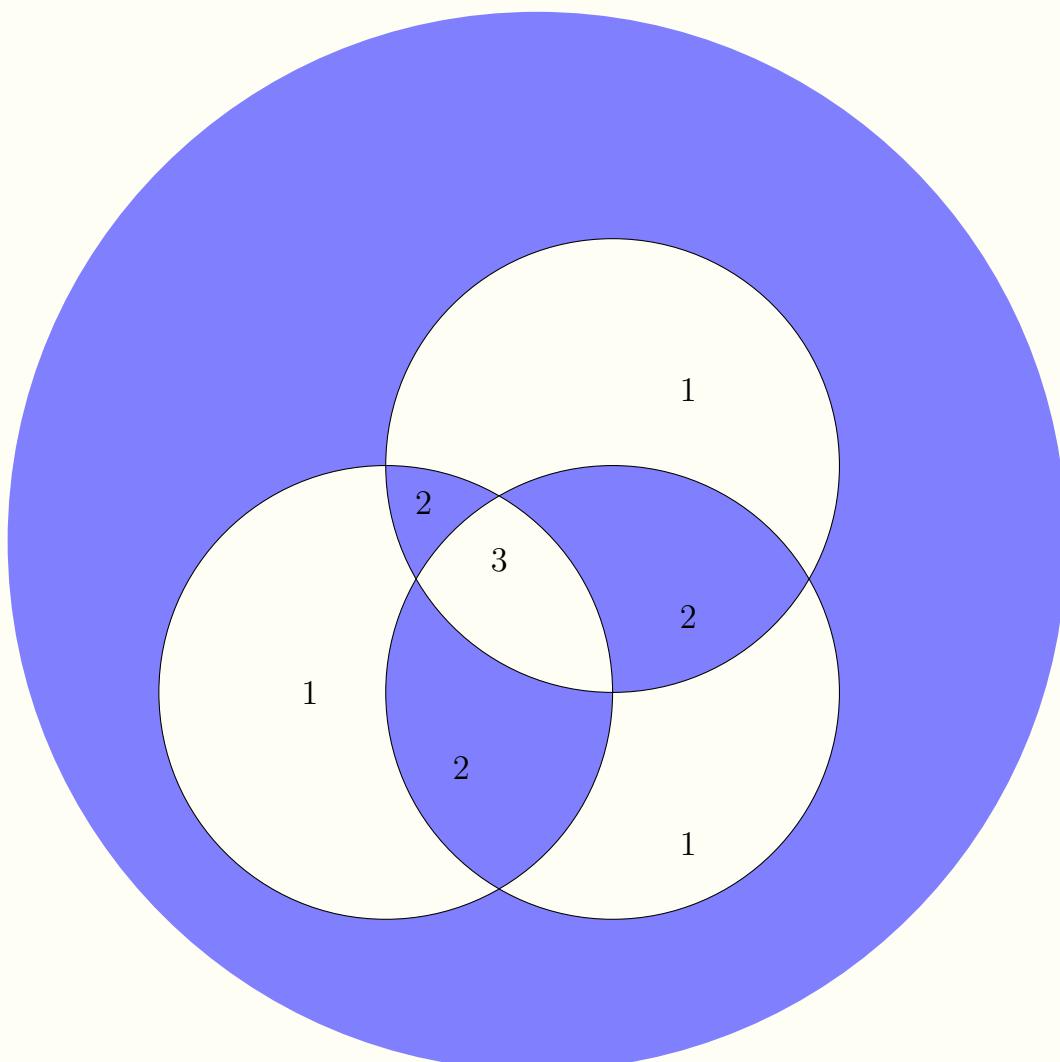
Nhìn hình vẽ trên, các miền là giao của số lẻ miền sẽ được tô, các miền là giao của số chẵn miền không được tô. Tác động này là do sử dụng lệnh:

```
\fill[blue!50,even odd rule](0,0) circle (3)
(3,0) circle (3) (3,3) circle (3);
```

Mặc định của tikz (khi không sử dụng even odd rule) là tô kín mọi miền kể cả số chẵn và số lẻ. Khi đó ta sử dụng khéo léo **even odd rule** sẽ tô được miền ta ưng ý. Như vậy, khi tô các miền lẻ thì ta sử dụng **even odd rule**. Vậy muốn tô miền chẵn thì làm thế nào? Ta hãy đánh lừa **Tikz** bằng cách thêm một miền bao bên ngoài tất cả các miền trên, khi đó các miền đang lẻ sẽ thành chẵn và ngược lại. Hoặc là bạn có thể sử dụng một cách khác, tức là bạn tô sẵn nền, sau đó to to các miền lẻ bằng **white**. Vậy là các miền chẵn sẽ được tô. Chú ý rằng muốn dùng thủ thuật đánh tráo này, bạn sẽ phải clip hình bởi nếu không sẽ có những miền lẻ mới và miền chẵn mới như hình dưới đây tôi đã thêm một miền mới bằng lệnh:

```
\fill[blue!50, even odd rule] (0,0) circle (3) (3,0) circle (3)
(3,3) circle (3) (2,2) circle (7);
```

```
\begin{tikzpicture}
\fill[blue!50, even odd rule] (0,0) circle (3) (3,0) circle (3) (3,3) circle
(3) (2,2) circle (7);
\draw (0,0) circle (3);
\draw (3,0) circle (3);
\draw (3,3) circle (3);
\draw (1,-1)node{2} (4,-2)node{1} (4,1)node{2} (4,4)node{1} (1.5,1.75)node{3}
(0.5,2.5)node{2} (-1,0)node{1};
\end{tikzpicture}
```



III.3 Các kiểu tô miền

Bạn thích tô màu các miền thì thực hiện các lệnh **fill** như trên đây tôi trình bày, chỉ việc đổi tên màu là bạn đã đạt được ý nguyện. Nếu bạn không muốn tô miền bằng màu sắc bạn cũng có các kiểu tô miền khác mà **Tikz** đã thiết kế sẵn gọi là **pattern**.

Để sử dụng **pattern**, trong file biên dịch bạn cần gọi thư viện này ra bằng

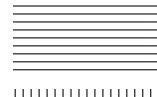
`\usetikzlibrary{patterns}` trước `\begin{document}`.

