How to draw these (closed contours) diagrams using TikZ or PSTricks?

by sagodev

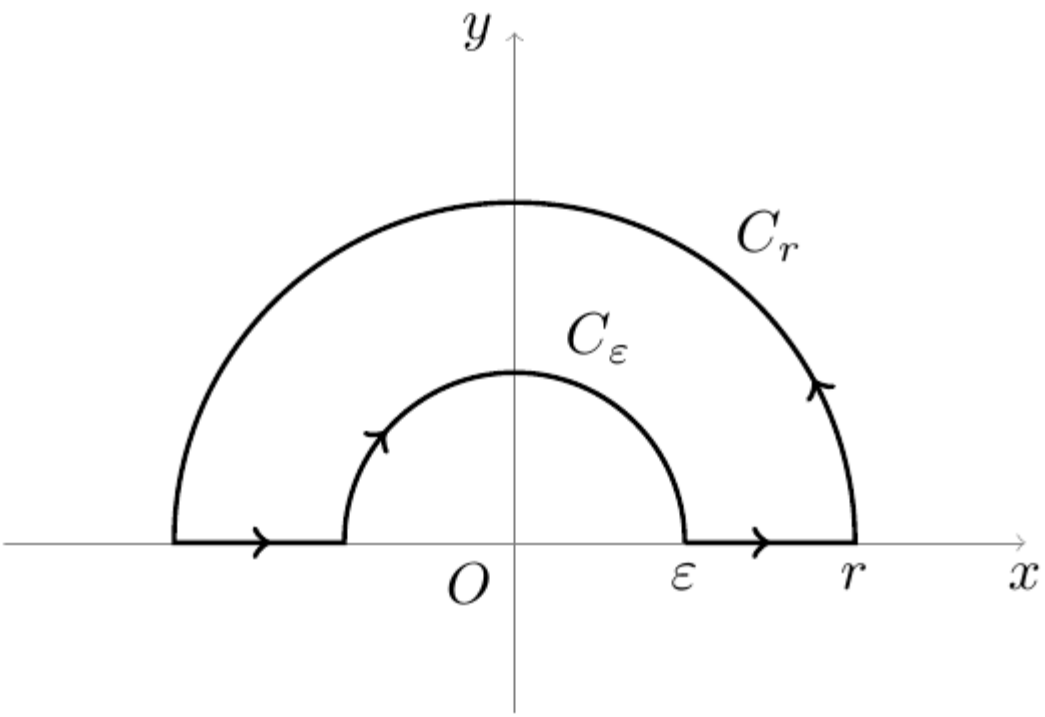
Question: **How to draw these (closed contours) diagrams using TikZ or PSTricks?** by a member on SagoDEV.com.

THE BEST ANSWERS COMPILED BY SAGODEV.COM TO THE QUESTION: How to draw these (closed contours) diagrams using TikZ or PSTricks?

The first one:

```
documentclass{article}
usepackage{tikz}
usetikzlibrary{decorations.markings}

begin{document}

begin{tikzpicture}[decoration={markings,
mark=at position 0.5cm with {arrow[line width=1pt]{>}},
mark=at position 2cm with {arrow[line width=1pt]{>}},
mark=at position 7.85cm with {arrow[line width=1pt]{>}},
mark=at position 9cm with {arrow[line width=1pt]{>}}
}
]
```

