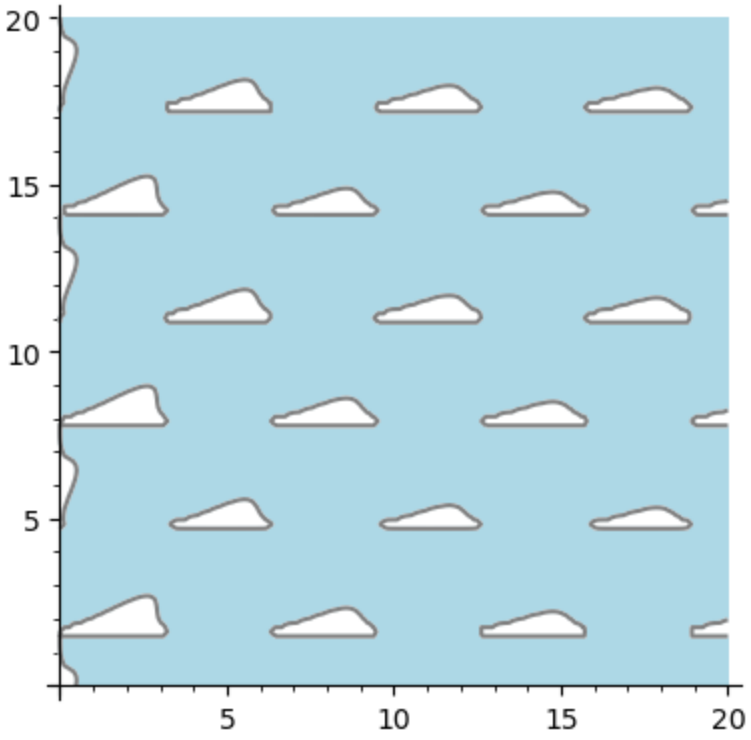
In [1]:
```
# Include these two lines of code at the beginning of every notebook you open.
# It will allow you to receive more than one output from a single code chunk/cell.
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

# MATH1110 Lab 2: Implicit plotting

Remember to run the chunk of code above as you get started!

In [3]:
```
x, y = var('x, y')
eq = (cos(x-y))^3/(e^tan(y))==sqrt(x)
implicit_plot(eq, (x, 0, 20), (y, 0, 20),\
              fill = True, fillcolor = 'lightblue', color = 'grey')
```

Out[3]:




**Objectives for today:**

1. Review from last week
2. Implicit plotting
3. Work on assignment if there's time

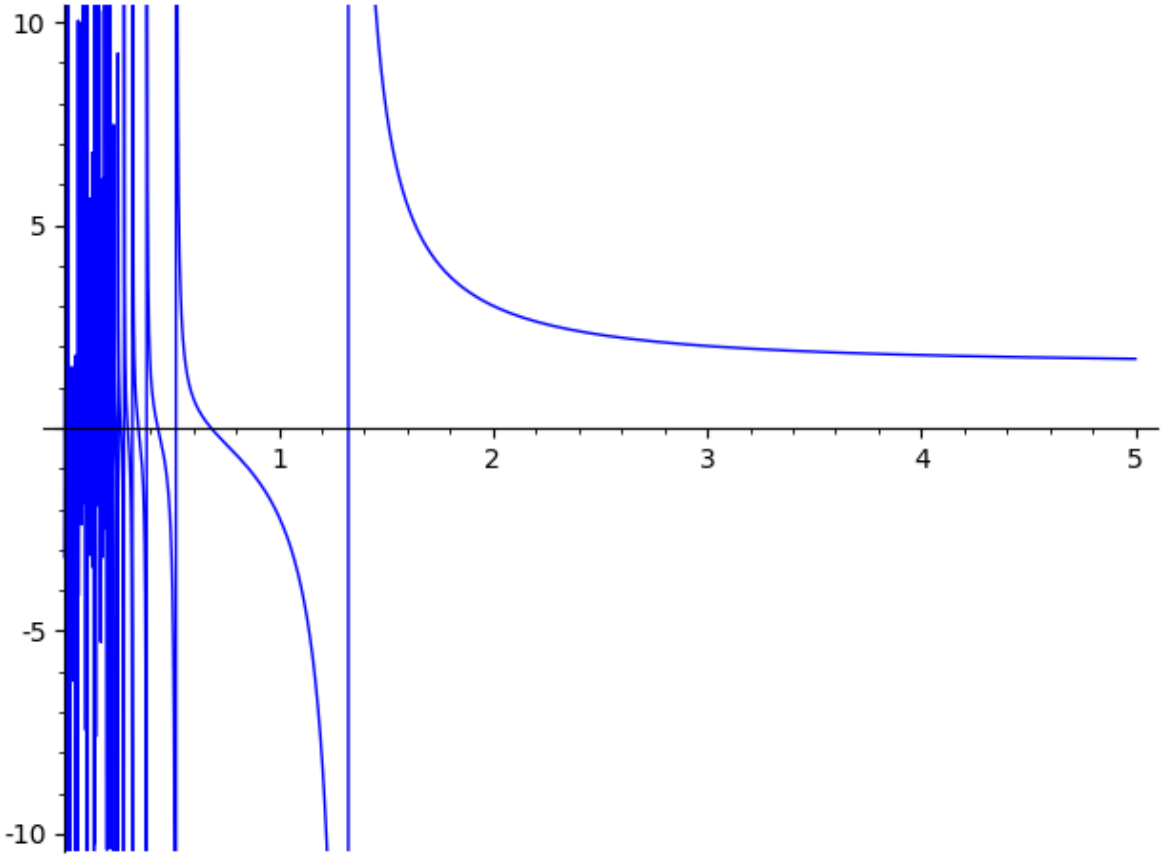
# 1. Review

Last week we covered a lot of ground! Let's review how to create functions and plot them in Sage.