Tại lệnh **fill** bạn sử dụng như sau:

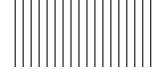
\fill [pattern=name pattern].

Một số lựa chọn kiểu tô:

- horizontal lines



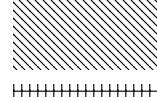
- vertical lines



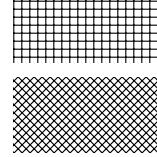
- north east lines



- north west lines



- grid



- crosshatch



- dots



- crosshatch dots



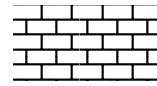
- fivepointed stars



- sixpointed stars



- bricks



- checkerboard



- checkerboard light gray



- horizontal lines light gray



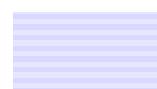
- horizontal lines gray



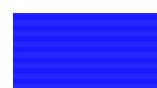
- horizontal lines dark gray



- horizontal lines light blue



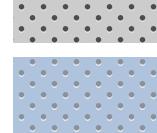
- horizontal lines dark blue



- crosshatch dots gray

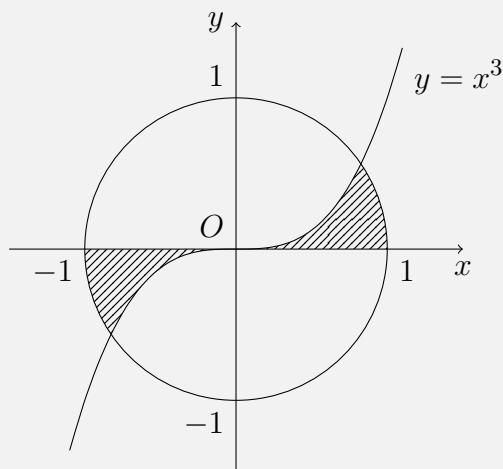


- crosshatch dots light steel blue



Chẳng hạn, ở ví dụ trên tôi sẽ dùng kiểu gạch sọc:

```
\begin{tikzpicture}[x=2cm,y=2cm]
\draw[->] (-1.5,0) -- (1.5,0) node[below]{$x$};
\draw[->] (0,-1.5) -- (0,1.5) node[left]{$y$};
\begin{scope}
\clip (0,0) circle (1);
\fill[pattern=north east lines,smooth] (-1,0) -- plot[domain=-1:1]
(\x,{(\x)^3}) -- (1,0) -- cycle;
\end{scope}
\draw plot[domain=-1.1:1.1,smooth] (\x,{(\x)^3}) node[below right]{$y=x^3$};
\draw (0,0) circle(1);
\path
(0,0) node[above left]{$0$}
(1,0) node[below right]{$1$}
(-1,0) node[below left]{$-1$}
(0,1) node[above left]{$1$}
(0,-1) node[below left]{$-1$};
\end{tikzpicture}
```



Vậy là bạn đã hoàn thành được việc vẽ tô miền cho các hàm số trong các bài tích phân rồi đó. Chúc bạn có thể đơn giản hơn một phần việc không hề dễ dàng này trong tương lai!

IV Tư duy hình họa để vẽ hình

Để có một hình vẽ khoa học đảm bảo độ chính xác, hơn nữa lại đẹp và code vẽ tối ưu. Khuyến cáo bạn nên tìm hiểu về phương pháp tư duy và trình tự phân tích, tổng hợp các yếu tố cơ bản của một hình vẽ. Từ đó bạn sẽ có thể vẽ bất cứ thứ gì bạn muốn.

IV.1 Trình tự cơ bản

Bạn hãy tham khảo sơ đồ sau:



Các bước khuyên cáo trên đây mặc dù nhìn có vẻ rât rối và nhiều thao tác. Tuy nhiên, khi bạn rèn luyện thường xuyên, quá trình phân tích hình vẽ sẽ rât nhanh, có thể chỉ xảy ra trong một khoảng thời gian rât ngắn khi nó đã trở thành thói quen của bạn. Việc phân tích hình vẽ là điều tối quan trọng, bởi từ đó bạn sẽ có thể tìm ra phương pháp vẽ tối ưu nhất cho hình vẽ của bạn.

IV.2 Lựa chọn hệ tọa độ hợp lý

Việc vẽ hình bằng **Tikz** là làm việc với các tọa độ (điểm đặt các đối tượng hình ảnh) và các câu lệnh (lệnh vẽ hình). Như vậy việc lựa chọn hợp lý khi vẽ hình là rất cần thiết.

Với mỗi hình vẽ, bạn cần hình dung vị trí đặt của hình. Vì vậy bạn lựa chọn gốc tọa độ là điểm nào để thuận tiện điều khiển các tọa độ tiếp theo cũng nên cân nhắc. Mặt khác, bạn sẽ sử dụng tọa độ **Đè-các** hay tọa độ cực cũng nên quan tâm. Chẳng hạn bạn muốn vẽ một tam giác với đường tròn nội (ngoại tiếp) bạn sẽ vẽ như thế nào? Vẽ đường tròn trước hay tam giác trước? gốc tọa độ nên đặt ở đâu? ... Với tôi thì tôi sẽ vẽ đường tròn trước, và tâm đường tròn nằm đúng gốc tọa độ. Ngay lập tức dùng hệ tọa độ cực thì có ngay ba điểm nằm trên đường tròn là ba đỉnh tam giác nội tiếp đường tròn. Nếu bạn vẽ đường tròn trước, việc bạn phải đi tìm tâm, bán kính đường tròn ngoại tiếp tam giác sẽ vất vả hơn. Tất nhiên trong một bài toán tổng hợp nhiều hơn nữa, có thể bạn vẫn phải lựa chọn vẽ tam giác trước, bởi ba đỉnh tam giác được sinh ra từ việc vẽ các đường trước đó. Khi ấy bạn sẽ cần đến việc tạo và sử dụng macro.