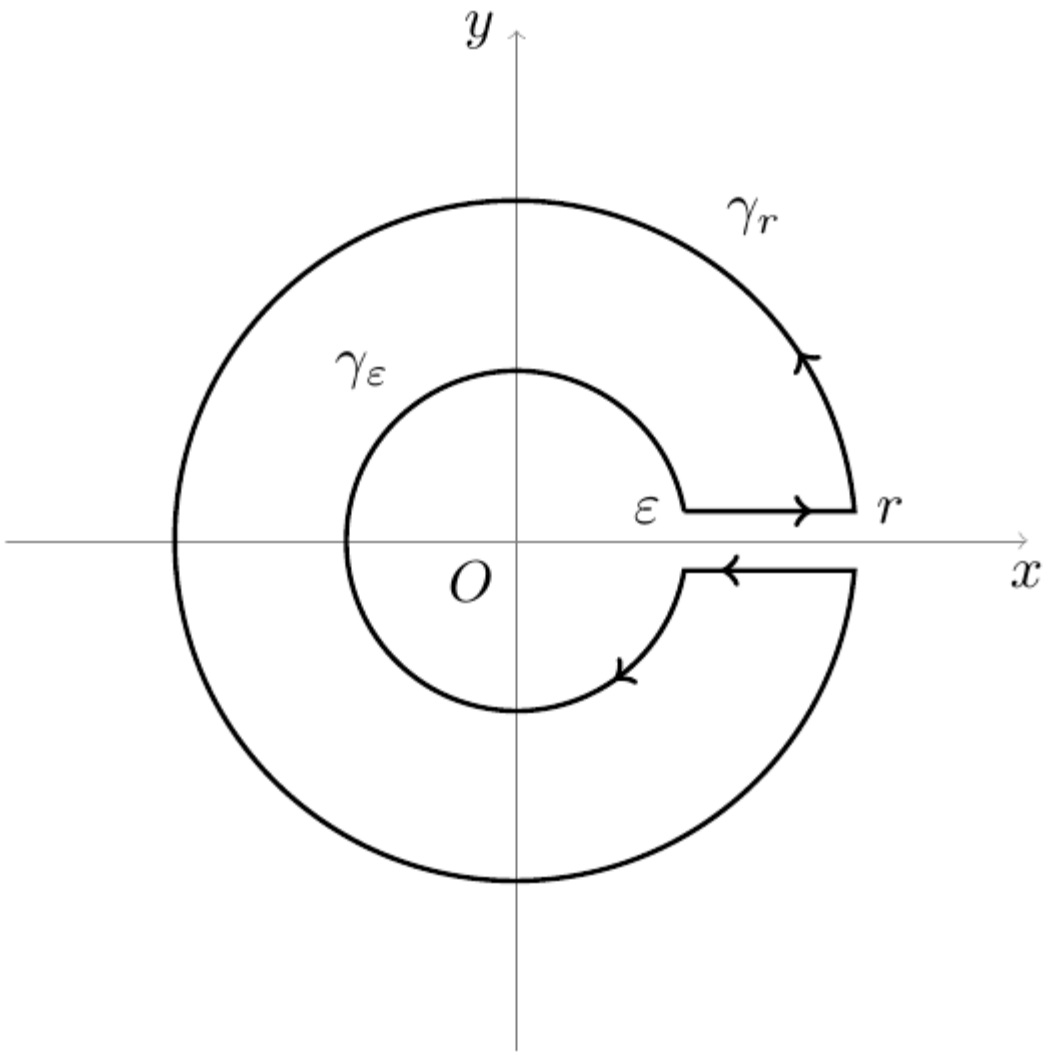
```
% The axes
draw[help lines,->] (-3,0) -- (3,0) coordinate (xaxis);
draw[help lines,->] (0,-1) -- (0,3) coordinate (yaxis);

% The path
path[draw,line width=0.8pt,postaction=decorate] (1,0) node[below] {$vareps

% The labels
node[below] at (xaxis) {$x$};
node[left] at (yaxis) {$y$};
node[below left] {$O$};
node at (0.5,1.2) {$C_{varepsilon}$};
node at (1.5,1.8) {$C_{r}$};
end{tikzpicture}

end{document}
```