We did an example with this function, and it's plot was pretty messy. What do you think is causing the vertical lines?

In [1]:
```
x = var('x')
y = tan(x^(-2)+1)
plot( y, (x, 0, 5), ymin = -10, ymax = 10)
```

Out[1]:




One other trick of the trade! You can layer plots, which makes them easier to compare. Let's try this out with $\sin(x)$ and $\cos(x)$ on the domain $-2\pi$ to $2\pi$. The first plot is already set up for you.

In [ ]:
```
g1 = plot(sin(x), x, -2*pi, 2*pi) # here we are storing the plots under the na
mes g1 and g2...
```

Not bad, but if you didn't know the y-intercepts of $\sin$ and $\cos$, you might get them mixed up. We can fix this by giving each curve a different colour. Add the argument *color* = '_'.

In [3]:
```
# feel free to copy and paste from above
```

To summarize, there are a lot of ways we can customize plots. Most of the time, it's not necessary to do much more than appropriately scale the x and y axes, and make sure in some way that different curves are distinguishable from each other in some way.

Here are some other ways to modify your plot, if you want to do that:

- *color* = *'_'* **Note that Sage uses the American spelling!** 'colour' is not recognized.
- *thickness* = #
- *linestyle* = *'_'* ... options include '-', ':', '--', '-.', or 'solid', 'dotted', 'dashed', 'dashdot'.
- *alpha* = # gives transparency of line. Value must be between 0 and 1.
- *title* = *'_'* puts a title at the top of your graph.

### Exercise question

1. Plot the three functions $e^x$, $(2e - x)^x$, and $(3e - x)^x$ together and give each a different colour.

In [ ]:

# 2. Implicit plotting

At the start of this document, you can see an example of an implicit plot. The *implicit_plot()* command is great for plotting curves that are implicitly defined, which means that the given variables are intertwined in an equation in a way that is not easily parsed apart, or the curve is not a properly defined function. This is the situation on Question 2 of your assignment.

The difference from the basic *plot()* is that now both x and y are defined as variables and your equation will require a proper equality (double **==**).

Plot the equation $\tan(x) = \frac{x}{\sin(y)}$ for x and y both within (-10, 10).

In [ ]:

You won't need this for the first assignment, but as a fun aside, Sage can also plot in polar coordinates. The arguments of *polar_plot()* look much the same as what we've done before, but the graphs are often quite loopy. Let's show $r = \arctan(\theta)$ with $0 < \theta < 4\pi$.

In [ ]:

### Exercise questions

1. Plot the curve $\ln\left(|t-3f|\right)=t$ in your favourite colour, choosing a domain for each variable that displays the curve well.

1. Plot the curve $y^5x^2-\frac{y^4}{x}+3y==x^3$.

In [ ]:

## 3. Any questions?

Feel free to spend the remaining time looking at your assignment.

In [ ]:

### Reminders

> Assignment 1 is due this friday! Submit a pdf of your code to Blackboard.
>
> Be sure to save your work and give your file a name.
>
> There are lots of great resources available on Blackboard. Please use them! Maya's office hours and email are on blackboard in an announcement.

In [ ]: