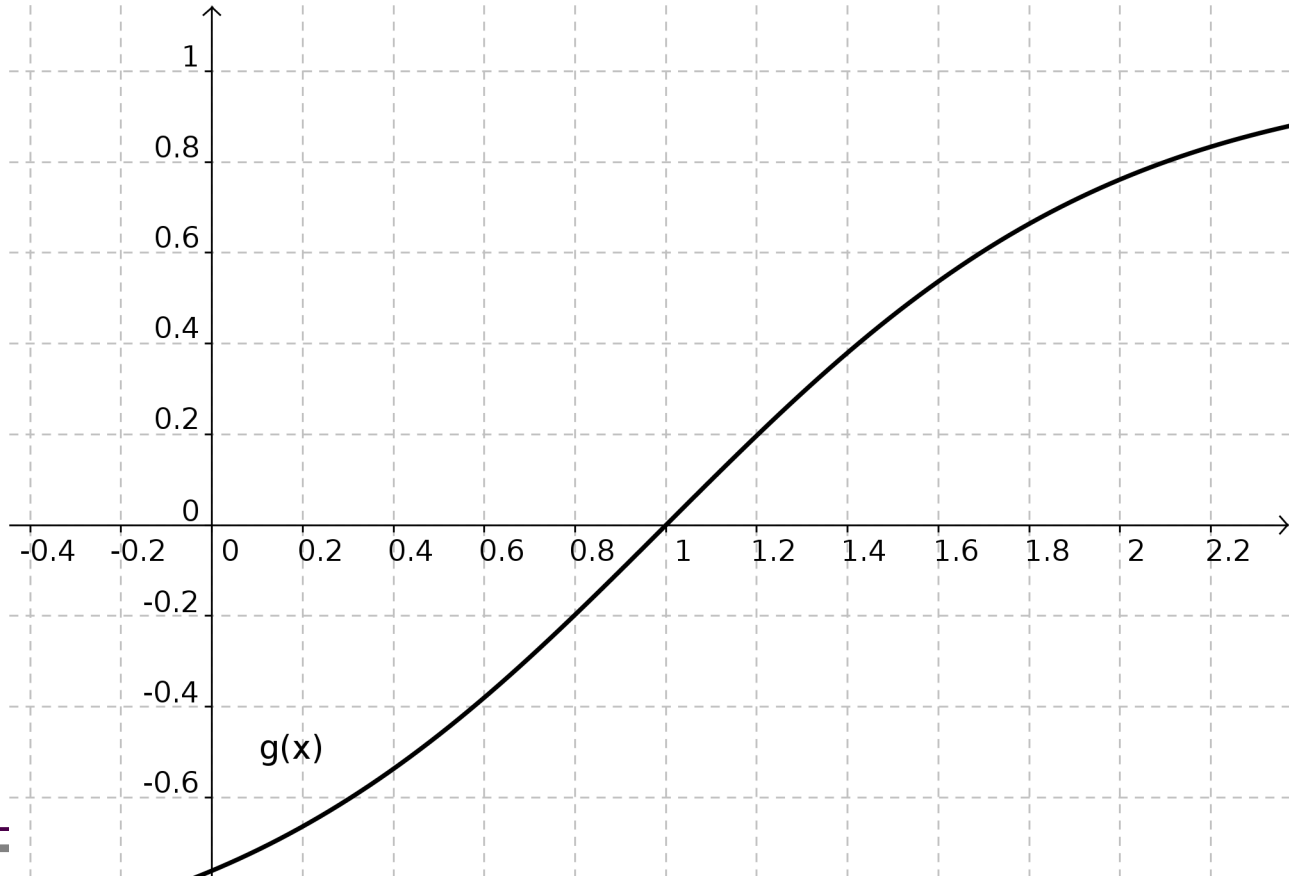
1. rootfinding

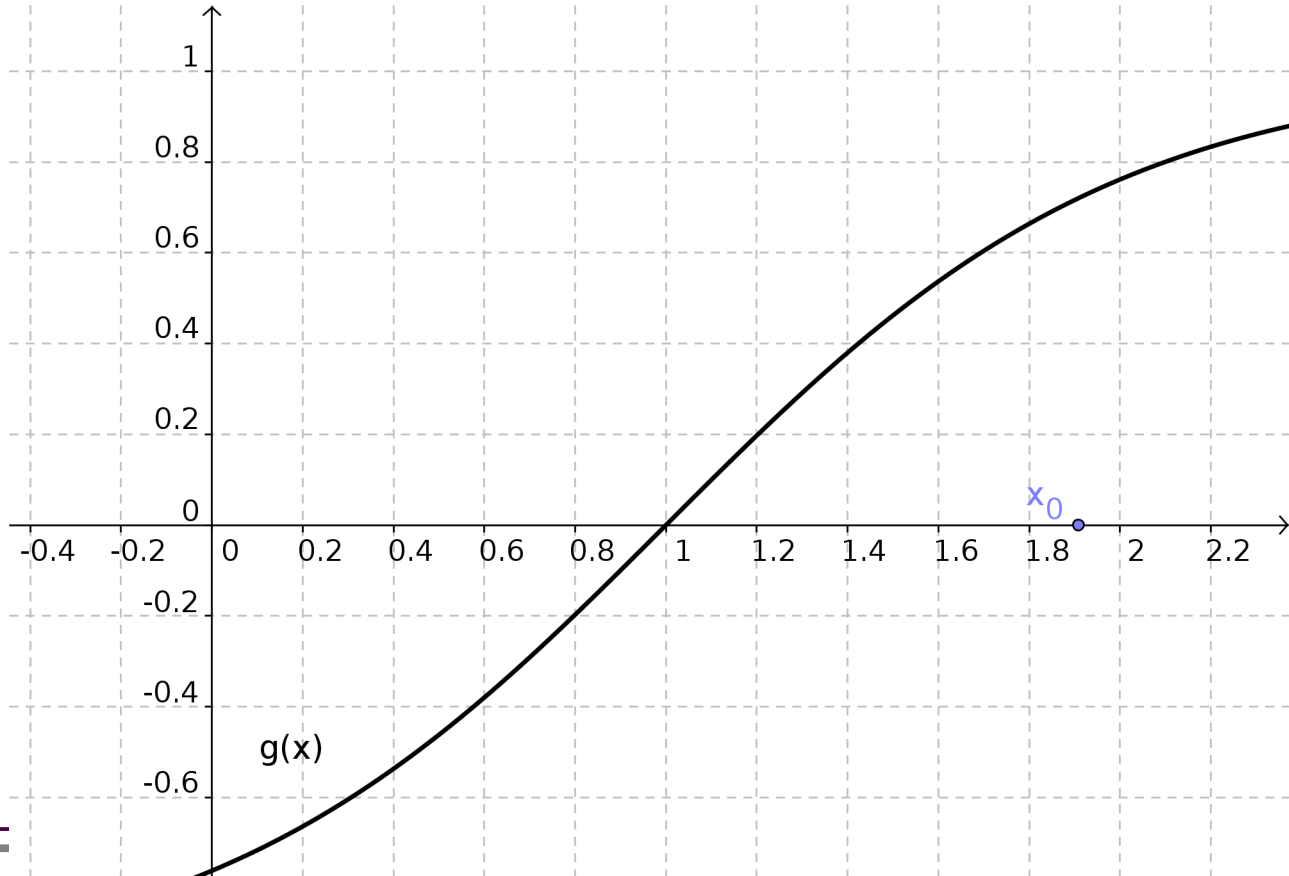
Newton's method (1D)

- Find x such that $g(x)=0$
- $x \in \mathbb{R}, g: \mathbb{R} \rightarrow \mathbb{R}$



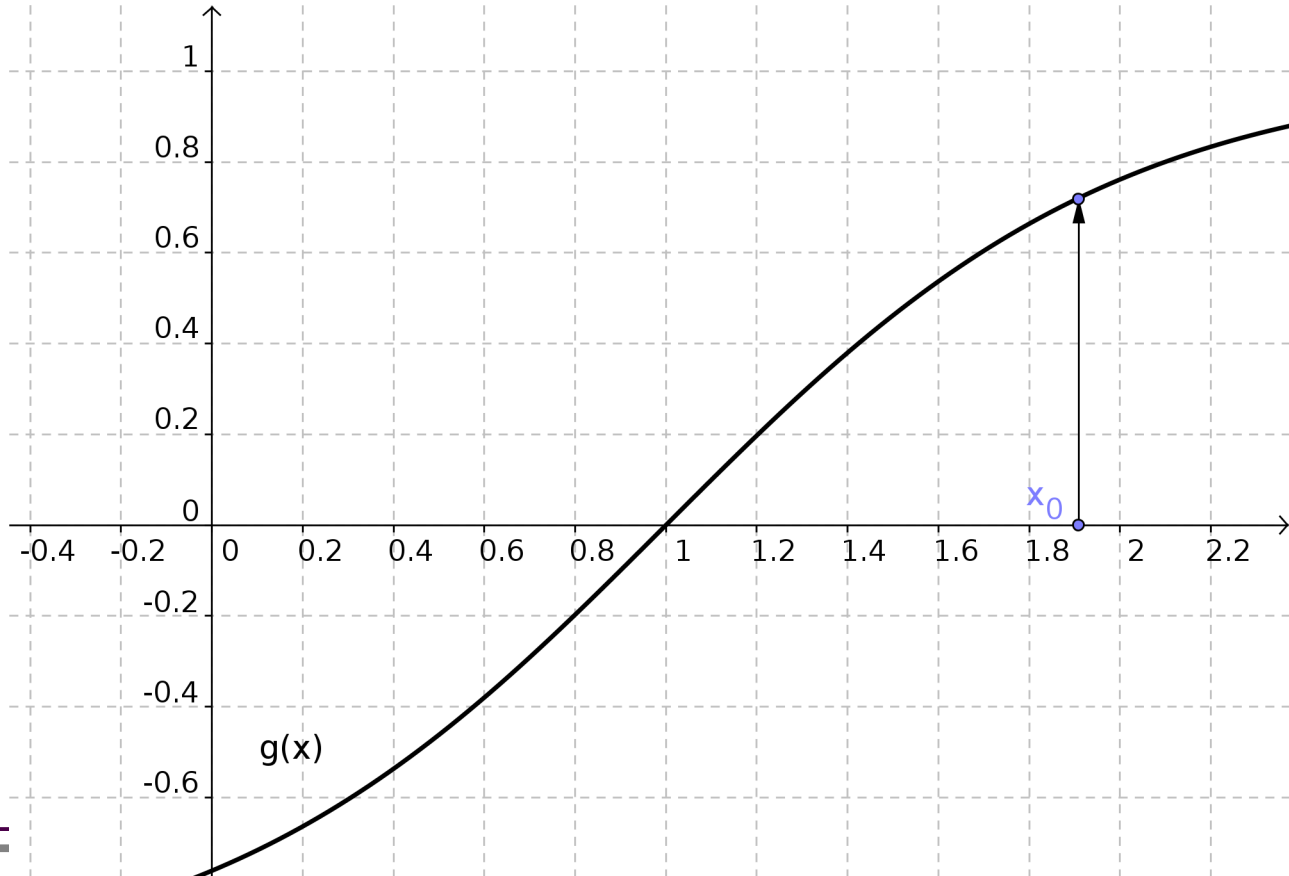
Newton’s method (1D)

- Iterative process: start with a guess for x



Newton's method (1D)

