Resumo dos Comandos

- Lista de comandos a ser tratada
  - `Plot`
  - `Plot3D`
  - `ContourPlot`
  - `ParametricPlot`
  - `StreamPlot`
  - `VectorPlot`
  - `Show`

1 Plot

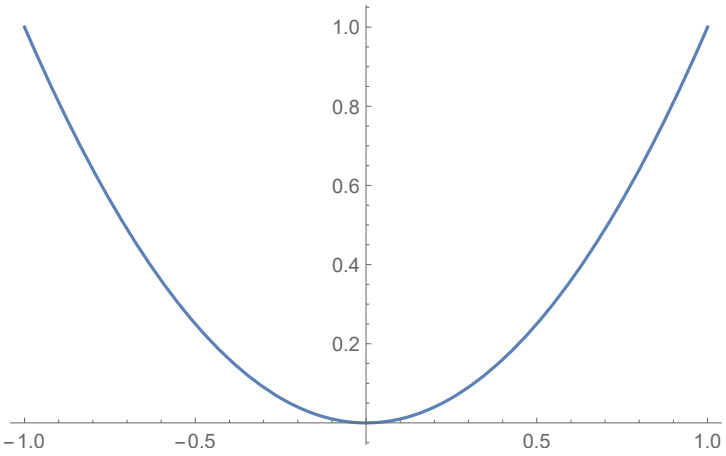
Normal recebe uma função que depende de somente uma variável seguida da definição do intervalo de amostragem desejado. Então imagine que se deseja plotar a função do segundo $f(x) = x^2$, para fazer isso basta tomar o lado direito da função e utilizar como primeiro parâmetro do `Plot` como segue.

```
Plot[x^2]
```

Porém, não é adequado parar por aqui, é necessário definir a variação de x em uma dimensão que vai abranger o "teatro" escolhido. Dessa maneira, imagine que queiramos plotar essa função de modo a abranger os valores de x que vão de -1 a 1, então aplicamos `{x,-1,1}`. Como consequência, percebe-se que o programa lança valores de x abrangendo o domínio explicitado de modo a conformar a representação gráfica no menor espaço possível de visualização do gráfico para os valores de x propostos

```
Plot[x^2,{x,-1,1}]
```

Da forma como está, se dermos `enter` o comando será rodado e vamos obter



Repare que sendo $x = -1$ ou $x = 1$, $f(x) = 1$, logo o máximo valor de f mostrado é 1, não precisamos a variação de y, até mesmo por que o `Plot` não aceitaria tal sintaxe.

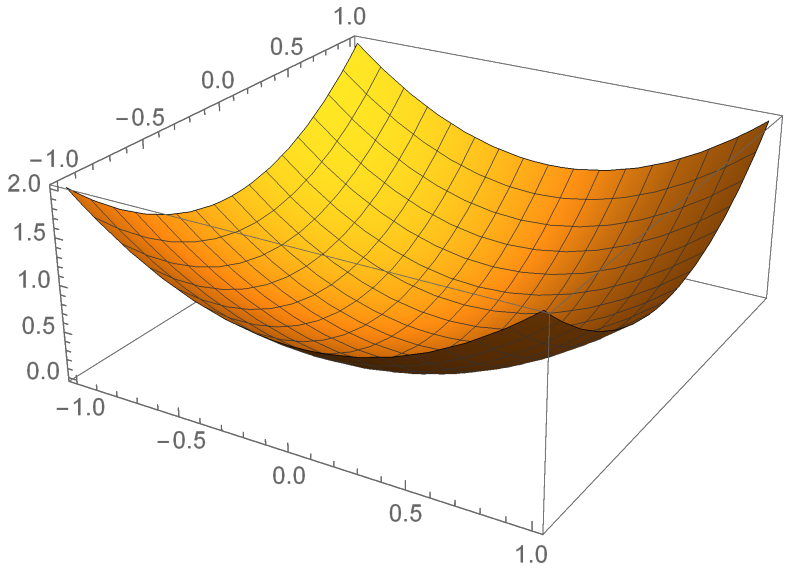
2 Plot3D

Podemos interpretar esse comando como expansão do `Plot`, porém em três dimensões. Sabendo que um ponto no espaço agora depende de x e y, consequentemente a função $f(x)$ em duas dimensões passará a ser $f(x, y)$. De forma análoga ao que foi feito para o comando anterior, imagine que queiramos plotar um paraboloide no espaço que obedeça