IV.3 Sử dụng Macro

Trong nhiều hình vẽ, việc sử dụng các thư viện của **Tikz** có thể đảm bảo vẽ cho bạn hầu hết các hình. Nhưng vấn đề lại là những thư viện đó bạn ít khi dùng đến nên việc nhớ các câu lệnh, các tùy chọn của thư viện không hề đơn giản. Chính vì vậy việc sử dụng macro hợp lý mặc dù có vẻ rườm rà nhưng lại tiết kiệm nơ ron thằn kinh của bạn một cách đáng kể.

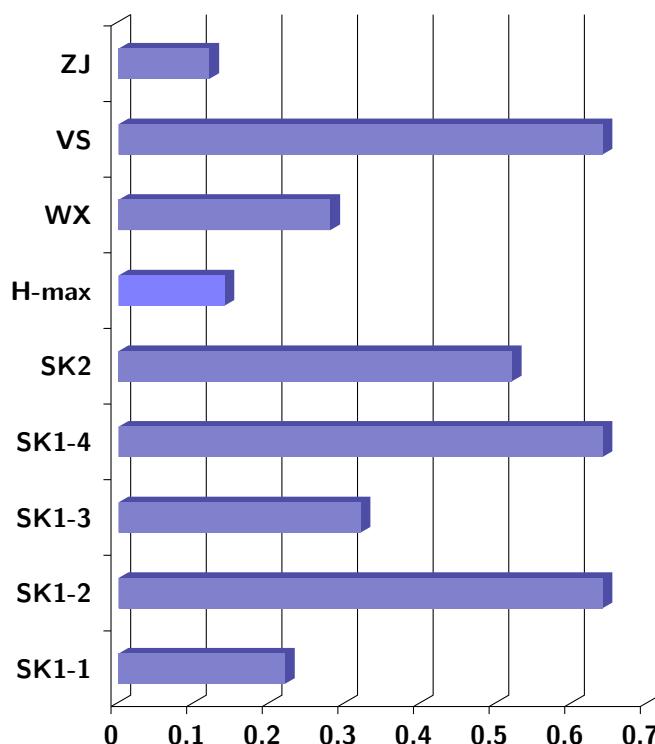
Hoặc hơn nữa, nhiều khi hình vẽ có những đối tượng tương tự nhau, chỉ sai khác một vài thông số cơ bản. Vậy bạn hãy tạo macro để tùy biến vừa đơn giản lại nhanh gọn.

Chẳng hạn như trong sơ đồ ở phần trên. Khi vẽ tôi đã hình dung có nhiều khung chữ cùng loại, điểm khác biệt là màu nền, nội dung chữ, độ lớn khung và điểm đặt nó. Mặc dù **Tikz** cũng có thư viện tùy chọn node nhưng không phong phú bằng. Vậy tôi sẽ nghĩ đến việc tạo macro cho việc này, thông số đầu vào sẽ là tọa độ hiển thị, độ lớn khung, màu nền và nội dung chữ trong khung Và tôi đã làm như thế.

IV.4 Sử dụng vòng lặp **foreach**

Nếu trong hình vẽ của bạn có nhiều tình huống lặp đi lặp lại, bạn cũng có thể sử dụng vòng lặp **foreach** để **Tikz** thực hiện cho bạn. Sử dụng hợp lý sẽ tiết kiệm thời gian soạn code vẽ và các chi tiết đảm bảo thống nhất trong một hình vẽ.

Chẳng hạn như bạn muốn vẽ một biểu đồ hình cột, rõ ràng **Tikz** có tùy chọn **ybar** để vẽ biểu đồ cột nhưng không thể linh hoạt như bạn tự vẽ:



Với biểu đồ trên, bạn chỉ cần tư duy một chút là bạn hoàn toàn có thể vẽ hàng loạt biểu đồ tương tự mà chỉ cần thay đổi một vài thông số.

IV.5 Kỹ năng tìm kiếm

Không phải lúc nào bạn cũng nhớ được tất cả, có những vấn đề bạn nhớ mang máng, có những vấn đề bạn chưa từng được nghe đến. Kỹ năng tìm kiếm sẽ giúp bạn rút ngắn thời gian vẽ hình. Những địa chỉ tìm kiếm tốt nhất khi vẽ hình cho bạn là cùn **pgfmanul.pdf** và trang web <https://tex.stackexchange.com>.

Bạn cũng nên chú ý cách tìm kiếm, cần đưa ra những từ khóa mẫu chốt cho quá trình tìm kiếm thu gọn hơn để tiết kiệm thời gian.

V Export hình ảnh

Khi bạn vẽ hình để soạn tài liệu, tài liệu của bạn có nhiều hình vẽ code Tikz sẽ làm chậm quá trình biên dịch. Vì vậy bạn có thể biên dịch mỗi hình thành một file pdf riêng để `\includegraphics` hình ảnh vào nhằm cải thiện thời gian biên dịch.

Mặt khác, khi bạn cần hình ảnh để upload lên web hoặc insert vào các trình soạn thảo tài liệu khác (chẳng hạn như MS Word) thì bạn làm như thế nào?