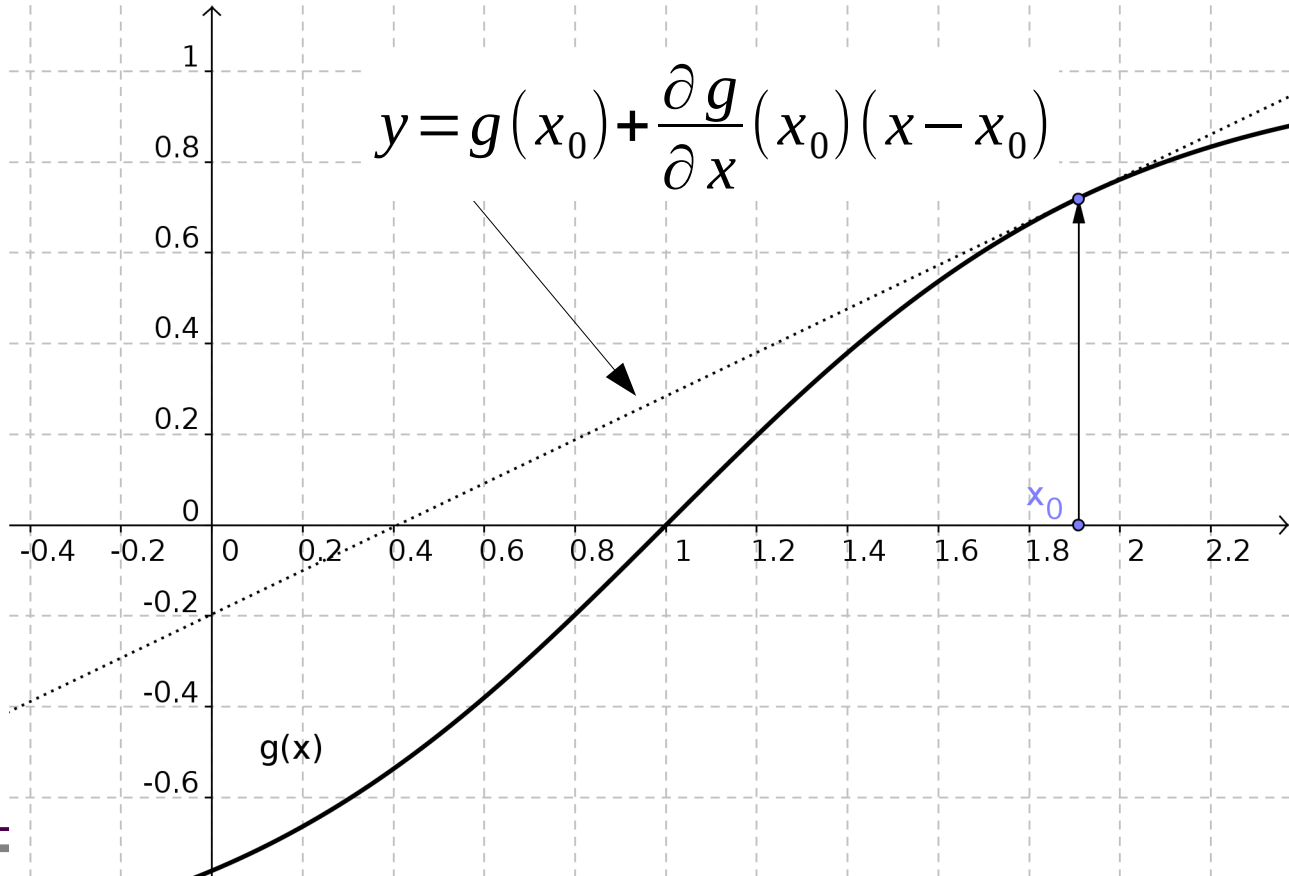
- Evaluate $g(x_0)$



Newton's method (1D)

- Compute tangent approximation

$$g(x) \approx g(x_0) + \frac{\partial g}{\partial x}(x_0)(x - x_0)$$



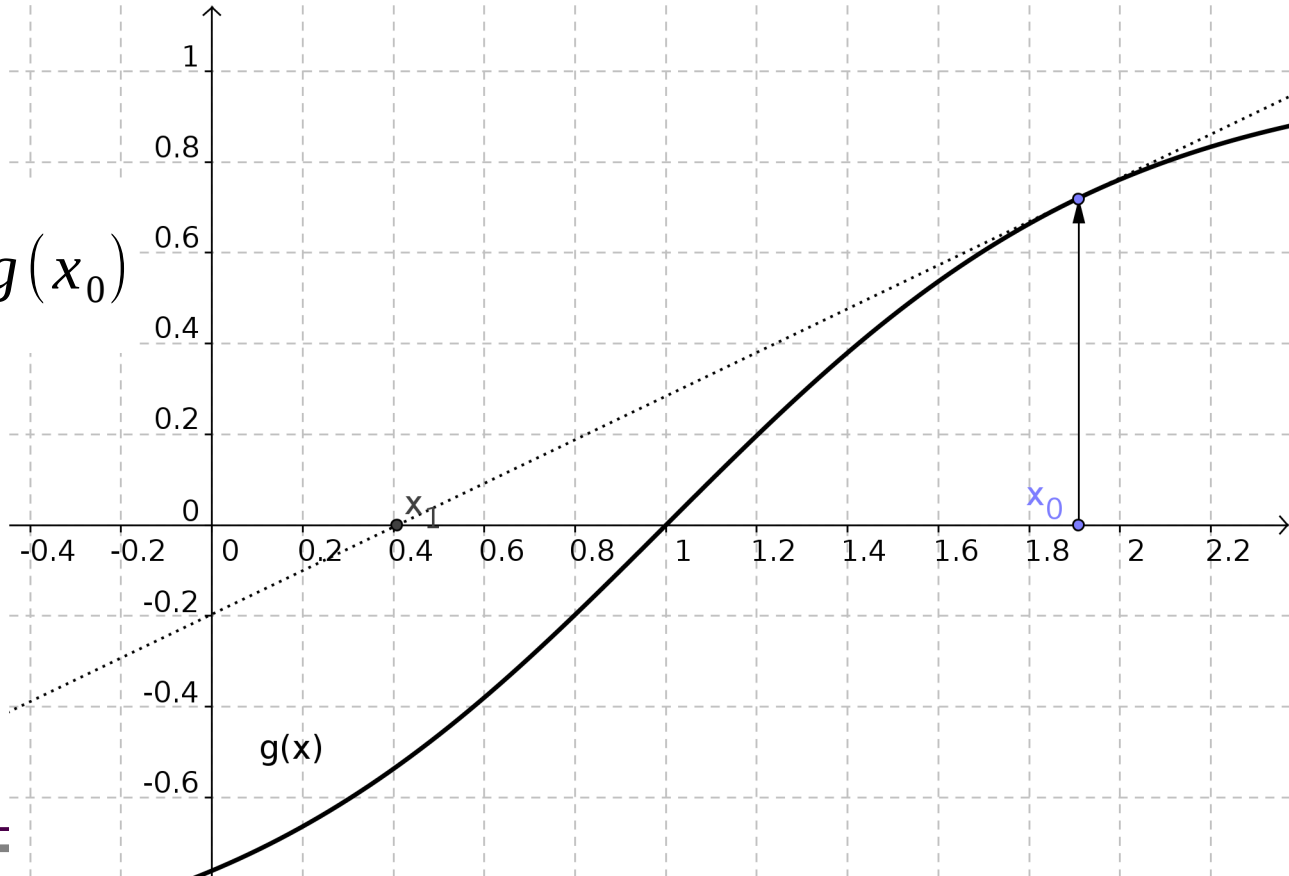
Newton's method (1D)

- Find x such that $g(x)=0$

$$g(x_0)+\frac{\partial g}{\partial x}(x_0)(x-x_0)=0$$

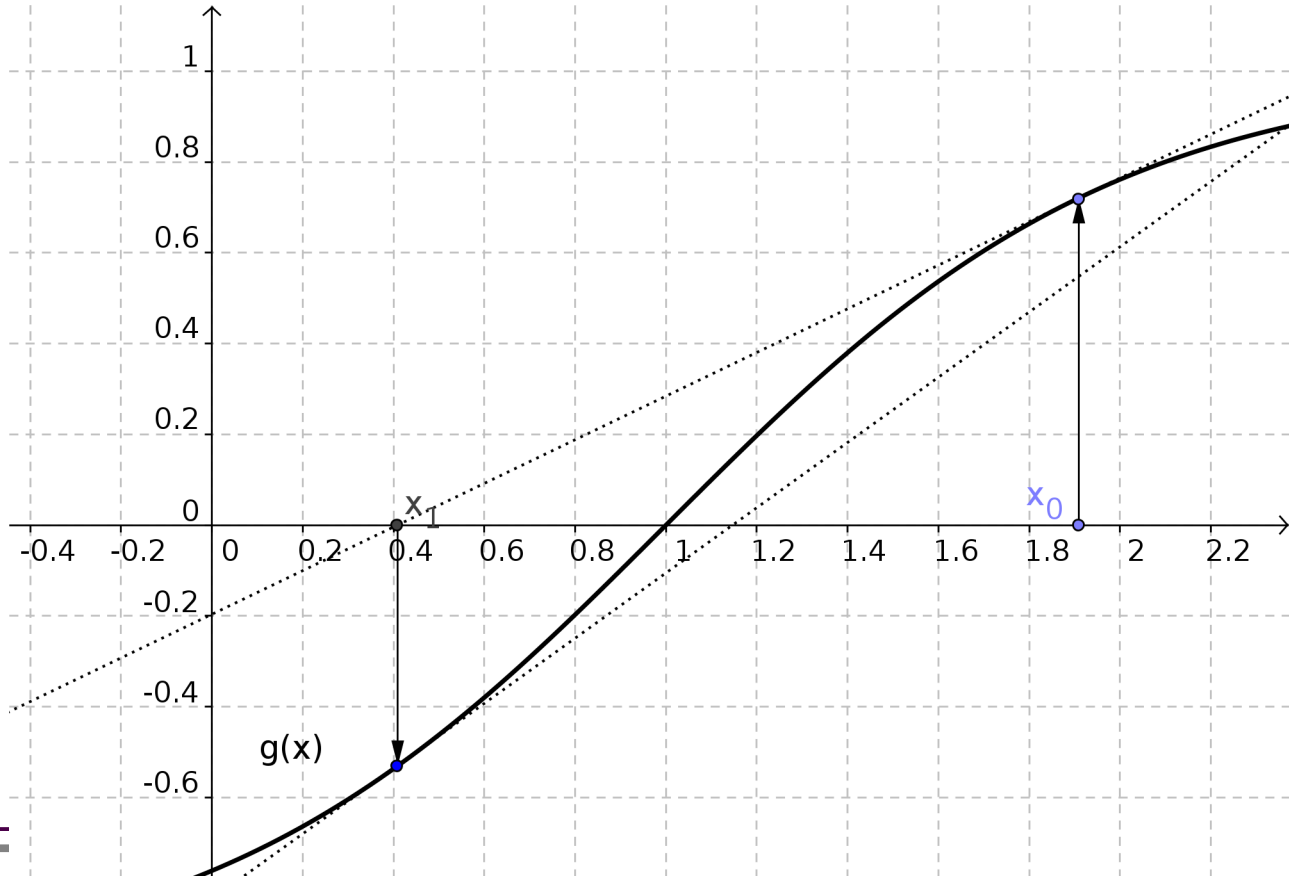
$$(x-x_0)=-\left(\frac{\partial g}{\partial x}(x_0)\right)^{-1} g(x_0)$$

$$\Delta x$$



Newton’s method (1D)