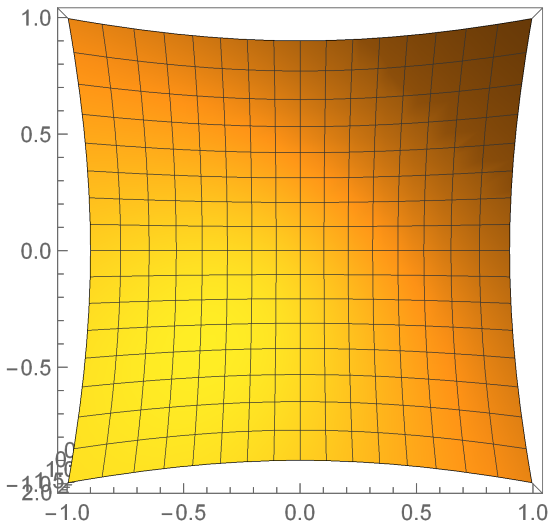
$$f(x, y) = x^2 + y^2 \tag{1}$$

Numa abordagem rápida, podemos imaginar tal superfície como a rotação de uma parábola tal como é visto em 2 em torno do eixo z no espaço, então sabendo que dependemos de mais uma variável será preciso acrescentar outro parâmetro ao `Plot3D` como segue

```
1  Plot3D[x^2+y^2,{x,-1,1},{y,-1,1}]
```

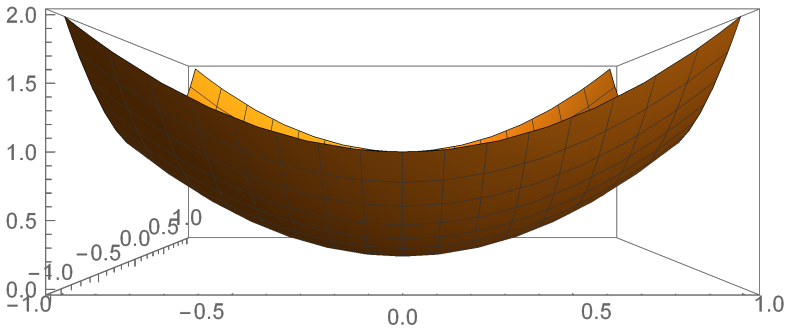


Note que se clicarmos com o botão direito do *mouse* e formos na opção *Top View* veremos



Ou seja, o intervalo de valores escolhido para x e y novamente é notado no domínio presente. Vê-se que a vista superior é um quadrado de lados valendo 2 unidades já que tanto x quanto y variam entre menos -1 e 1.

Partindo para análise em z, vamos mudar a opção de *Top View* para *Front View*. Fazendo uma análise visual vemos que a máxima altura da "caixa" que comporta o gráfico é de duas unidades. Isso ocorre pois o máximo valor que $f(x, y)$ pode assumir para o domínio escolhido ocorre quando as coordenadas são $(-1, -1)$, $(-1, 1)$, $(1, -1)$ ou $(1, 1)$ como vemos a seguir



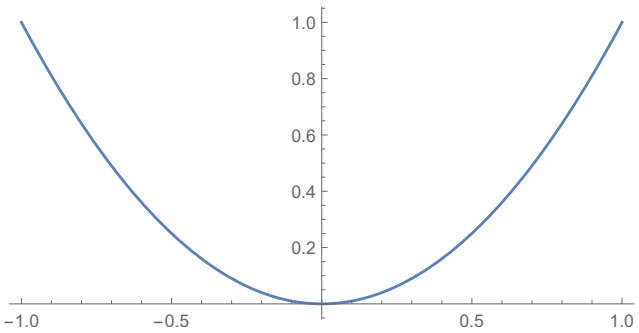
Sabemos disso devido a simplicidade do gráfico e prevemos que a função é sempre crescente no intervalo dado.

Alguns atributos de estilo podem ser passados para os `Plot`s de modo a melhorar a visualização ou visando cumprir alguma meta de representação gráfica. Alguns dos comandos a seguir são interessantes de serem entendidos pelo menos no básico.

Para o `Plot` (Gráficos em 2d no geral)

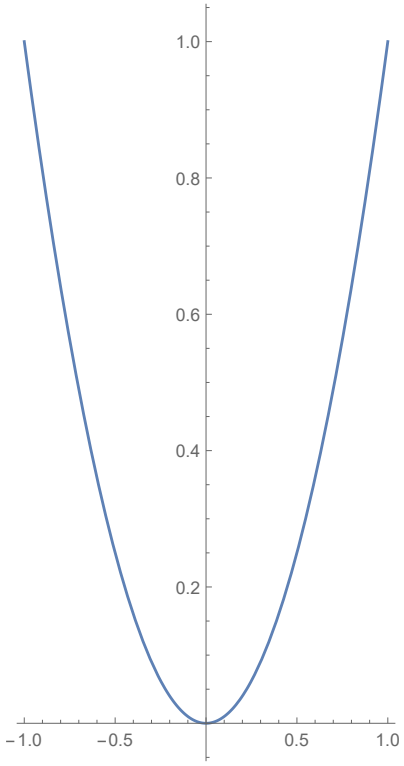
- `AspectRatio`
  - Permite editar a proporção entre os eixos do gráfico. Imagine que quiséssemos uma proporção de 1 em x para dois em y, teríamos uma razão igual a 0.5 e podemos prever que o gráfico terá maior comprimento em x que em y como vemos a seguir com base nas linhas de comando

```
Plot[x^2,{x,-1,1},AspectRatio -> .5]
```



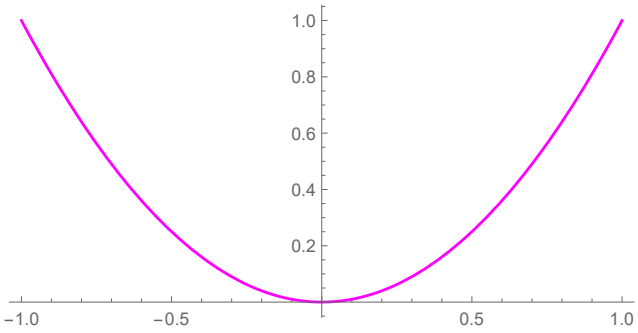
Trocando por 2 o valor da propriedade vemos

```
Plot[x^2,{x,-1,1},AspectRatio -> 2]
```

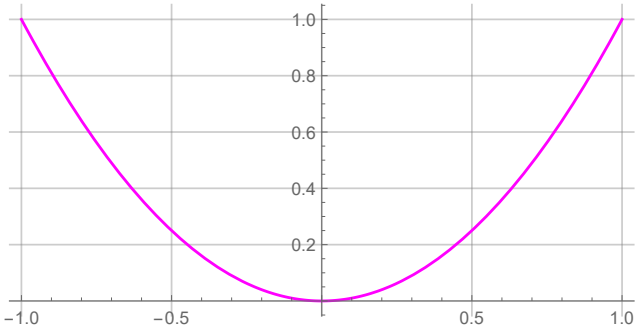


- `PlotStyle`
  - Um dos atributos mais utilizados, possibilita mudar a cor do gráfico, espessura das linhas, opacidade, entre muitos outros parâmetros.
    - Mudando a cor
      - Uma maneira aconselhável é passar diretamente o atributo de cor, como por exemplo `Magenta`, `Red`, `Blue`, `Cyan` ou utilizar o `RGBColor` para customizar com maior liberdade como segue

```
1 Plot[x^2,{x,-1,1},AspectRatio -> .5,PlotStyle -> Magenta]
```