The second one:

```
documentclass{article}
usepackage{tikz}
```

```
usetikzlibrary{decorations.markings}

begin{document}

begin{tikzpicture}
[decoration={markings,
mark=at position 0.75cm with {arrow[line width=1pt]{>}},
mark=at position 2cm with {arrow[line width=1pt]{>}},
mark=at position 14cm with {arrow[line width=1pt]{>}},
mark=at position 15cm with {arrow[line width=1pt]{>}}
}
]
% The axes
draw[help lines,->] (-3,0) -- (3,0) coordinate (xaxis);
draw[help lines,->] (0,-3) -- (0,3) coordinate (yaxis);

% The path
path[draw,line width=0.8pt,postaction=decorate] (10:1) node[left] {$vareps

% The labels
node[below] at (xaxis) {$x$};
node[left] at (yaxis) {$y$};
node[below left] {$O$};
node at (-0.9,1) {$gamma_{varepsilon}$};
node at (1.4,1.9) {$gamma_{r}$};
end{tikzpicture}

end{document}
```



For those who don't know how to compile PSTricks codes: Compile each of the following codes with either a combo (much faster) `latex` followed by `dvips` followed by `ps2pdf` or a single run (much slower) of `xelatex` to get a PDF output. Once you get PDF images, you can import these PDF images within your main TeX input file using `includegraphics{filename}`. The main TeX input file must be compiled with pdflatex (faster) or `xelatex` (much slower).

First Diagram

```
documentclass[pstricks,border={2pt 5pt 13pt 13pt}]{standalone}
usepackage{pstricks-add}

begin{document}

begin{pspicture}(-3,-0.25)(3,3)
% draw cartesian axes
psaxes[ticks=none,labels=none,linecolor=lightgray]{->}(0,0)(-3,-0.25)(3,3)
% global setting
psset{linecap=2}
% draw the outer arc
psarc[arcsepB=-3pt]{->}(0,0){2.5}{0}{60}
psarc(0,0){2.5}{60}{180}
% draw the left line
psline[ArrowInside=->](-2.5,0)(-1.5,0)
% draw the inner arc
psarcn[arcsepB=-3pt]{->}(0,0){1.5}{180}{150}
psarcn(0,0){1.5}{150}{0}
% draw the right lint
psline[ArrowInside=->](1.5,0)(2.5,0)
% draw label
uput[60](2.5;60){$C_r$}
uput[80](1.5;80){$C_varepsilon$}
uput[-135](0,0){$O$}
end{pspicture}

end{document}
```

Second Diagram

```
documentclass[pstricks,border={2pt 2pt 13pt 13pt}]{standalone}
usepackage{pstricks-add}
defh{0.2}
```

```
begin{document}

begin{pspicture}(-3,-3)(3,3)
% draw cartesian axes
psaxes[ticks=none,labels=none,linecolor=lightgray]{->}(0,0)(-3,-3)(3,3)[$x
% global setting
psset{linecap=2}
% declare nodes
pnode(!2.5 2 exp hspace 2 exp sub sqrt h){A}
pnode(!1.5 2 exp hspace 2 exp sub sqrt h){D}
pnode(!2.5 2 exp hspace 2 exp sub sqrt hspace neg){B}
pnode(!1.5 2 exp hspace 2 exp sub sqrt hspace neg){C}
% draw the outer arc
psarc[arcsepB=-3pt]{->}(0,0){2.5}{(A)}{60}
psarc(0,0){2.5}{60}{(B)}
% draw the bottom line
psline[ArrowInside=->](B)(C)
% draw the inner arc
psarcn[arcsepB=-3pt]{->}(0,0){1.5}{(C)}{300}
psarcn(0,0){1.5}{300}{(D)}
% draw the top line
psline[ArrowInside=->](D)(A)
% draw label
uput[60](2.5;60){$gamma_r$}
uput[0](A){$r$}
uput[150](1.5;150){$gamma_varepsilon$}
uput[45](D){$varepsilon$}
uput[-135](0,0){$O$}
end{pspicture}

end{document}
```