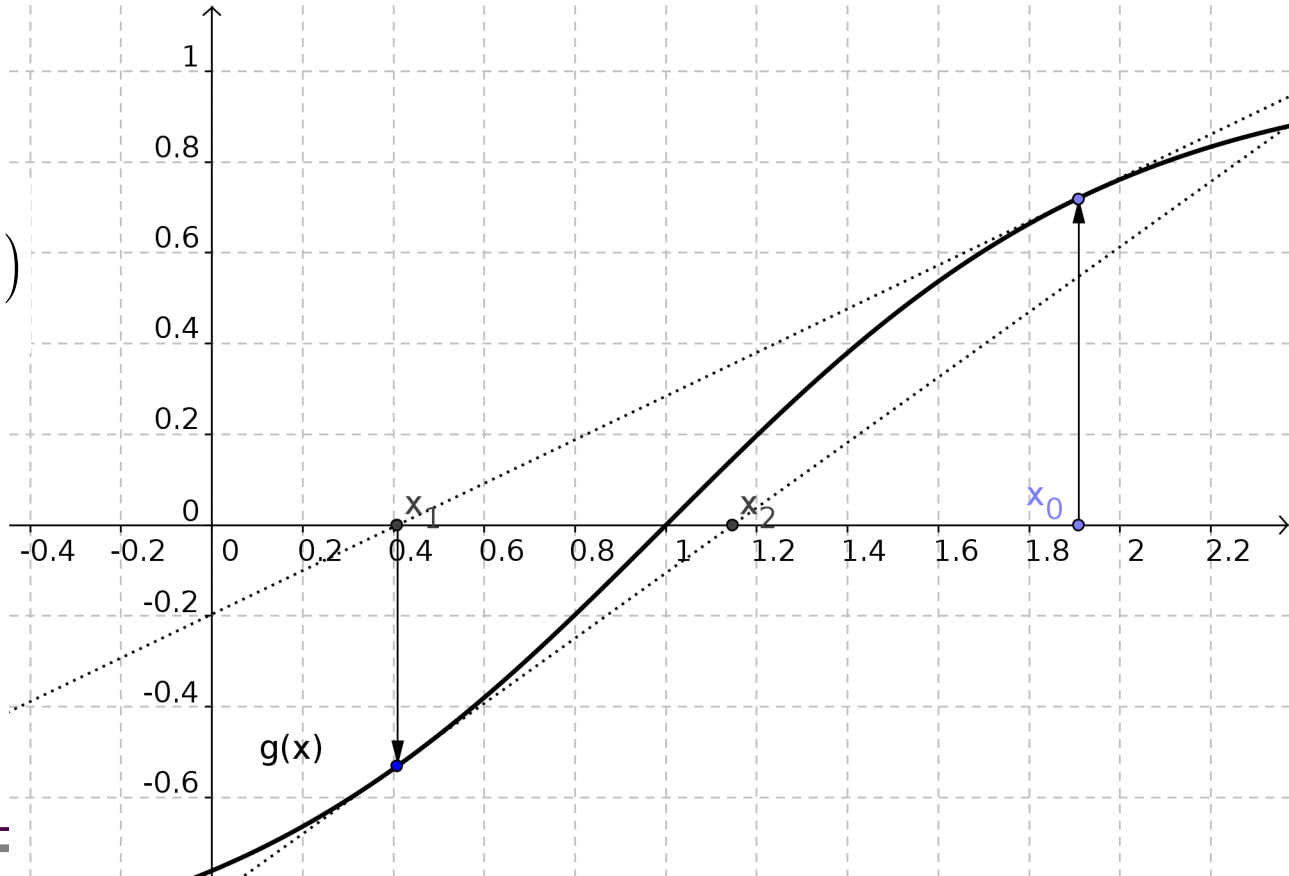
- Start again with new guess



Newton's method (1D)

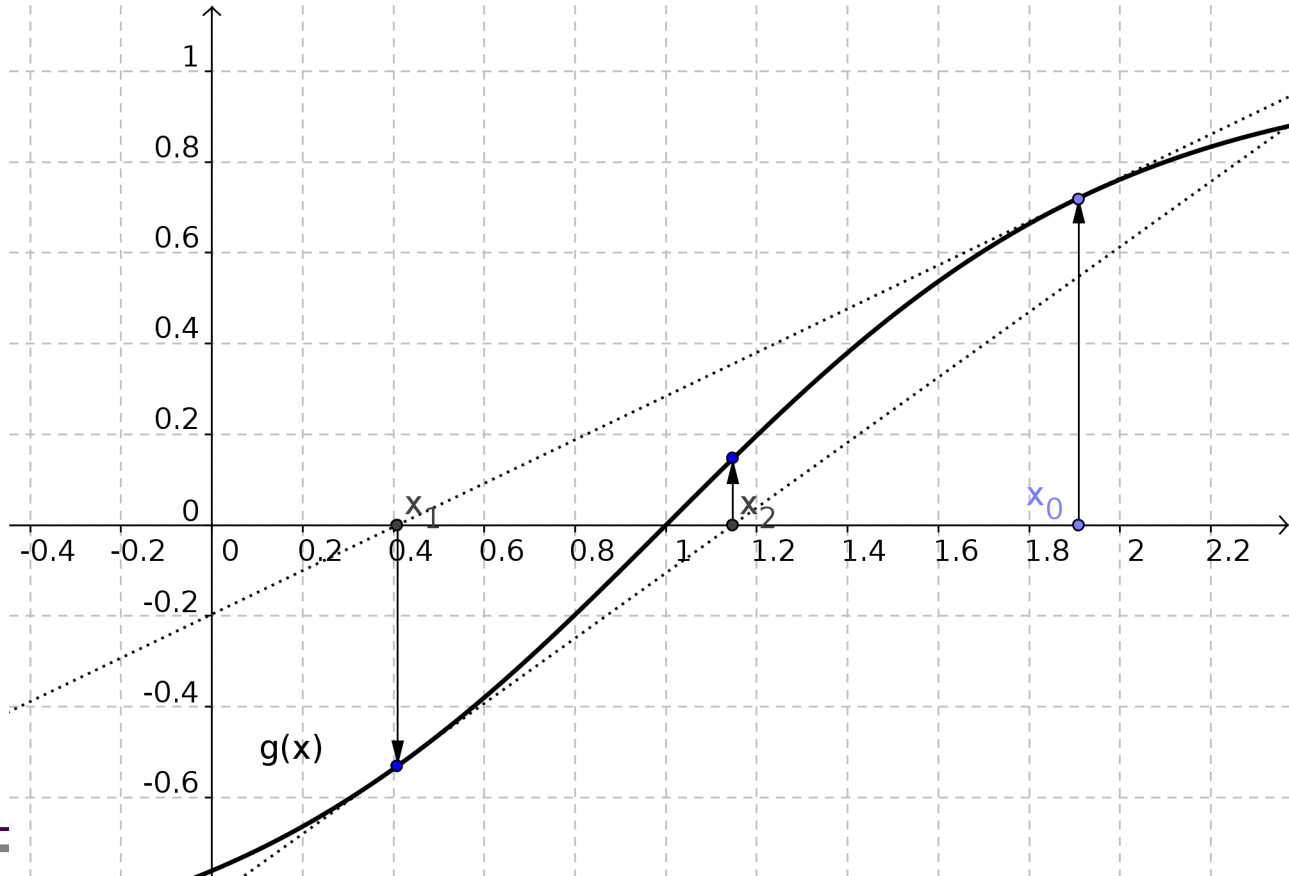
- Take a newton step

$$\Delta x = -\left(\frac{\partial g}{\partial x}(x_k)\right)^{-1} g(x_k)$$



Newton's method (1D)

- Repeat until $\|g(x)\| \leq tol$ $\|\Delta x\| \leq tol_{step}$

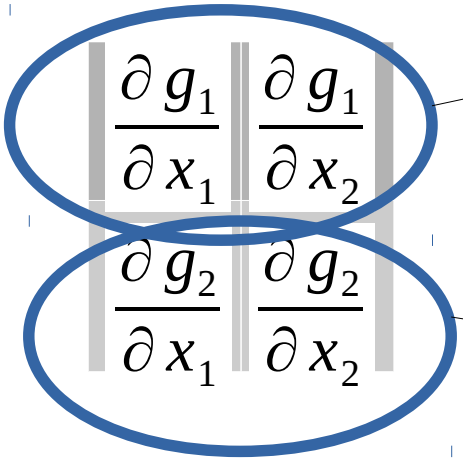


Newton's method

- This trick generalizes to more dimensions
- Find x such that $g(x)=0$
- $x \in \mathbb{R}^2, g: \mathbb{R}^2 \rightarrow \mathbb{R}^2$

$$g(x^0)+\frac{\partial g}{\partial x}(x^0)(x-x^0)=0$$

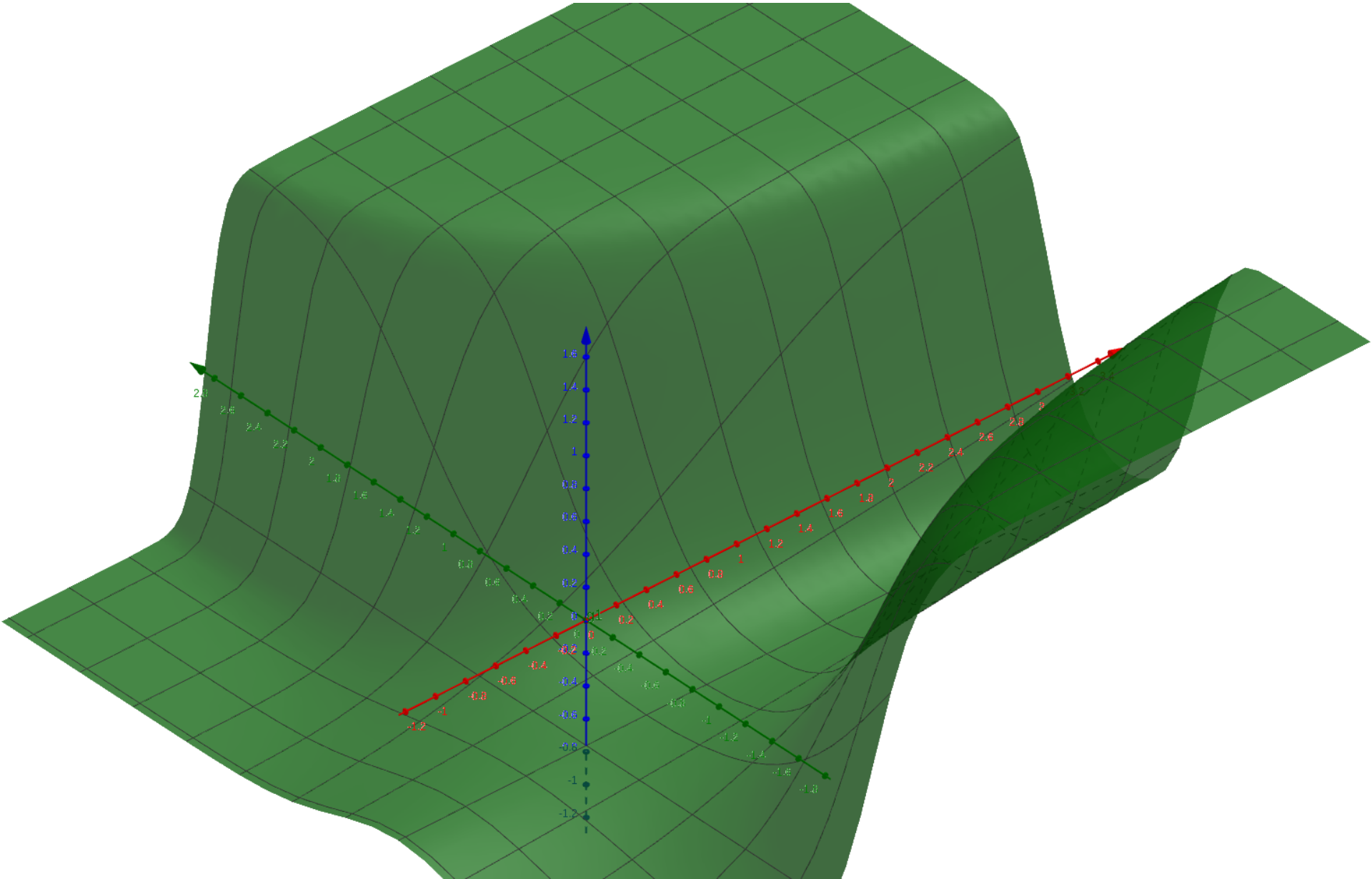
Jacobian $\frac{\partial g}{\partial x}$:



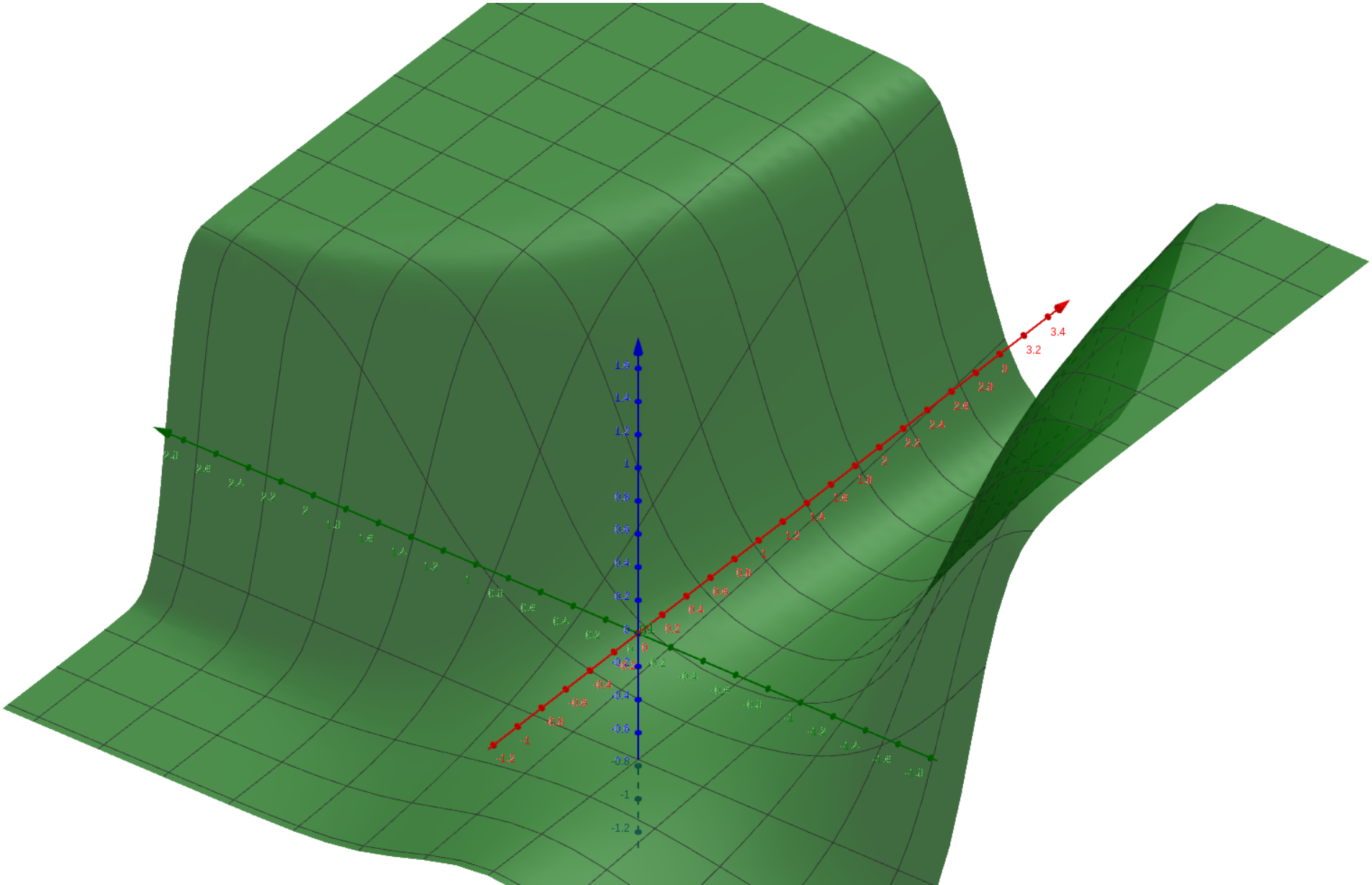
Transpose(gradient $\nabla_x g_1$)

Transpose(gradient $\nabla_x g_2$)

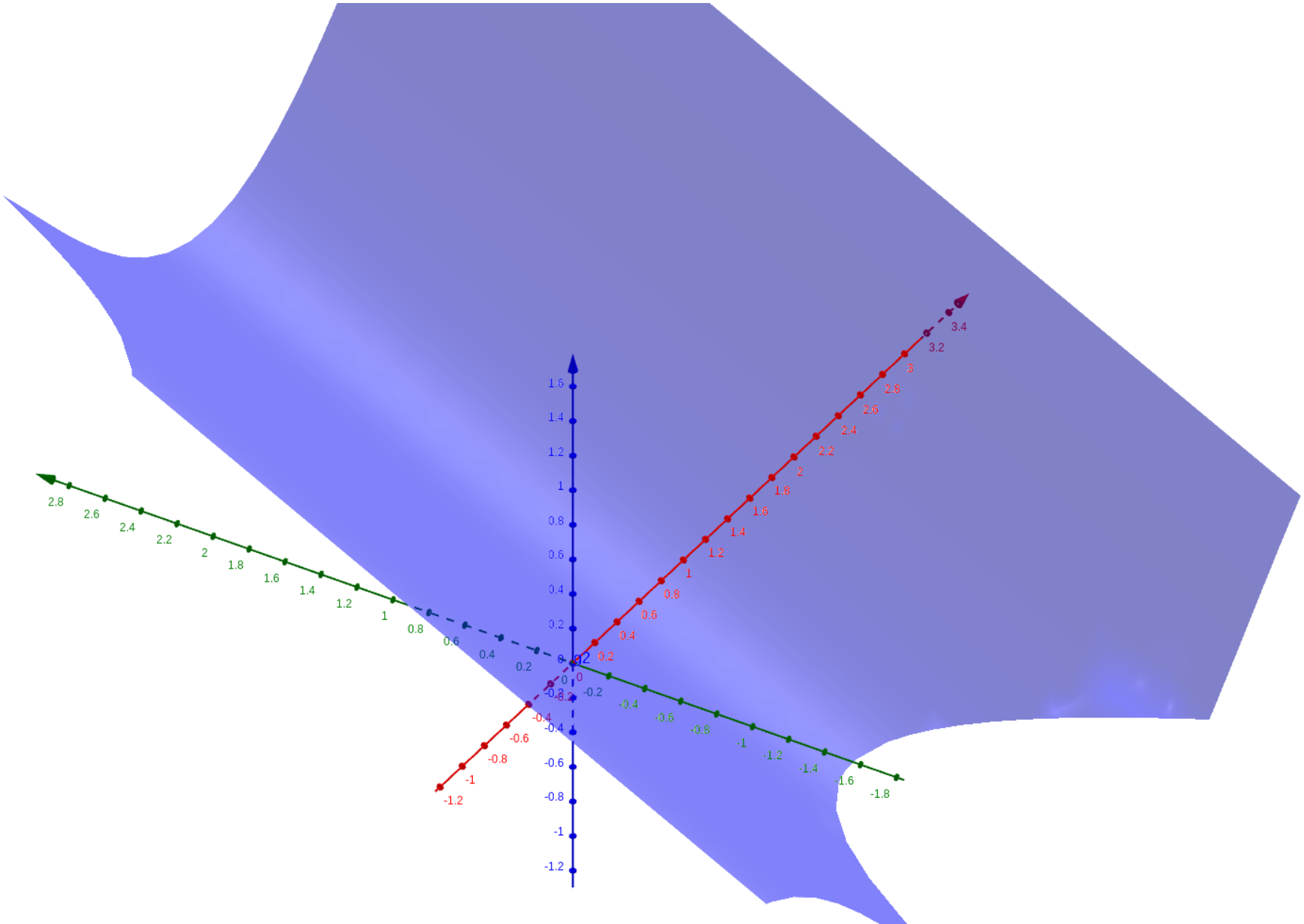
$g_1(x, y)$



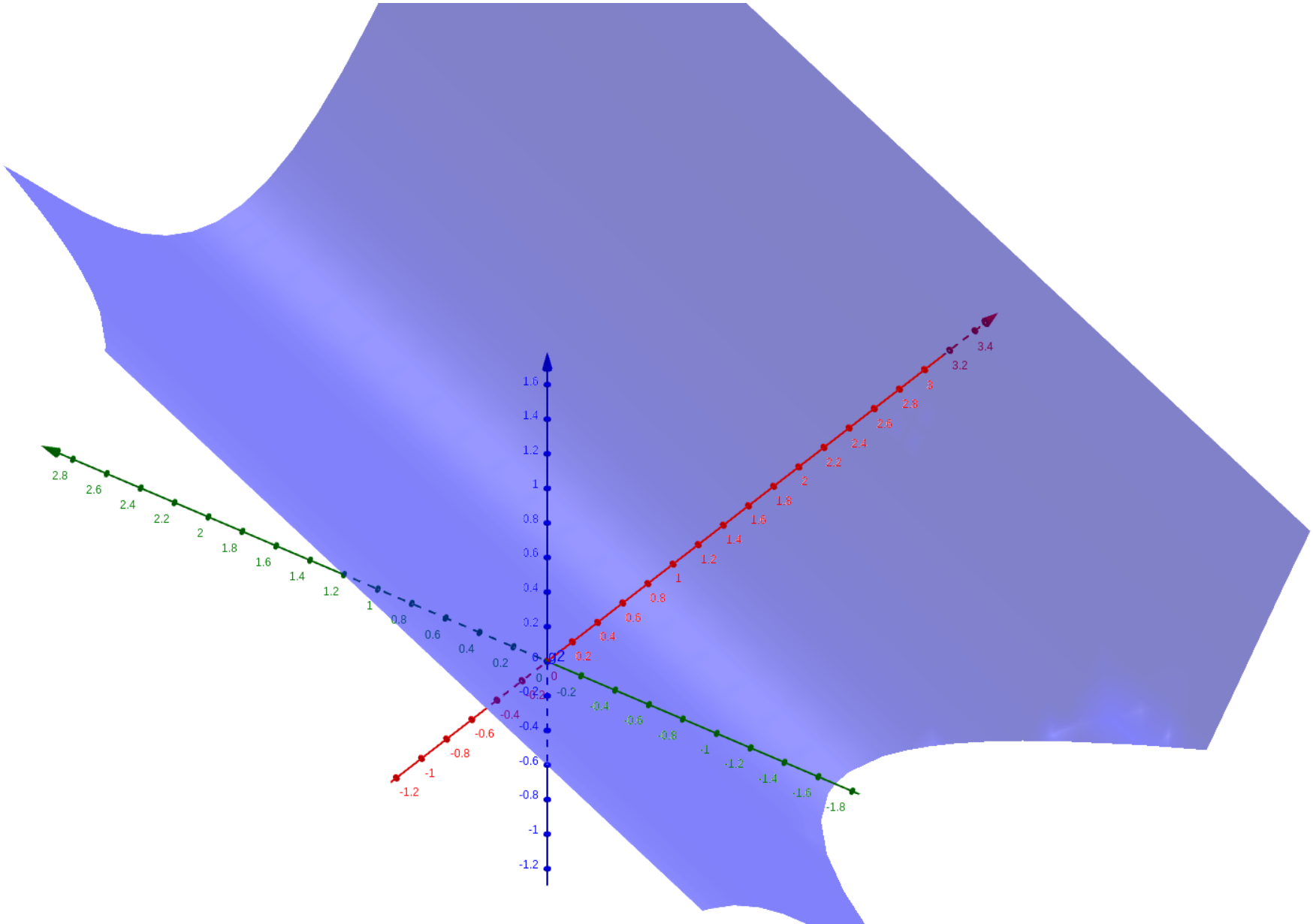
$g_1(x, y)$



$g_2(x, y)$


2.8
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3
3.2
3.4

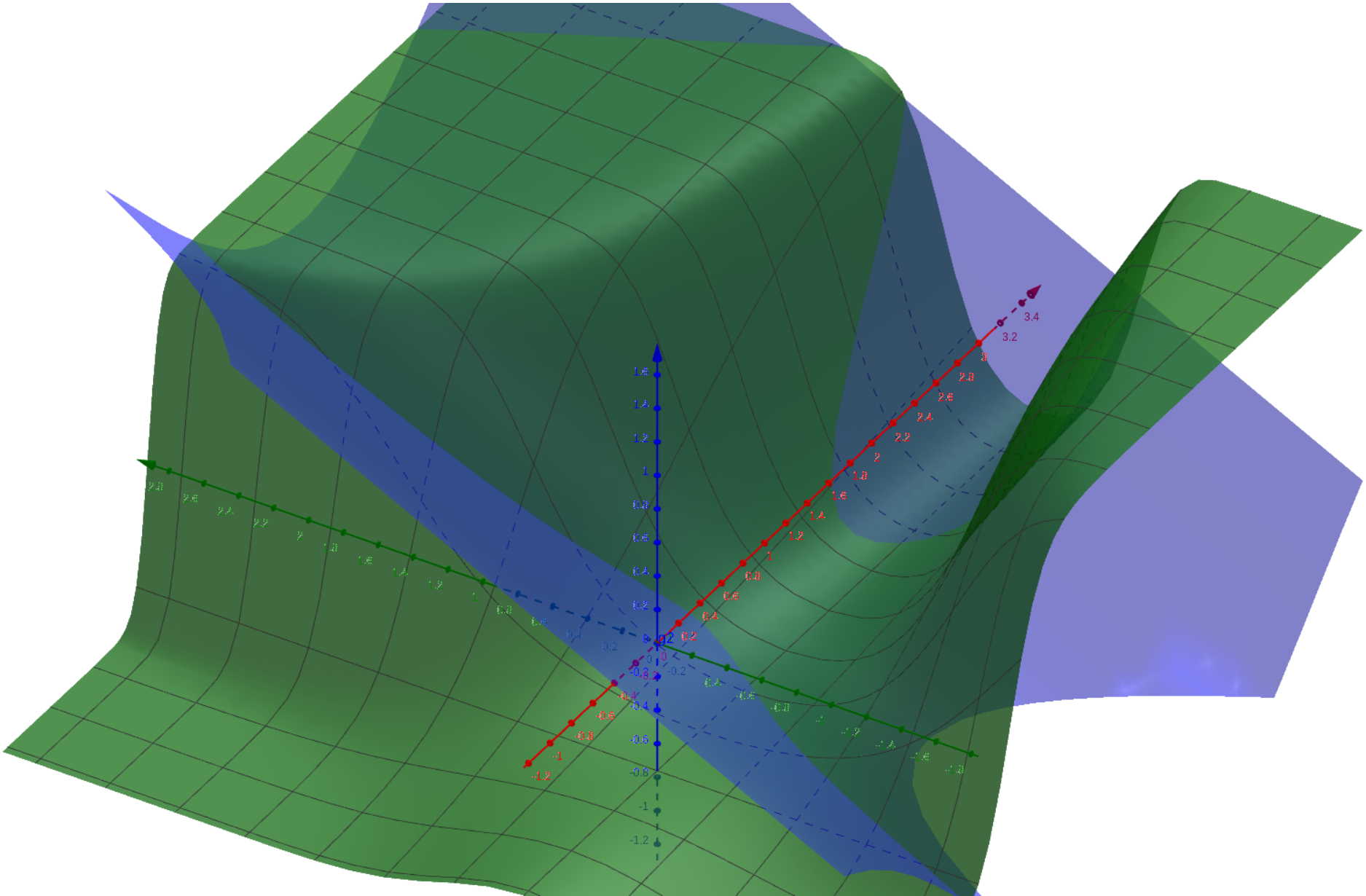
$g_2(x, y)$


2.8
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3
3.2
3.4

$g_1(x,y)$
$g_2(x,y)$

$g(x,y)$

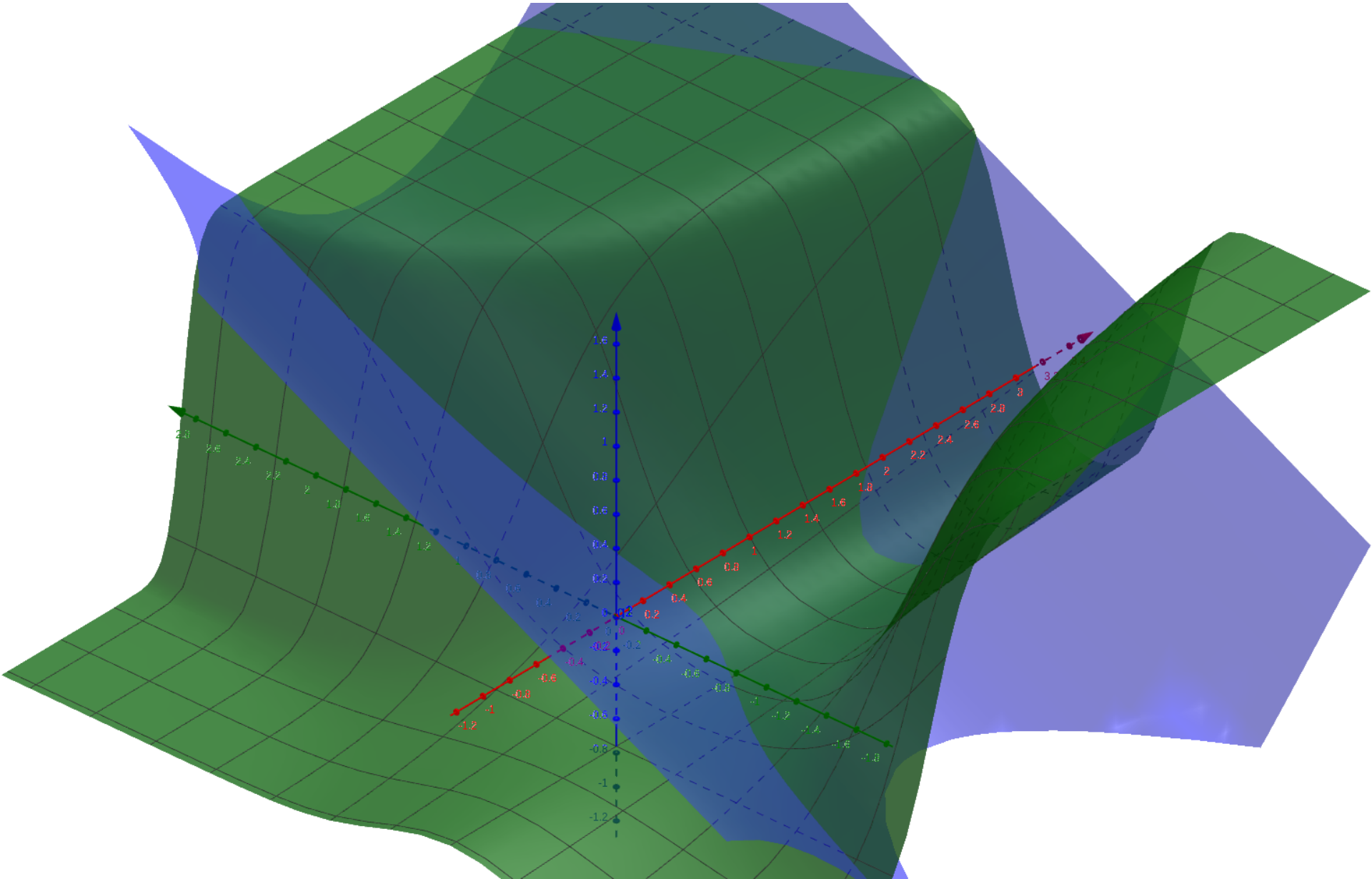
$g:\mathbb{R}^2 \to \mathbb{R}^2$



$g_1(x,y)$
$g_2(x,y)$

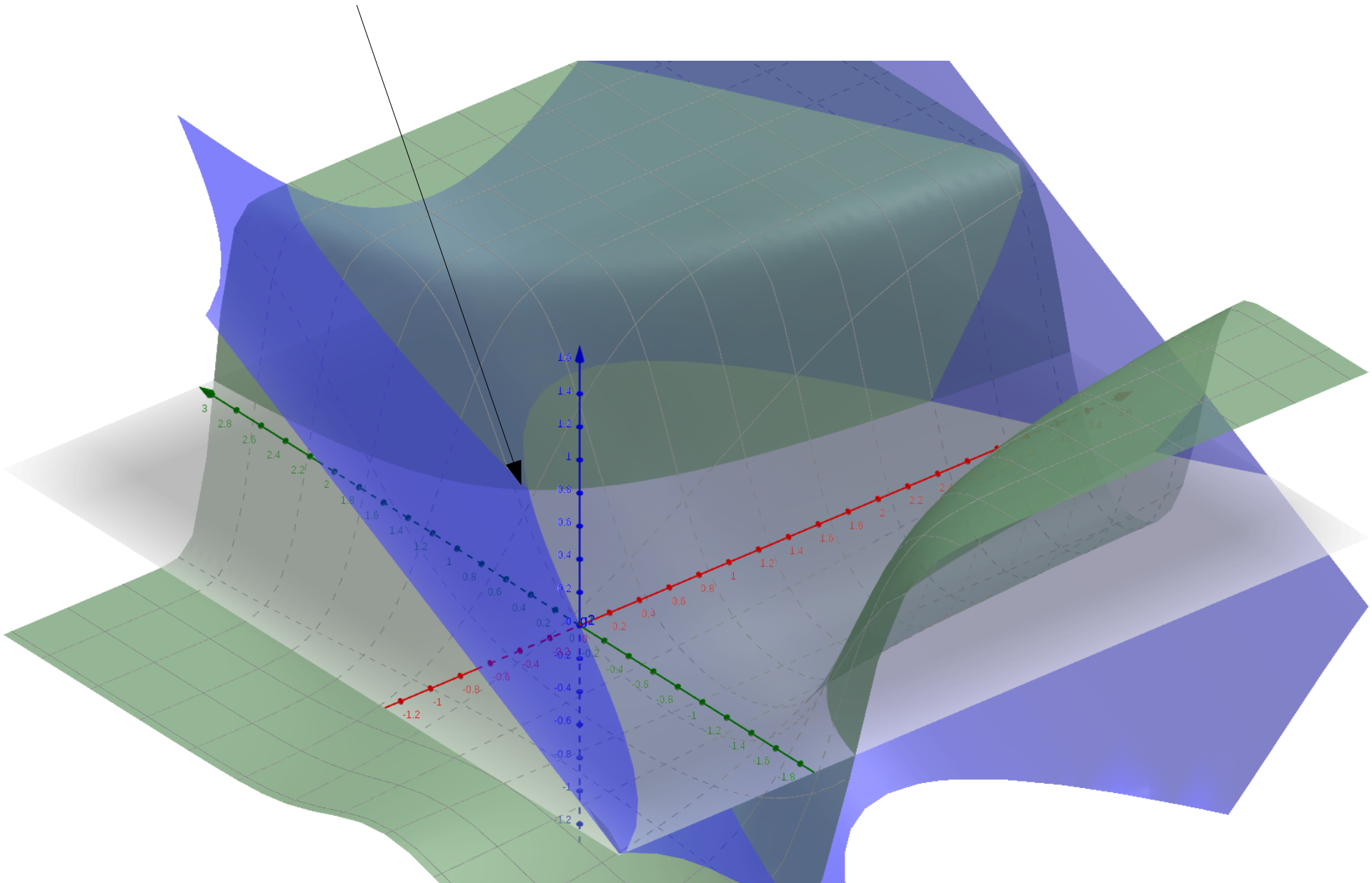
$g(x,y)$

$g:\mathbb{R}^2 \to \mathbb{R}^2$

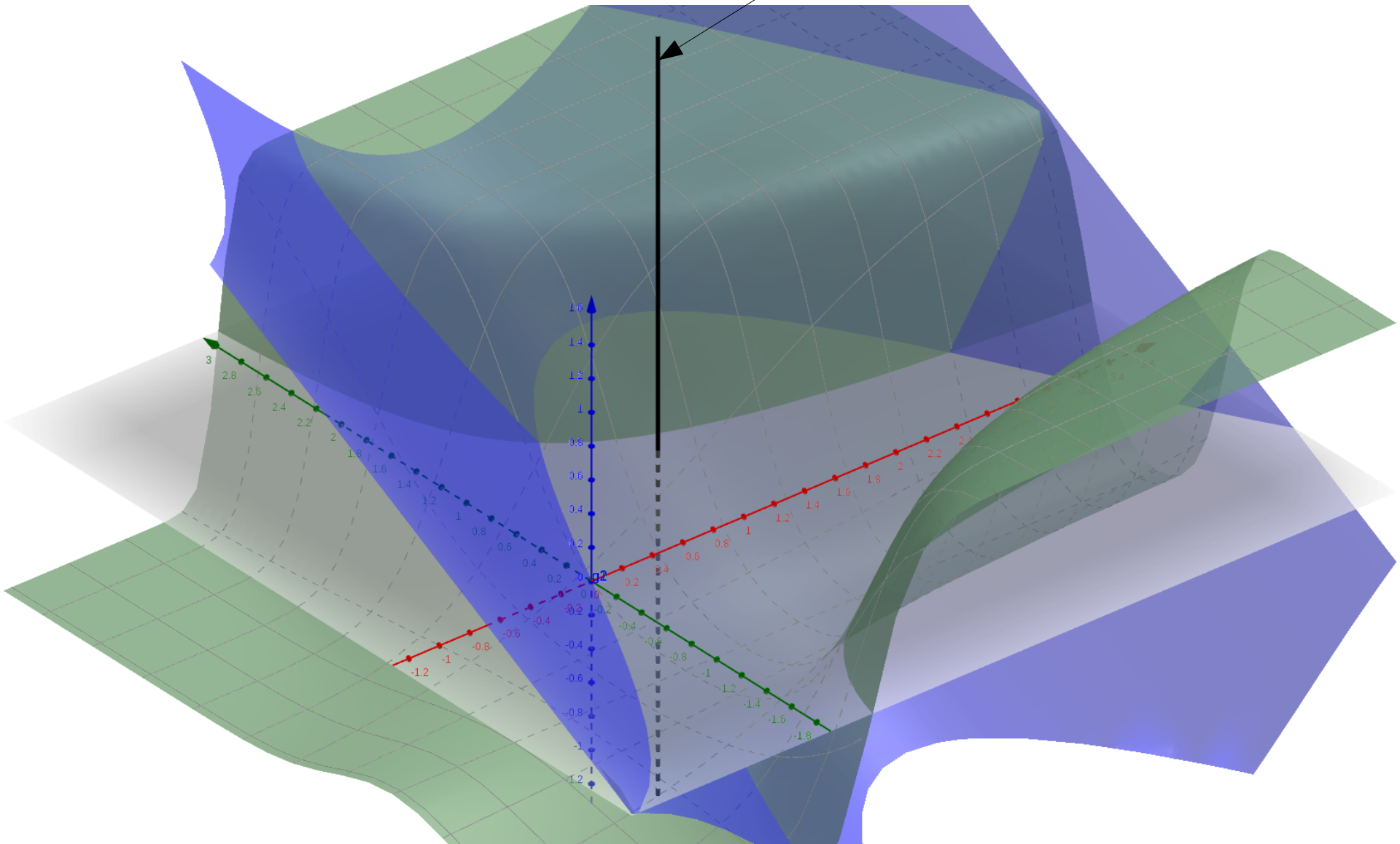


Finding this point

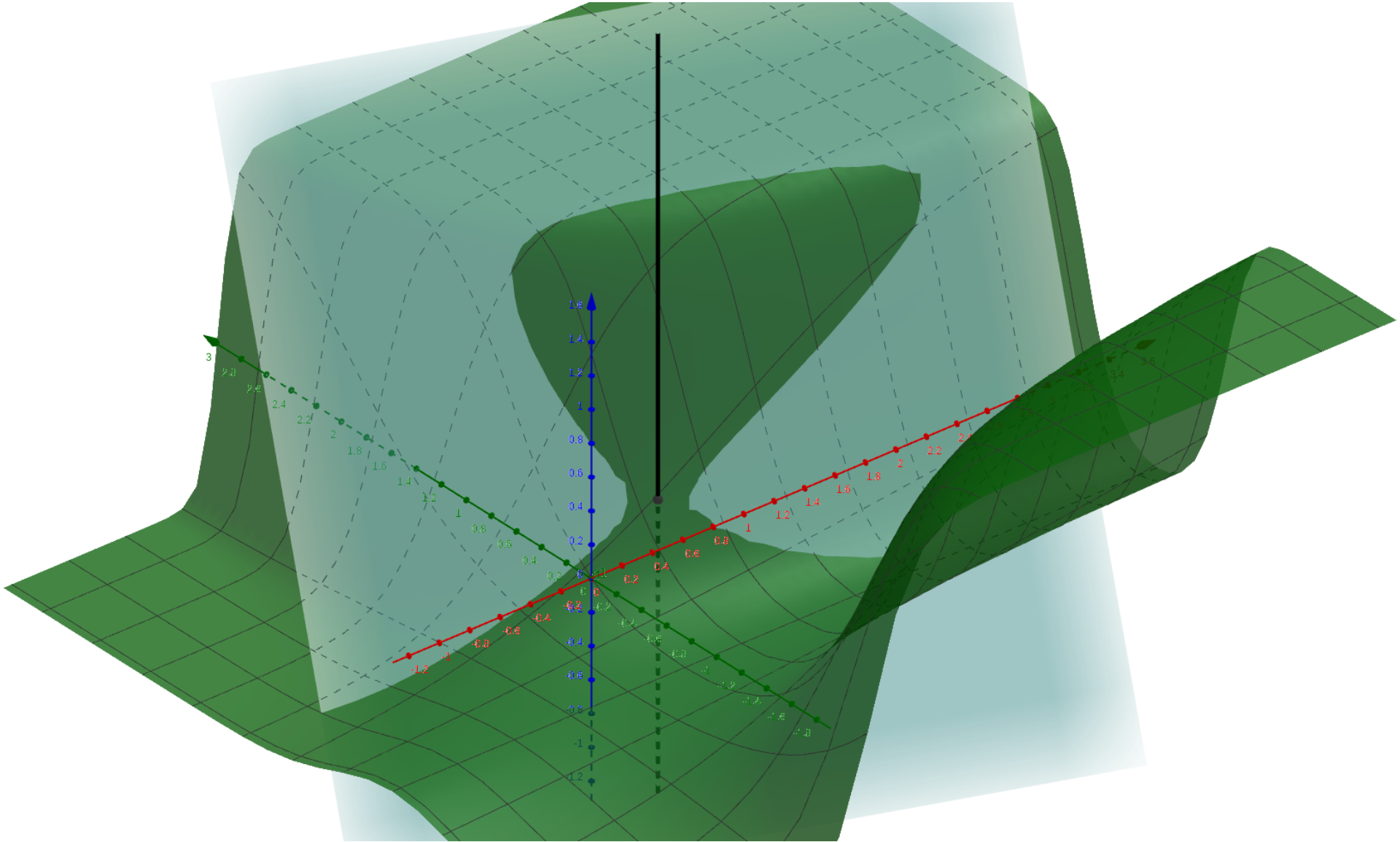
$$g(x, y)=0$$



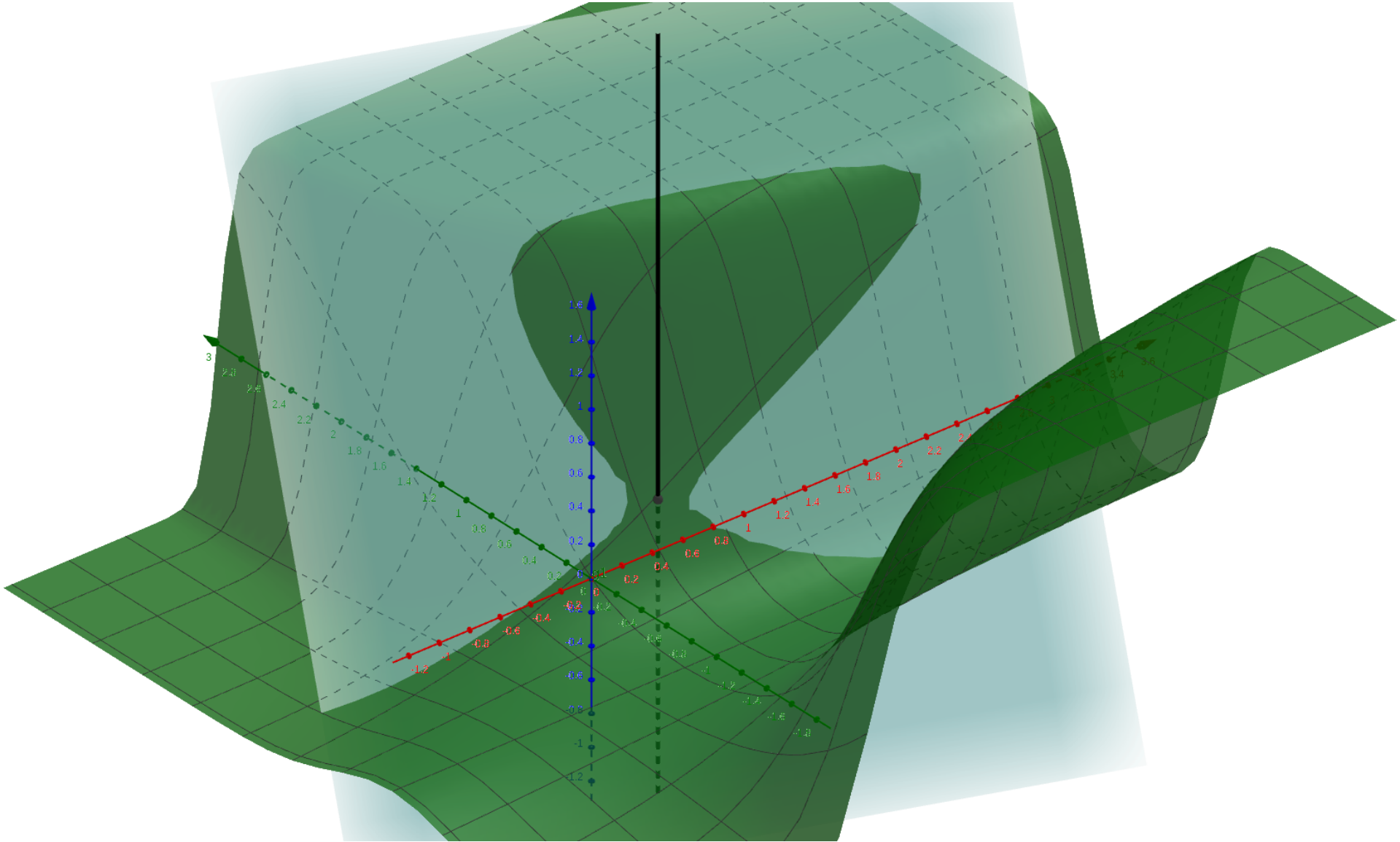
Start with (x_0, y_0)