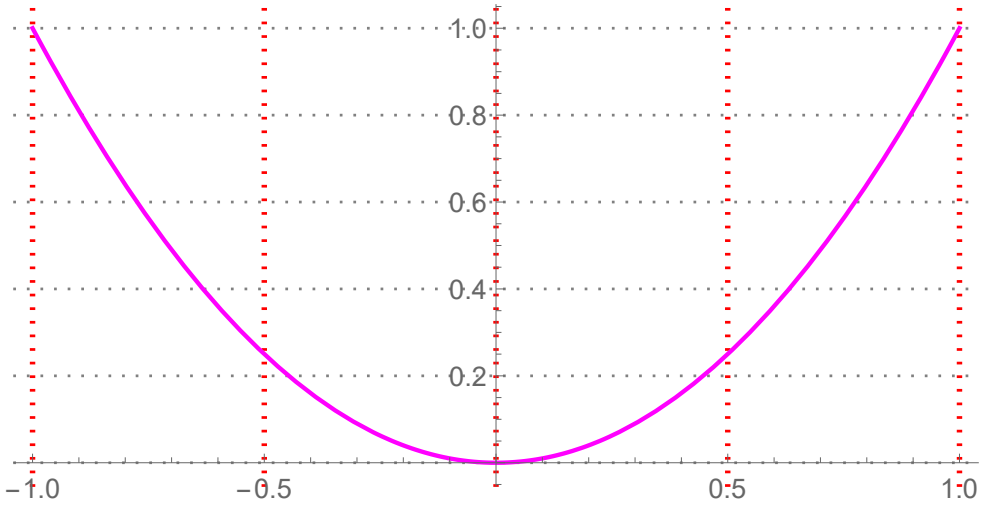
- `GridLines`
  - Nos permite adicionar uma malha quadriculada ao fundo da plotagem. Esse recurso muitas vezes facilita a associação entre os eixos coordenados e nos permite ter maior precisão quantitativa para aproximações feitas visualmente. Abaixo temos alguns exemplos.




```
Plot[x^2,{x,-1,1},AspectRatio->.5,PlotStyle->Magenta,GridLines
    ->Automatic]
```

Por padrão o `GridLines` com `Automatic` renderiza linhas cheias, mas podemos alterálas passando `GridLinesStyle->{{Dashed},{Dashed}}`. Isso fará com que tanto as linha horizontais quanto as verticais se tornem tracejadas. De forma semelhante usamos o `Dotted` para deixá-las pontilhadas. O atributo de cor também pode ser passado no interior das chaves como vemos a seguir

```
Plot[x^2,{x,-1,1},AspectRatio->.5,PlotStyle->Magenta,GridLines
    ->Automatic,GridLinesStyle->{{Red, Dotted, Thick},{Gray,
    Dotted}}}]
```

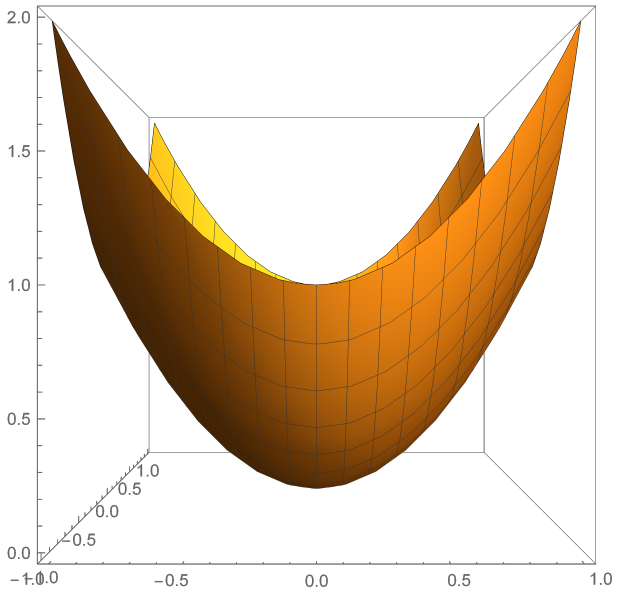


Aumentando-se a proporção do gráfico vê-se que a primeira lista de valores contendo `Red`, `Dotted` e `Thick` tornou as linhas verticais vermelhas, pontilhadas e, como foi passado o `Thick` houve uma leve aumentada na espessura, somente para melhorar a visualização do leitor.