Miscellaneous

```
documentclass[pstricks,border={2pt 2pt 13pt 13pt}]{standalone}
```

```
usepackage{pstricks-add}


begin{document}
multido{n=0.1+0.1}{10}{
begin{pspicture}(-3,-3)(3,3)
% draw cartesian axes
psaxes[ticks=none,labels=none,linecolor=lightgray]{->}(0,0)(-3,-3)(3,3)[$x
% global setting
psset{linecap=1}
% declare nodes
pnode(!2.5 2 exp nspace 2 exp sub sqrt n){A}
pnode(!1.5 2 exp nspace 2 exp sub sqrt n){D}
pnode(!2.5 2 exp nspace 2 exp sub sqrt nspace neg){B}
pnode(!1.5 2 exp nspace 2 exp sub sqrt nspace neg){C}
%
pscustom*[linecolor=lightgray]{psarc(0,0){2.5}{(A)}{(B)}psline(B)(C)psarcn
% draw the outer arc
psarc[arcsepB=-3pt]{->}(0,0){2.5}{(A)}{60}
psarc(0,0){2.5}{60}{(B)}
% draw the bottom line
psline[ArrowInside=->](B)(C)
% draw the inner arc
psarcn[arcsepB=-3pt]{->}(0,0){1.5}{(C)}{300}
psarcn(0,0){1.5}{300}{(D)}
% draw the top line
psline[ArrowInside=->](D)(A)
% draw label
uput[60](2.5;60){$gamma_r$}
uput[0](A){$r$}
uput[150](1.5;150){$gamma_varepsilon$}
uput[45](D){$varepsilon$}
uput[-135](0,0){$O$}
end{pspicture}}

end{document}
```

Here three solutions.

With Tikz we have several problems. First we need to use some angles to draw arcs and it's not easy if you don't know some mathematics notions like asin and atan2, then there is the problem to draw arrow at specific places.
The first solutions are based on tkz-euclide. The problem with tkz-eucide is the syntax base on pst-eucl and latex. I understand that a lot of users prefer to use only tikz. The main problem is that tkz-euclide is not very flexible and it's not easy to extend the commands. The last point is the notion of path, we can't use this notion as in tikz.

A fine solution is the last one based only on tikz.

It's possible to use tkz-euclide. The solution uses the same way as pst-eucl, because we can draw an arc from one point in the direction of another point. We don't need to calculate angles

1) I define four points B, C and D,E and I draw the arc with center O from B to C and from D to E.

```
documentclass{article}
usepackage{tkz-euclide}
usetkzobj{all}
begin{document}

begin{tikzpicture}
  tkzInit[xmin=-5,ymin=-5,xmax=5,ymax=5]
  tkzDrawXY[noticks]
   tkzDefPoint(0,0){O}
   tkzDefPoint(.5,.2){B} tkzDefPoint(.5,-.2){C}
   tkzDefPoint(4,.2){D} tkzDefPoint(4,-.2){E}
  tkzDrawArc[color=red,line width=1pt](O,B)(C)
  begin{scope}[decoration={markings,
      mark=at position .5 with {arrow[scale=2]{>}};}]
    tkzDrawSegments[postaction={decorate},color=red,line width=1pt](B,D E,
```

```
  end{scope}
  begin{scope}[decoration={markings,
     mark=at position .20 with {arrow[scale=2]{>}},
     mark=at position .70 with {arrow[scale=2]{>}};}]
    tkzDrawArc[postaction={decorate},color=red,line width=1pt](O,D)(E)
  end{scope}
end{tikzpicture}
end{document}
```