V.1 Cài đặt ImageMagick và Ghostscript

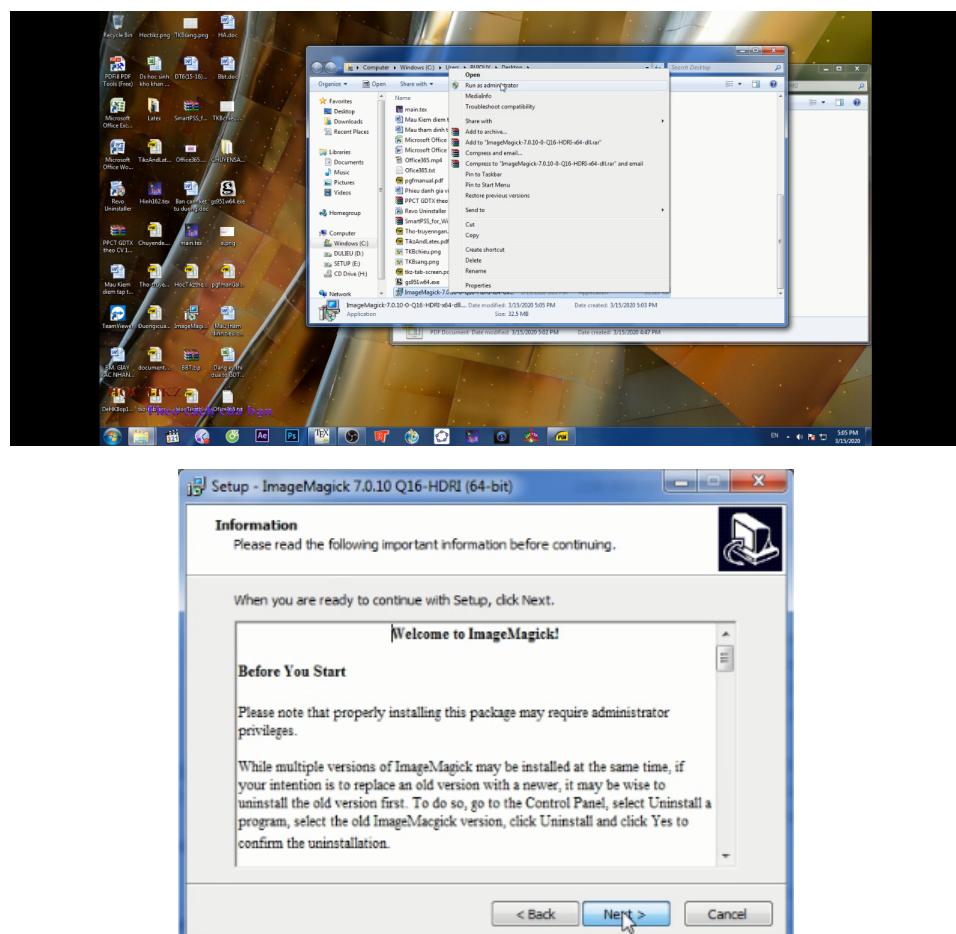
Bạn cần download ImageMagick và Ghostscript từ Internet về máy tính và cài đặt một cách bình thường. Bạn có thể vào link sau:

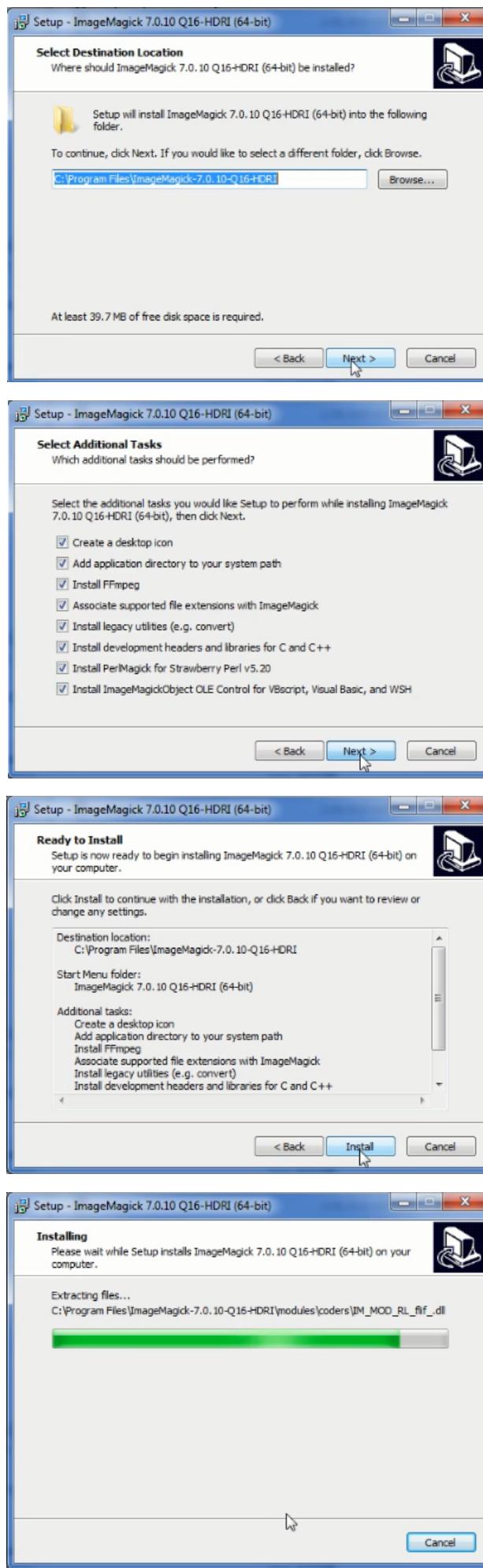
ImageMagick: <https://imagemagick.org>

Ghostscript <https://www.ghostscript.com>

Tất nhiên bạn cũng có thể vào Google và seach với các từ khóa **ImageMagick** và **Ghostscript** và bạn sẽ tìm thấy ngay những trang này. Vào đây bạn hãy chọn phiên bản cho phù hợp với hệ điều hành máy của bạn (32 bit hoặc 64 bit) và cài đặt theo thứ tự như sau:

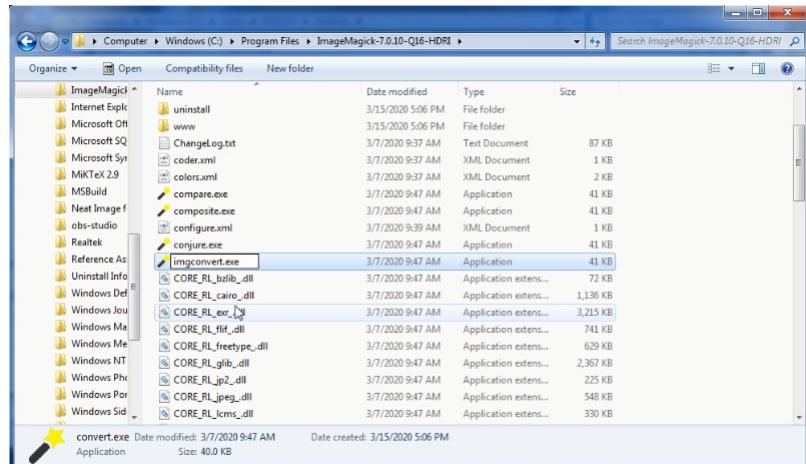
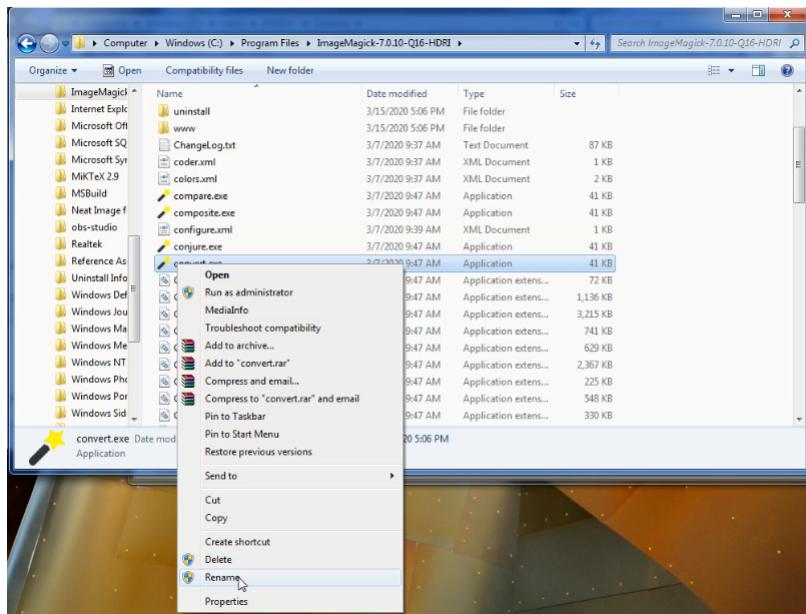
- **Bước 1:** Cài đặt ImageMagick (lưu ý khi cài đặt chọn đầy đủ các tính năng).



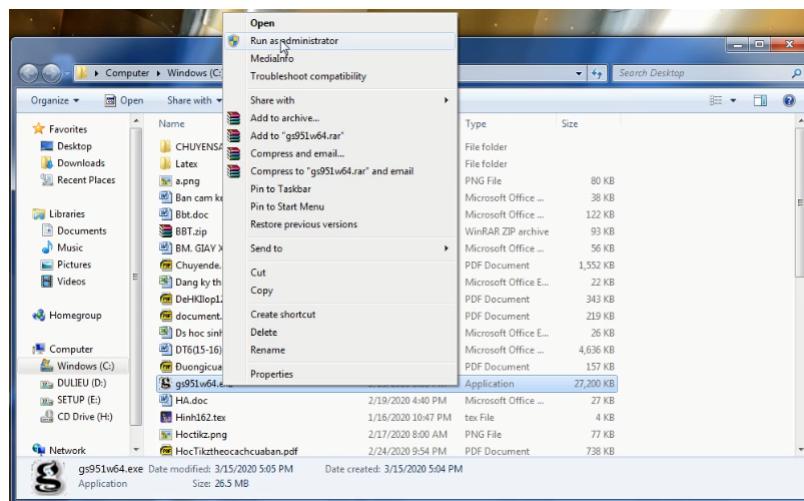




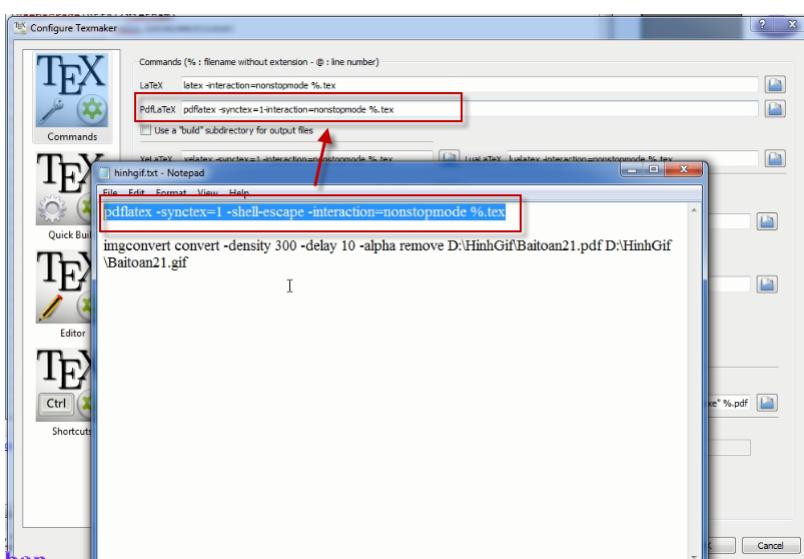
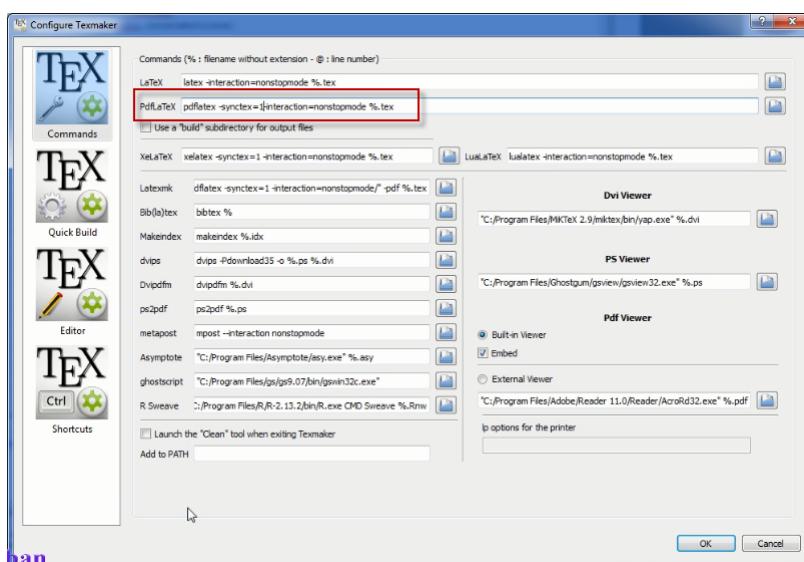
- **Bước 2:** Khi cài xong ImageMagick, bạn vào vào thư mục cài đặt ImageMagick và tìm đến file convert.exe và đổi tên file này thành imgconvert.exe

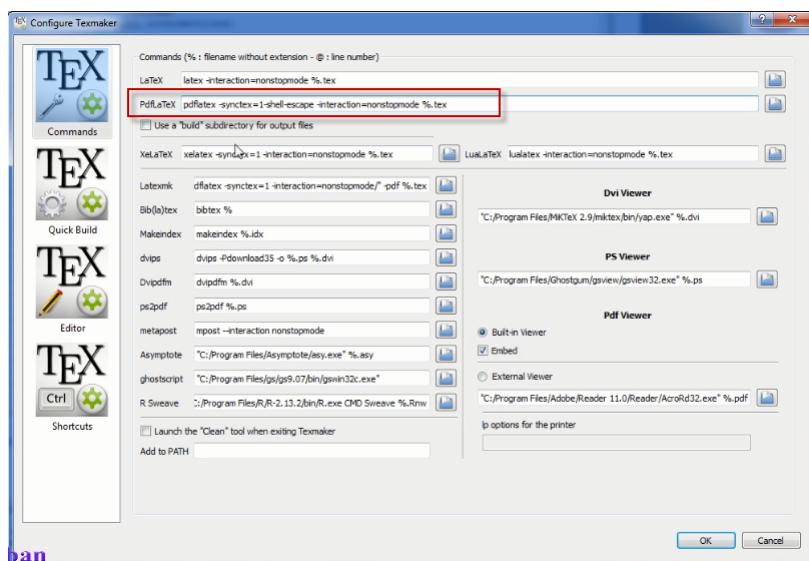


- **Bước 3:** Cài đặt Ghostscript.



- **Bước 4:** Cấu hình Configuration. Bạn thay thế dòng pdflatex trong Configuration từ pdflatex -synctex=1 -interaction=nonstopmode %.tex thành: pdflatex -synctex=1 -shell-escape -interaction=nonstopmode %.tex





- **Bước 5:** Thêm tùy chọn vào khai báo lớp văn bản:

```
\documentclass[tikz,border=1mm,convert={outfile=\jobname.png}]{standalone}
```

Trong đó bạn có thể thay định dạng file ảnh mà bạn muốn như jpg, svg ...

- **Bước 6:** Nhấn biên dịch và hưởng thành quả.

Bạn cũng cần chú ý rằng cách thiết lập này hoàn toàn hiệu quả với các trình soạn thảo như **TexMaker** và **Texstudio**. Riêng với **Vietex** thì tôi chưa tìm ra cách cấu hình cho phù hợp mặc dù cũng đã từng thử qua. Tất nhiên bạn có thể cài song song vài trình soạn thảo trên máy tính vì những trình soạn thảo này không hề làm tăng dung lượng đáng kể cho máy tính của bạn (khoảng vài chục Mb với mỗi trình soạn thảo thì quá đơn giản).

V.2 Làm ảnh động trực tiếp trên PDF

Bạn muốn tạo những ảnh động trên file PDF trực tiếp bằng **Tikz**? Bạn hãy dùng gói **animate**. Trước hết bạn gọi gói lệnh làm ảnh động bằng việc sử dụng `\usepackage{animate}`. Sau đó là bạn tạo code vẽ.

Cấu trúc của code bao gồm:

```
\begin{animateinline}[controls,autoplay,loop]{s hình/giây}
\multiframe{\n}{i=0+1}{
\begin{tikzpicture}
.....
...Lnh v hình...
.....
\end{tikzpicture}
}
\end{animateinline}
```

Một vài tùy chọn của môi trường **animate**:

- **controls**: Hiển thị bảng điều khiển (play,pause, stop ...)
- **autoplay**: Hình tự động chạy khi mở file PDF.
- **loop**: Chạy lặp đi lặp lại không dừng.
- **palindrome**: Chạy xuôi, ngược.
- **số hình/giây**: là một con số cụ thể quy định sự nhanh/chậm của chuyển động hình ảnh.
- **multiframe**: Biến chạy i chạy từ 0 đến n theo quy luật $i = i + 1$

Trong phần lệnh vẽ hình, nếu code hình của bạn không có sự tham gia của biến i thì hình vẽ của bạn sẽ luôn đúng yên (tác động của sự thay đổi biến i làm cho thay đổi hình ảnh).

Ngoài ra bạn cần chú ý rằng, khi tạo ảnh động do việc thay đổi hình ảnh nên khung hình sẽ có sự thay đổi. Khi đó bạn muốn ảnh chạy mà khung hình đứng yên bạn cần khéo léo clip khung hình ngay trước tất cả các lệnh vẽ. Tôi sẽ làm một ví dụ đơn giản như sau:

```
\begin{animateinline}[controls,autoplay,loop,palindrome]{5}
\multiframe{90}{i=0+1}{
\begin{tikzpicture}
\clip (-10:3) arc (0:190:3)--cycle;
\draw[-stealth,thick] (180:2.5)--(0:2.5);
\draw[-stealth,thick] (0:0)--(90:2.5);
\draw[blue,thick] (0:2) arc (0:2*\i : 2);
\end{tikzpicture}
}
\end{animateinline}
```

và ta sẽ được một ảnh động vẽ nửa đường tròn bán kính 2 đơn vị:

Bạn muốn xem được hình động trực tiếp trên PDF thì bạn cần phải cài đặt Adobe AcrobatReader nhé. Các trình xem PDF khác không hỗ trợ bạn điều này.

V.3 Làm ảnh động gif để upload lên web

Ngoài việc bạn làm ảnh động chạy trực tiếp trên file PDF, bạn có thể nhờ vào **ImageMagick** để tạo ảnh động định dạng gif mà bạn thường thấy trên web.

Về bản chất ảnh gif chỉ là tập hợp các tấm ảnh xuất hiện liên tục. Do đó bạn cần tạo nhiều ảnh liên tục rồi nhờ **ImageMagick** tạo ảnh gif cho bạn mà thôi. Như vậy trước hết bạn cần tạo một file PDF có nhiều trang, mỗi trang là một hình ảnh, từ đó bạn dùng lệnh convert của **ImageMagick**.