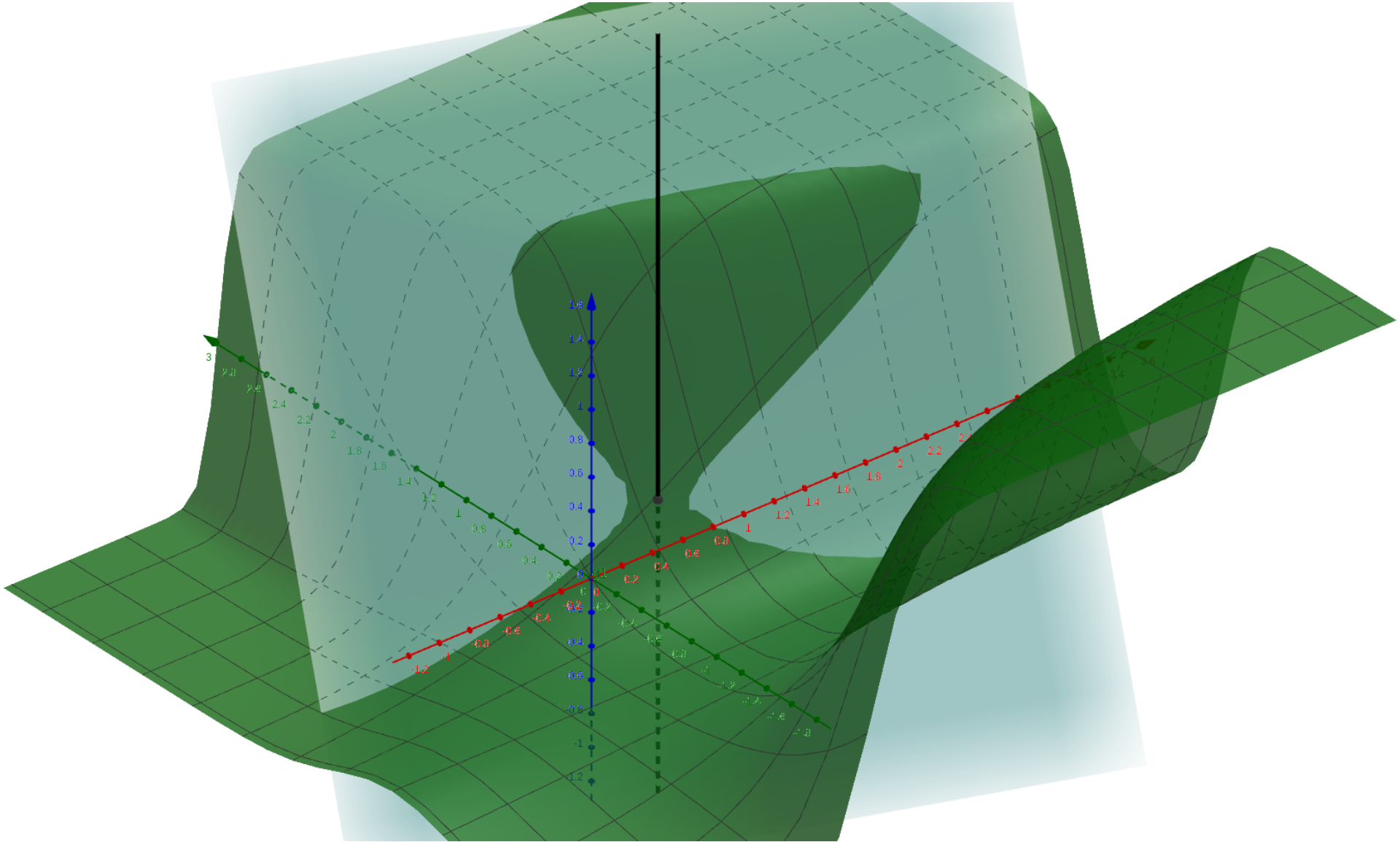
$$g_1(x,y) \approx g_1(x_0,y_0) + \frac{\partial g_1}{\partial x}(x_0,y_0)(x-x_0) + \frac{\partial g_1}{\partial y}(x_0,y_0)(y-y_0)$$



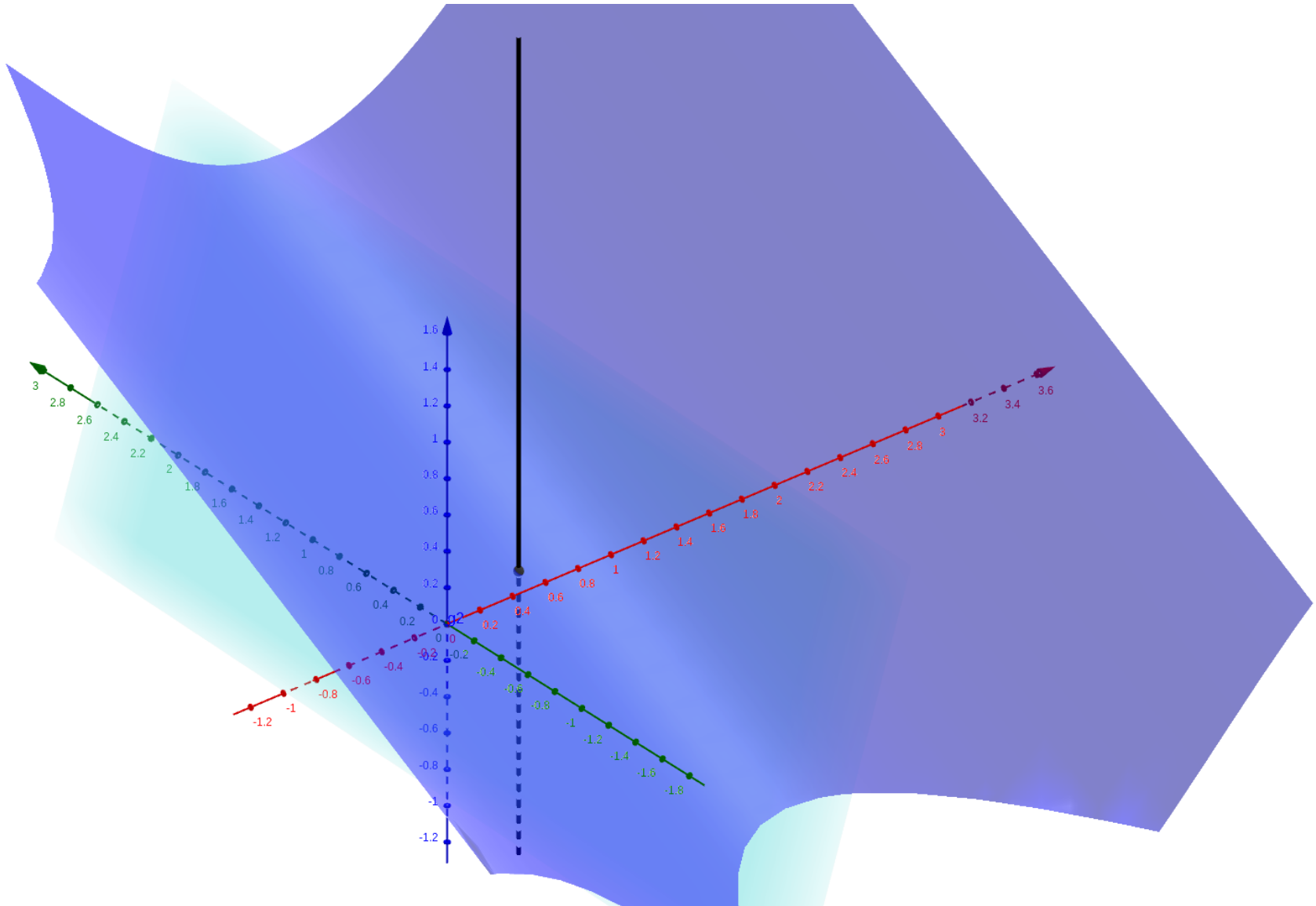
$$g_1(x,y) \approx \bar{g}_1 + \frac{\partial g_1}{\partial x}(x - x_0) + \frac{\partial g_1}{\partial y}(y - y_0)$$



$$g_1(x,y) \approx \bar{g}_1 + \left[\frac{\partial g_1}{\partial x} \quad \frac{\partial g_1}{\partial y}\right] \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix}$$

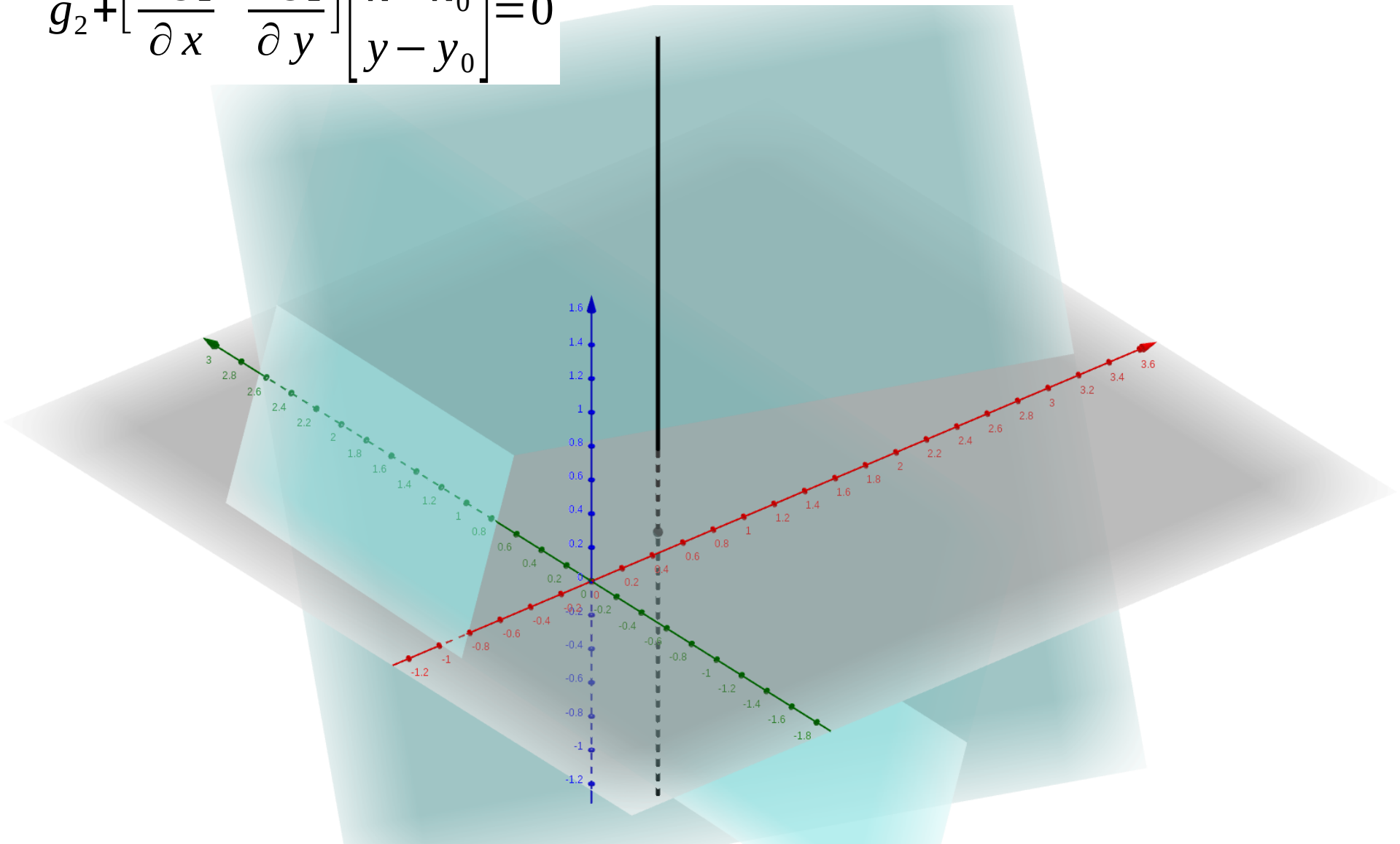


$$g_2(x,y) \approx \bar{g}_2 + \left[\frac{\partial g_2}{\partial x} \quad \frac{\partial g_2}{\partial y}\right] \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix}$$



$$\bar{g}_1 + \begin{bmatrix} \frac{\partial g_1}{\partial x} & \frac{\partial g_1}{\partial y} \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix} = 0$$