2) Always with tkz-euclide but I don't use here the *decoration library* because it's not easy to place the arrow. Here I draw paths with the option `->`. I need to cut some paths in small paths

```
documentclass{article}
usepackage{tkz-euclide}
usetkzobj{all}
begin{document}

begin{tikzpicture}
  tkzInit[xmin=-5,ymin=-5,xmax=5,ymax=5]
  tkzDrawXY[noticks]
  tkzDefPoint(0,0){O}
  tkzDefPoint(3:4){D} tkzDefPoint(90:4){M} tkzDefPoint(270:4){N}
  tkzDefPoint(-3:4){E}
  tkzDefPoint[shift={(-3.5,0)}](3:4){B} tkzDefMidPoint(B,D) tkzGetPoint{B'
  tkzDefPoint[shift={(-3.5,0)}](-3:4){C} tkzDefMidPoint(C,E) tkzGetPoint{C

 tkzDrawArc[color=red,line width=1pt](O,N)(E)
 tkzDrawArc[color=red,line width=1pt](O,B)(C)
 tikzset{compass style/.append style={->}}
 tkzDrawArc[color=red,line width=1pt](O,D)(M)
 tkzDrawArc[color=red,line width=1pt](O,M)(N)

tkzDrawSegments[color=red,line width=1pt,->](D,B' C,C')
tkzDrawSegments[color=red,line width=1pt](B',B C',E)
```

```
end{tikzpicture}

end{document}
```

3) The last solution is to use tikz and to define a new macro to get polar coordinates of the last point. I named this macro `pgfgetlastar` angle for a, and r for radius.

The code of the macro

```
defpgfgetlastar#1#2{%
  pgfmathparse{veclen(pgf@x,pgf@y)/28.45274}
    edef#1{pgfmathresult}%
    pgfmathparse{atan2(pgf@x,pgf@y)}
    edef#2{pgfmathresult}%
}%
```

With `veclen` I get the length of OM if M is the last point used in the path and O the origin. atan2 gives the angle of OM with the horizontal axe.

Now the next code is to use the macro in the options of a path

```
tikzset{
    last polar/.code 2 args=
     {pgfgetlastar{#1}{#2} }
    }
```

The macro in action : We draw an arc then a horizontal line. We determine the polar coordinates of the last point before to draw the last arc and the last line.

```
begin{tikzpicture}[deco]
draw[red,postaction=decorate]
      (4:4 cm) arc (4:356:4 cm) -- +(-3,0) [last polar={r}{a}] arc (a:-360
end{tikzpicture}
```

We only need to define the decoration :

The complete code :

```
documentclass{article}
usepackage{tikz}
usetikzlibrary{decorations.markings,arrows}

makeatletter
defpgfgetlastar#1#2{%
  pgfmathparse{veclen(pgf@x,pgf@y)/28.45274}
    edef#1{pgfmathresult}%
    pgfmathparse{atan2(pgf@x,pgf@y)}
    edef#2{pgfmathresult}%
}%

begin{document}

tikzset{
    last polar/.code 2 args=
     {pgfgetlastar{#1}{#2} }
    }

tikzset{deco/.style= {decoration={markings,
      mark=at position .17 with {arrow[scale=2]{>}},
      mark=at position .51 with {arrow[scale=2]{>}},
      mark=at position .72 with {arrow[scale=2]{>}},
      mark=at position .95 with {arrow[scale=2]{>}}
      }}}

begin{tikzpicture}[deco]
```

```
  draw[red,postaction=decorate]
        (4:4 cm) arc (4:356:4 cm) -- +(-3,0) [last polar={r}{a}] arc (a:-360
  end{tikzpicture}
  end{document}
```

I'm in a complex mood today 😉

Here is a path drawn using small tikz library `avoidpath`.

```
  documentclass[border=7mm]{standalone}
  usepackage{mathtools}
  usepackage{tikz}
  usetikzlibrary{avoidpath}

  begin{document}
    begin{tikzpicture}
      draw
        (-3,0) edge[-latex] node[at end, right]{$operatorname{Re} z$} (3,0)
        (0,-3) edge[-latex] node[at end, right]{$operatorname{Im} z$} (0,3)
        (0,0)  node[scale=3](O){.}
      ;
      draw[avoid={pole={(O)}, neck=8mm}, with arrows, thick]
        (1.5,4mm)
          to[avoid={radius=2cm}]
        (1.5,-4mm)
          to[avoid={radius=1cm}]
        cycle;
    end{tikzpicture}
  end{document}
```

For precision I computed internally in the tikz code angles and distances.
Needs at least PGS 3.0