Para o `Plot3D` (Gráficos em 3d no geral)

- `BoxRatios`
  - Esse atributo é análogo ao `AspectRatio`, porém fazemos sua aplicação para três dimensões com base numa lista de três posições indicando a proporção em cada eixo como segue

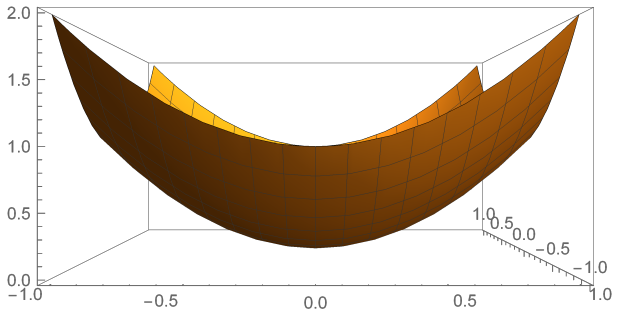
```
Plot3D[x^2+y^2,{x,-1,1},{y,-1,1},BoxRatios->{1,1,1}]
```



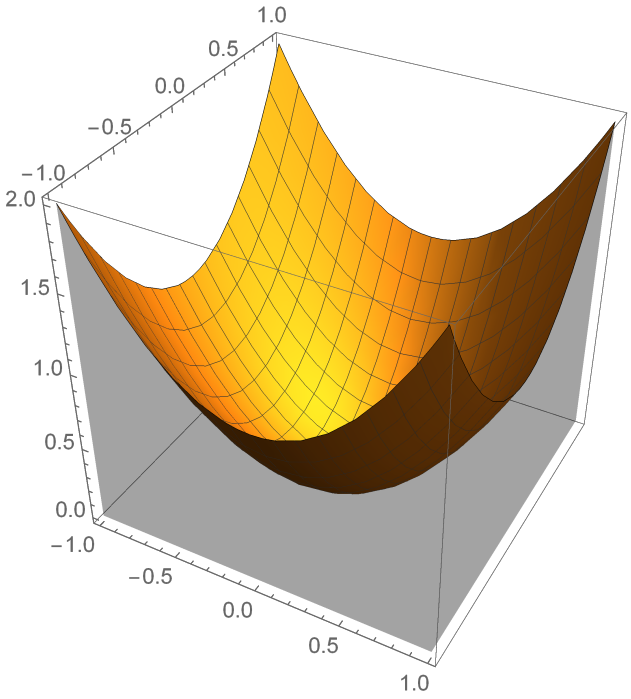
Como é visto, a proporção do gráfico foi aumentada ocasionando um estreitamento de sua base e aumento de sua altura na escala.

Caso quiséssemos testar fazer uma figura com o eixo z com metade da proporção, note a diferença

```
Plot3D[x^2+y^2,{x,-1,1},{y,-1,1},BoxRatios->{1,1,.5}]
```



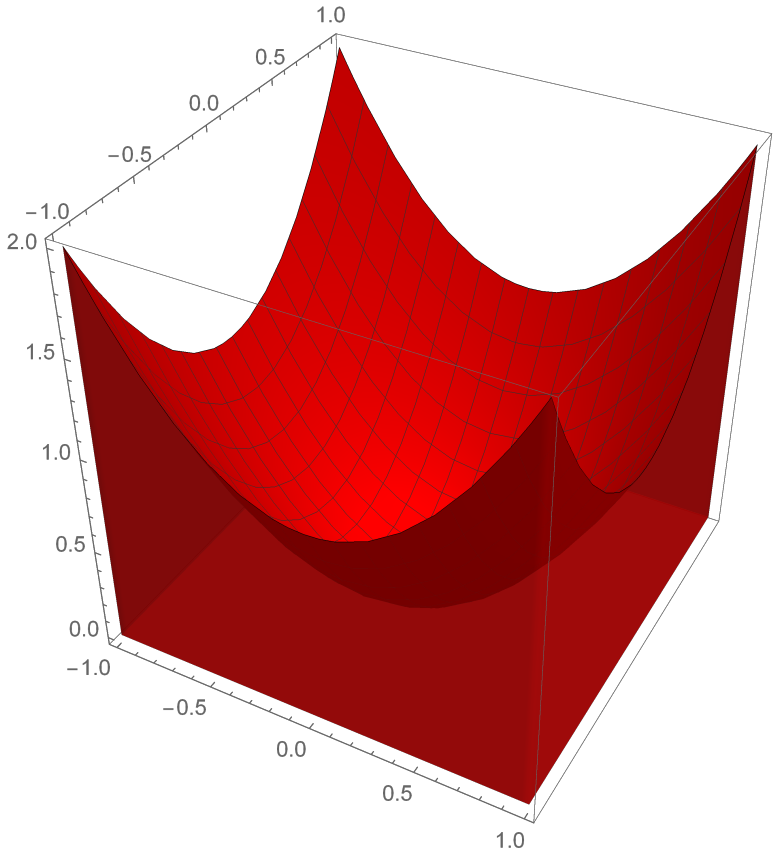
- `Filling`
  - Ferramenta bastante útil na estilização dos gráficos, nos permite criar um sombreado preenchendo a parte inferior das superfícies. Também é aplicável no `Plot`




```
Plot3D[x^2+y^2,{x,-1,1},{y,-1,1},Filling->Bottom]
```

∗ Aplicando o `FillingStyle`, até então não mencionado podemos obter

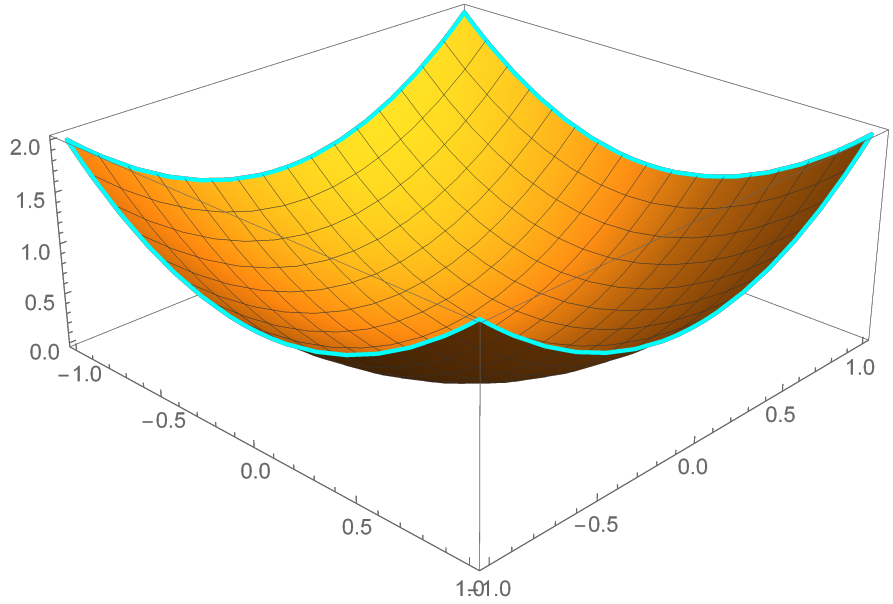
```
Plot3D[x^2+y^2,{x,-1,1},{y,-1,1},Filling->Bottom,
    FillingStyle->Opacity[.8],PlotStyle->Red]
```



- `BoundaryStyle`

  Cria um contorno ao longo das bordas da superfície.

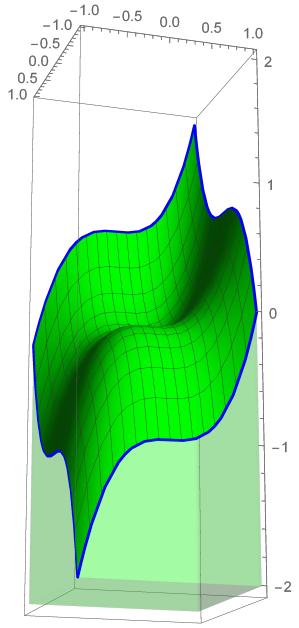
```
Plot3D[x^2+y^2,{x,-1,1},{y,-1,1},BoundaryStyle->{Thick,RGBColor
    [0,255,255]}]
```