Để tạo một file PDF có nhiều trang liên tục, chẳng hạn như hình bạn vừa vẽ, bạn sẽ có code như sau:

```
\documentclass[tikz, border=5mm]{standalone}
\begin{document}
\foreach \i in {0, ..., 90} {
\begin{tikzpicture}
\clip (-10:3) arc (0:190:3)--cycle;
\draw[-stealth, thick] (180:2.5)--(0:2.5);
\draw[-stealth, thick] (0:0)--(90:2.5);
\draw[blue, thick] (0:2) arc (0:2*\i : 2);
\end{tikzpicture}
}
\end{document}
```

Khi bạn biên dịch file này, bạn sẽ có một file PDF gồm 90 trang, mỗi trang là một hình ứng với một giá trị của biến i . Bây giờ bạn hãy đặt nó ở đâu đó trong máy tính và mở cmd của window lên (Nhấn nút start rồi chọn run, gõ cmd). Bạn sẽ có cửa sổ cmd như sau:



Bạn sử dụng dòng lệnh sau:

```
imgconvert convert -density 300 -delay 10 -alpha remove Duong dan den file
pdf Duong dan den file gif
```

Trong dòng lệnh trên, ta thấy một vài thông số cần hiểu:

- **imgconvert convert**: Lệnh chuyển đổi.
- **-density 300**: Chất lượng hình ảnh, thông thường để con số 300 là hợp lý. Con số này càng lớn thì thời gian convert càng lâu đồng nghĩa với chất lượng hình ảnh càng tốt.
- **-delay 10**: Thời gian chuyển hình ảnh (độ trễ). Như ở đây cứ 10/100 giây sẽ chuyển một hình. Con số càng tăng thì hình càng chạy chậm và ngược lại.
- **-alpha remove**: Lệnh tạo gif và xóa bỏ những thiết đặt tạm sau khi chuyển đổi. Khuyến cáo không thay đổi chỗ này.
- **Đường dẫn đến file pdf**: Chẳng hạn tôi đặt file **Hinh01.pdf** của tôi trong thư mục **HinhGif** của ổ D thì tôi sẽ đặt đường dẫn này là:
D:\HinhGif\Hinh01.pdf
- **Đường dẫn đến file hình gif**: Tương tự, tôi muốn convert file **Hinh01.pdf** thành **Hinh01.gif** và đặt vào cùng thư mục file PDF thì tôi sẽ đặt đường dẫn này là:
D:\HinhGif\Hinh01.gif

Vậy là tôi có thể đặt một dòng lệnh như sau:

```
imgconvert convert -density 300 -delay 10 -alpha remove D:\HinhGif\Hinh01.pdf
D:\HinhGif\Hinh01.gif
```

Xong xuôi việc đặt lệnh. Việc còn lại là copy dòng lệnh này vào cmd và nhấn **Enter**.



Chờ đợi cho đến khi cmd làm việc xong thì ta sẽ có một file ảnh gif để sử dụng.



Lời khuyên cho bạn là việc nhớ dòng lệnh trên khá khó nên bạn cứ copy rồi dán dòng lệnh trên vào một file txt rồi lưu lại cho thuận tiện, khi muốn tạo hình gif thì mở ra thay đổi thông số cho phù hợp yêu cầu rồi copy, paste cho khỏi dính lỗi sai lệnh.

VI Các thư viện thường dùng

Trong việc vẽ hình **Tikz**, bạn có thể cần một số thư viện sẵn có của Tikz để việc vẽ đơn giản và hiệu quả hơn, chúng ta có thể tìm hiểu một số thư viện thường dùng nhất.

Cú pháp để gọi thư viện của Tikz như sau:

```
\usetikzlibrary{calc,angles,intersections,patterns,decorations.text,shadings}
```

Trong cú pháp gọi thư viện trên, **calc**, **angles**, **intersections** ... là tên các thư viện ta cần gọi ra. Khi cần gọi nhiều thư viện bạn chú ý chỉ cần dùng một lệnh **\usetikzlibrary{ }** và liệt kê tên các thư viện trong cặp ngoặc nhọn (cách nhau bởi dấu “phẩy”) chứ không cần thiết dùng quá nhiều lệnh gọi nhé.

VI.1 calc

Thư viện cung cấp cho ta các phép tính đối với tọa độ của Tikz. Thư viện này được sử dụng hàng đầu khi vẽ hình. Đặc biệt là các bài hình học phẳng. Khi chúng ta sử dụng đến các mối liên quan giữa các điểm. Chẳng hạn muốn khai báo trung điểm của đoạn thẳng AB ta sử dụng lệnh

```
\path ($(A)! .5! (B)$) coordinate (A);
```

Khi đó nếu bạn không gọi thư viện **calc** thì Tikz sẽ báo lỗi. Như vậy, khi bạn muốn dùng các phép tính tọa độ thì việc gọi thư viện **calc** là bắt buộc.

VI.2 intersections

Thư viện để tính toán và tìm tọa độ giao điểm giữa các đường bằng cách đặt tên đường với tùy chọn **name path** và lệnh lấy giao điểm bằng tùy chọn **name intersections**.

Chú ý rằng riêng việc lấy giao điểm của hai đường thẳng ta hoàn toàn không cần gọi thư viện **intersections**. Tuy nhiên việc sử dụng thư viện **intersections** cũng có những hạn chế chỉ lấy được những giao điểm “thực” giữa các đường mà thôi (nói cách khác là nó chỉ lấy được giao điểm của các đoạn). Còn khai báo

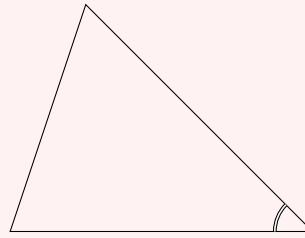
```
(intersection of A-B and C-D) coordinate M
```

sẽ lấy được giao điểm của hai đường thẳng cho dù đoạn AB và CD có cắt nhau thực hay không. Lệnh này lấy chính xác giao điểm của hai đường đi qua hai đoạn thẳng đó. Do vậy bạn cũng nên cân nhắc khi nào dùng khai báo trực tiếp, khi nào gọi thư viện và đặt tên đường. Khuyên cáo là khi cần giao điểm của hai đường thẳng thì nên dùng trực tiếp, còn các đối tượng khác (đường thẳng và đường cong, đường cong và đường cong) thì nên gọi thư viện.