Here the code:

```
documentclass[12pt]{article}
usepackage{etex}
usepackage{pstricks-add}
usepackage{relsize}
usepackage{slashbox}
usepackage[toc,page]{appendix}
usepackage{amssymb,amsmath}
usepackage{cancel}
usepackage[stable]{footmisc}
usepackage{setspace}
usepackage{hyperref}
usepackage{verbatim}
usepackage{pgfplots}
usepackage{tikz}
usetikzlibrary{matrix}
usetikzlibrary{decorations.markings}
usetikzlibrary{calc}
usetikzlibrary{shapes}
usetikzlibrary{arrows.meta, bending}


begin{document}
begin{center}
  begin{tikzpicture}[scale=0.5, very thick, decoration={markings, mark=at
      position 0.8 with {arrow{>}}}]
      % coordinates for axis
    coordinate (A) at (-8,0);
```

```
coordinate[label=$x$] (B) at (8,0);
coordinate (C) at (0,-8);
coordinate[label=$y$] (D) at (0,8);

% axis
draw[-latex, line width=2pt, color=gray] (A)--(B);
draw[-latex, line width=2pt, color=gray] (C)--(D);
% origin
coordinate (O) at (0,0);

% eps, R small and large radius
defeps{0.5} % epsilon
defR{7} % epsilon
defangeps{30} % angle for epsilon
defangepsend{330} % angle for epsilon
pgfmathsetmacroxeps{eps*cos(angeps)}
pgfmathsetmacroyeps{eps*sin(angeps)}
pgfmathsetmacrosinangeps{yeps/R}
pgfmathsetmacroangR{asin(sinangeps)}
pgfmathsetmacroangRend{360-angR}
pgfmathsetmacroxR{R*cos(angR)}
pgfmathsetmacroyR{R*sin(angR)}
pgfmathsetmacroyRm{-yR}


 % points for ends of small circle
 coordinate (E) at (xeps, yeps);
 coordinate (F) at (xeps, -yeps);

 % points for ends of large circle
 coordinate (G) at (xR, yR);
 coordinate (H) at (xR, yRm);


% draw large circle with arrow inside the path
 pgfmathsetmacroangm{angR+40}
draw[line width=2pt, ->]  (G) arc (angR:angm:R);
draw[line width=2pt]  (G) arc (angR:angRend:R);

% draw small circle with arrow inside the path
```

```
      defangeps{-30} % angle for epsilon
      defangepsend{-180} % angle for epsilon
      defangepsendf{-330} % angle for epsilon
       pgfmathsetmacroangm{angepsend-60}
       draw[->, {->[length=5pt,bend]}, line width=2pt]  (F) arc (angeps:angm
       draw[line width=2pt]  (F) arc (angeps:angepsendf:eps);


       % points for ends of small circle
       pgfmathsetmacroxepsd{xeps-0.05};
       pgfmathsetmacroxRd{xR+0.07};
       coordinate (E2) at (xepsd, yeps);
       coordinate (F2) at (xepsd, -yeps);

       % points for ends of large circle
       coordinate (G2) at (xRd, yR);
       coordinate (H2) at (xRd, yRm);


      draw[postaction={decorate}, line width=2pt] (E2)--(G2);
      draw[postaction={decorate}, line width=2pt] (H2)--(F2);


    end{tikzpicture}
  end{center}

  end{document}
```

Now the figure below:

Thank you for viewing this post: **How to draw these (closed contours) diagrams using TikZ or PSTricks?** . We hope that the sharing of SagoDEV.com will help you.

You can view similar posts by selecting the tags below.

pstricks, tikz-pgf

< It doesn't hyphenate words ending with “—”

> hyperref Warning: Draft mode on

Recent Posts

What's the accepted "abbreviation" for the original iPhone?

Do any current (early 2012) Macs support Intel's Smart Response Technology (SRT)

Is there any way to view word documents within Safari on the Mac?

How to force coloured display of visited links in Safari 5+?

Low-resolution icons displayed in Dock and Launchpad