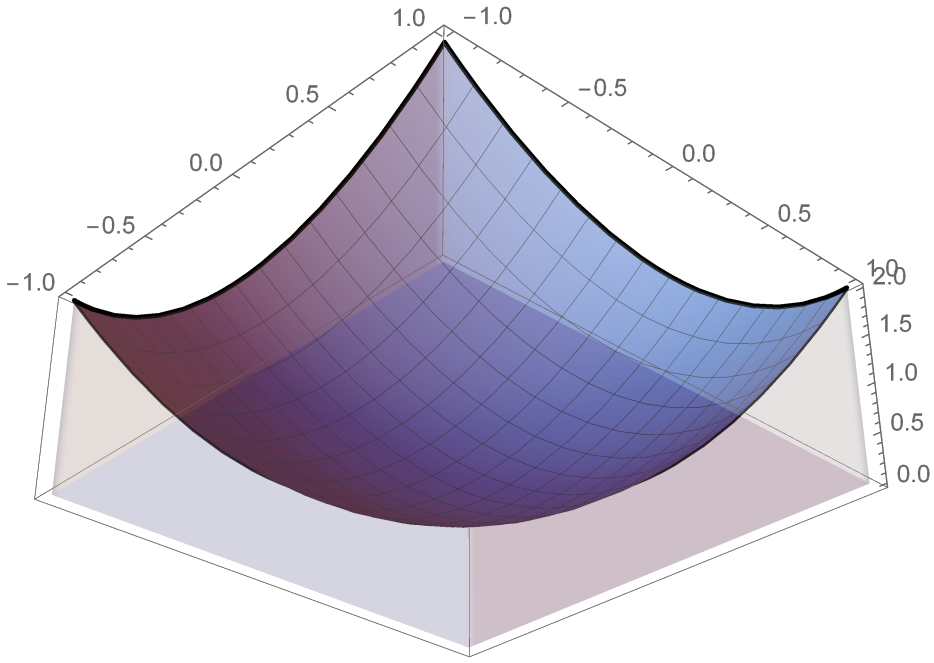
- `PlotStyle`

  Vamos resgatar algumas propriedades de estilização já conhecidas mostrando alguns exemplos de superfícies

```
Plot3D[x^3 + y^3, {x, -1, 1}, {y, -1, 1}, PlotStyle -> {Green},
BoundaryStyle -> {Thick, Blue}, Filling -> Bottom,
FillingStyle -> {Opacity[.2]}, BoxRatios -> {1, 1, 3}]
```



```
Plot3D[x^2 + y^2, {x, -1, 1}, {y, -1, 1}, PlotStyle -> {
   LightBlue},BoundaryStyle -> {Thick, Blue}, Filling ->
   Bottom,FillingStyle -> {Opacity[.2]}]
```

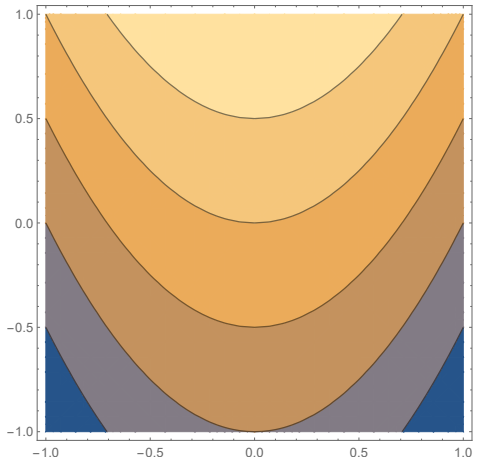


3 ContourPlot

Esse comando nos permite plotar os contornos de funções a depender de como passamos os parâmetros. Podemos imaginar que caso passarmos uma expressão seguindo a mesma estrutura do `Plot` vamos ter a replicação da função f ao longo do eixo y em diferentes alturas. Vamos exemplificar para tornar mais fácil.

Imagine que desejamos novamente plotar a função do segundo grau $f(x) = x^2$, só que dessa vez, ao invés de plotar somente uma curva queremos uma família de funções que seguem o mesmo molde porém transladadas em y. Dessa forma, passaríamos o comando como é visto abaixo

```
ContourPlot[y - x^2, {x, -1, 1}, {y, -1, 1}]
```



Afinal, o que foi feito para que chegássemos nessa conformação de curvas? A resposta é simples. Simplesmente assumimos que $f(x) = x^2$, trocamos $f(x)$ por y e subtraímos x^2 de ambos os lados, logo

$$y - x^2 = 0 \tag{2}$$

Isso significa que temos um contorno da função f cujo rótulo corresponde a 0. Ou seja, o valor nulo encontrado demonstra que a função f ao longo de toda a curva $y - x^2$ está rotulado como 0 em três dimensões (Altura constante ao longo dessa parábola). Nesse caso, como estamos tratando de rótulos um atributo interessante de ser usado é o `ContourLabels`. Ele vai mapear a altura de cada parábola na representação em duas dimensões. Vejamos

```
1 ContourPlot[y - x^2, {x, -1, 1}, {y, -1, 1}]
```