VI.3 angles

Thư viện cho ta các ký hiệu góc trong hình học phẳng. Chẳng hạn bạn xem ví dụ sau:

```
\begin{tikzpicture}
\draw (0,0) coordinate (A)
--(4,0) coordinate (B)
--(1,3) coordinate (C)
--cycle
;
\draw
pic[draw,double]{angle = C--B--A}
;
\end{tikzpicture}
```



Với **pic**, bạn có nhiều tùy chọn trong cặp ngoặc vuông, có thể là tùy chọn mũi tên **->**, tùy chọn màu **red, blue ...**, tùy chọn kiểu đường, độ đậm nhạt của đường, tùy chọn tô hoặc không (fill), ... Một chú ý khác là bạn cần nhớ đến thứ tự các điểm sẽ cho ta góc trong hoặc góc ngoài của tam giác trong ví dụ trên nhé.

Riêng ký hiệu vuông góc thì trước đây Tikz cũng đã có nhưng sau đó phát sinh một số lỗi. Phiên bản hiện nay không có ký hiệu này, bạn hoàn toàn có thể tạo macro riêng cho mình bằng vài lệnh đơn giản hoặc bạn vẽ trực tiếp.

Tất nhiên là các ký hiệu này cũng chỉ có một vài mẫu cố định, còn nếu bạn muốn tùy biến cách ký hiệu theo ý bạn thì tôi gợi ý cho bạn tạo macro để biểu diễn ký hiệu góc cho thoải mái.

VI.4 patterns

Thư viện dùng để tô nền một miền phẳng nào đó giới hạn bởi các đường. Phần này tôi đã giới thiệu khá kỹ trong phần tô miền đồ thị hàm số.

VI.5 Decoration

Một thư viện cũng khá thường dùng khi vẽ các hình minh họa, đặc biệt là các hình vẽ bài toán thực tế. Chúng ta thường dùng nhất là **decorations.pathmorphing** và **decorations.text**. Bạn có thể ít dùng **decorations.pathmorphing** hơn nên phần này tôi dành cho các bạn tra cứu pgfmanual.pdf

Với **decorations.text**, bạn sẽ thường dùng hơn khi bạn muốn hiển thị chữ chạy theo một đường path nào đấy. Chẳng hạn bạn muốn dòng chữ chạy theo một đường tròn:

```
\begin{tikzpicture}
\draw[decoration={text along path,
    text={M{}t d{\circ}ng ch{} ch{}y
    theo n{}a {}ng
    tr{\circ}n{}},decorate}] (0,0) arc
(180:0:3);
\end{tikzpicture}
```

Một dòng chữ chạy theo nửa đường tròn

Bạn cần chú ý rằng, trong tùy chọn **text along path** thì dòng chữ cần hiển thị là chữ tiếng Anh hoặc tiếng Việt không dấu bạn có thể gõ bình thường, nhưng nếu như bạn muốn hiển thị tiếng Việt, tốt nhất bạn đặt cả cụm từ tiếng Việt trong cặp dấu ngoặc nhọn thì Tikz mới biên dịch cho bạn. Nếu bạn không để ý đến chú ý này thì việc biên dịch của bạn sẽ gặp lỗi.

Bạn muốn thêm các tùy chọn khác cho dòng chữ của bạn đẹp hơn. Vui lòng tra cứu thêm pgfmanual.pdf

VI.6 Shadings

Thư viện giúp bạn tô màu, pha trộn màu hoàn hảo hơn. Với thư viện này bạn hoàn toàn có thể tùy biến tô màu khá đẹp và đảm bảo tính phức tạp của miền được tô màu. Chúng ta có một số tùy chọn:

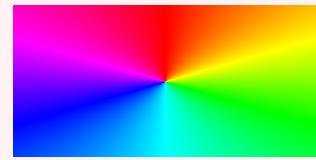
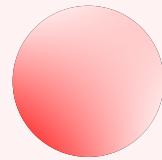
- **top color:** Màu trên đỉnh.
- **bottom color:** Màu dưới chân.
- **middle color:** Màu ở giữa.
- **left color:** Màu bên trái.
- **right color:** Màu bên phải.
- **ball color:** Tô màu dạng cầu.
- **lower left:** Góc trái bên dưới.
- **upper left:** Góc trái bên trên.
- **upper right:** Góc phải bên trên.
- **lower right:** Góc phải bên dưới.
- **color wheel:** Dạng đĩa màu.
- **inner color:** Tô từ trong ra.
- **outer color:** Tô từ ngoài vào.

Với các tùy chọn trên, bạn cần nhập tên màu với mỗi tùy chọn để kiểm chứng và học hỏi. Tất nhiên bạn có thể phối trộn cùng lúc vài ba tùy chọn để việc tô màu của bạn hoàn hảo hơn.

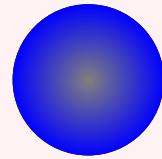
```
\begin{tikzpicture}
\fill[ball color=red] (0,0) circle
(1);
\shade[top color=blue,bottom
color=orange,middle color=white]
(2,-1) rectangle (6,1);
\end{tikzpicture}
```



```
\begin{tikzpicture}
\fill[lower left=red,upper
right=pink] (0,0) circle (1);
\shade[shading=color wheel] (2,-1)
rectangle (6,1);
\end{tikzpicture}
```



```
\begin{tikzpicture}
\fill[outer color=blue] (0,0)
circle (1);
\shade[inner color=yellow] (2,-1)
rectangle (6,1);
\end{tikzpicture}
```



Chắc rằng với vài ba ví dụ trên, bạn sẽ rút ra việc sử dụng các tùy chọn để tô màu hình vẽ của mình cho hợp lý.

Tất nhiên rằng **Tikz** còn có nhiều thư viện khác, nếu bạn cần bạn hãy tra cứu để biết thêm. Một vài thư viện tôi liệt kê trên đây chỉ là những thư viện thường dùng nhất khi vẽ hình mà thôi.