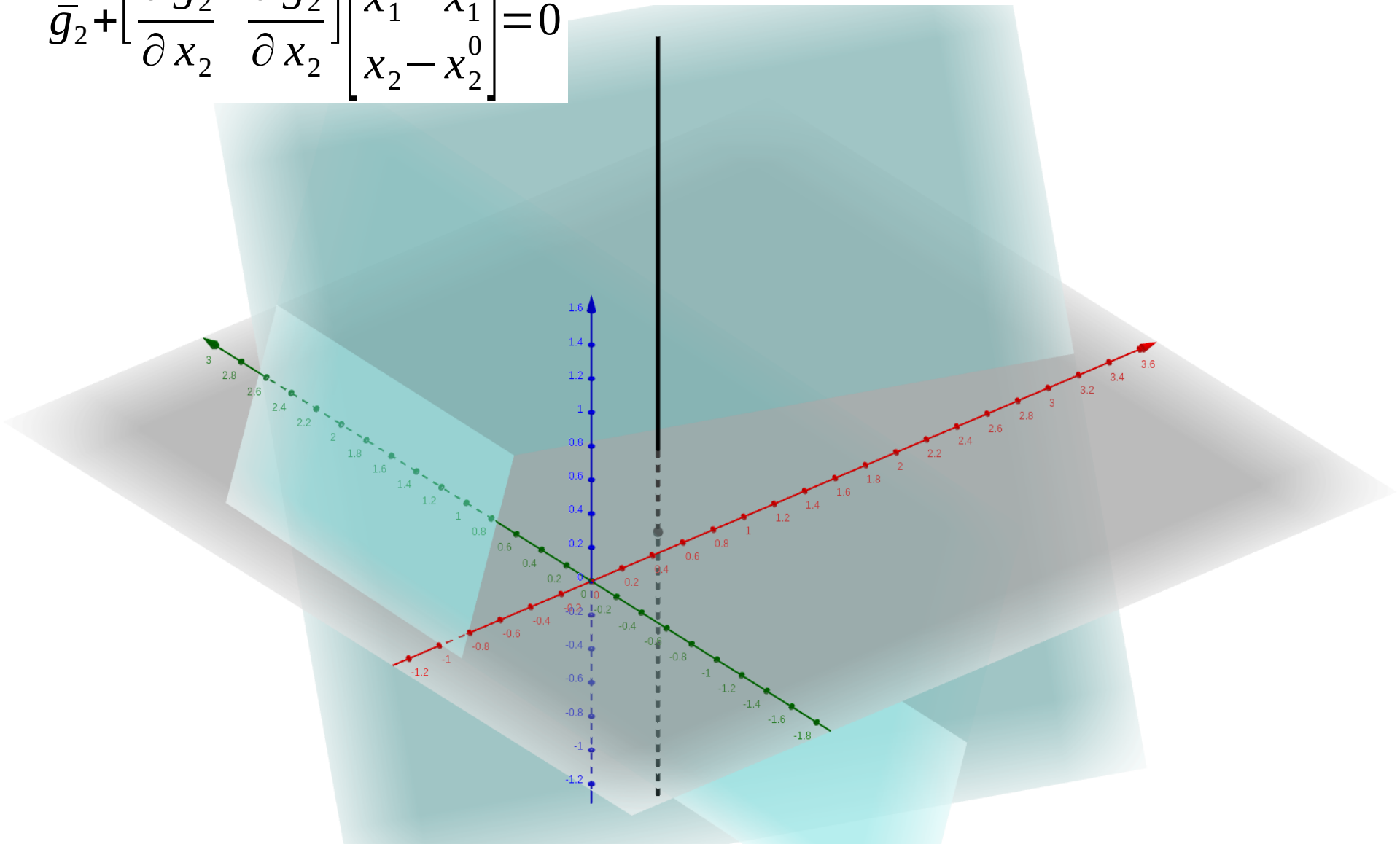
$$\bar{g}_2 + \begin{bmatrix} \frac{\partial g_2}{\partial x} & \frac{\partial g_2}{\partial y} \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix} = 0$$



$$\bar{g}_1 + \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} \end{bmatrix} \begin{bmatrix} x_1 - x_1^0 \\ x_2 - x_2^0 \end{bmatrix} = 0$$

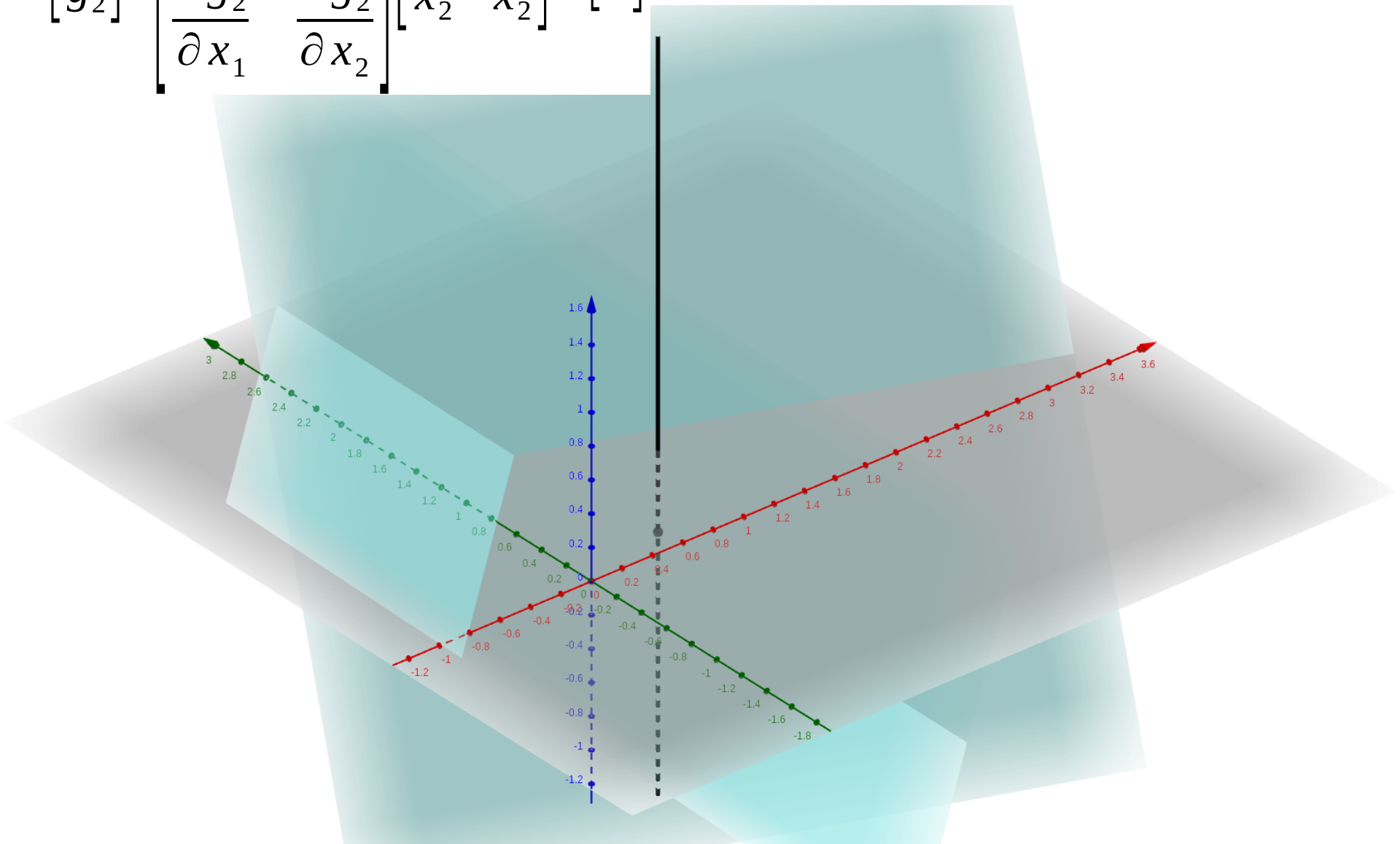
$$\bar{g}_2 + \begin{bmatrix} \frac{\partial g_2}{\partial x_2} & \frac{\partial g_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} x_1 - x_1^0 \\ x_2 - x_2^0 \end{bmatrix} = 0$$

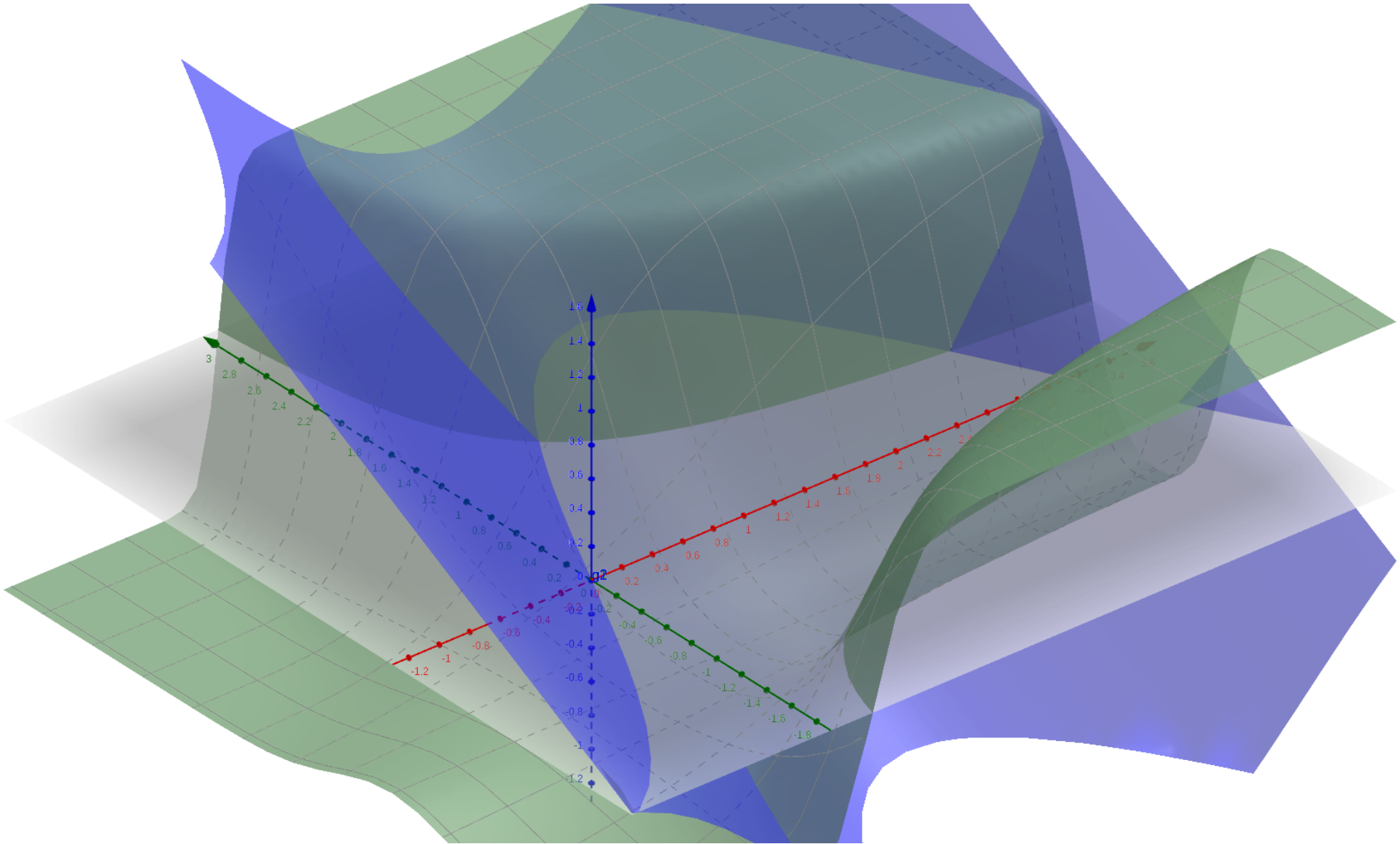
$$x \in \mathbb{R}^2$$



$$\begin{bmatrix} \bar{g}_1 \\ \bar{g}_2 \end{bmatrix} + \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} x_1 - x_1^0 \\ x_2 - x_2^0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$g(x_0) + \frac{\partial g}{\partial x}(x_0)(x - x_0) = 0$$





Newton step

$$\Delta x = -\left(\frac{\partial g}{\partial x}\left(x^k\right)\right)^{-1} g\left(x^k\right)$$

$$x^{k+1} = x^k + \Delta x$$

Convergence

$$x^2 - 2, x_0 = 100$$

- Quadratic near to the solution
- Number of correct digits doubles each iteration

100.000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

50.0100000000000000000000000000000000000000000000000000000000000000000000000000000000000000

25.0249960007998400319936012797440511897620475904819036192761447710457908418316336732653469

12.5524580467459029825975610633334549412082572619915124668270193615937313749375525928557755

6.3558946949311400346340728291734818162471708100439903105518281993476227364099693199814164 9

3.3352816092804338382821264034622432221395467534338596651371731309167015090540808854585527 5

1.9674655622311490233742943200226956429961066884206785013631928921715893491740740409009733 8

1.4920008896897231264880530162450767425038090919908433488103235667601318503684424970969324 3

1.41624133203894376448081891385156098091309363702026785530856022656341364998549645041300827

1.41421501405005317585927106194897166768116945249787874089207131893236997785448800159000403

1.41421356237384011442813894696613640381815292297399094713705788849173532023881516700177698

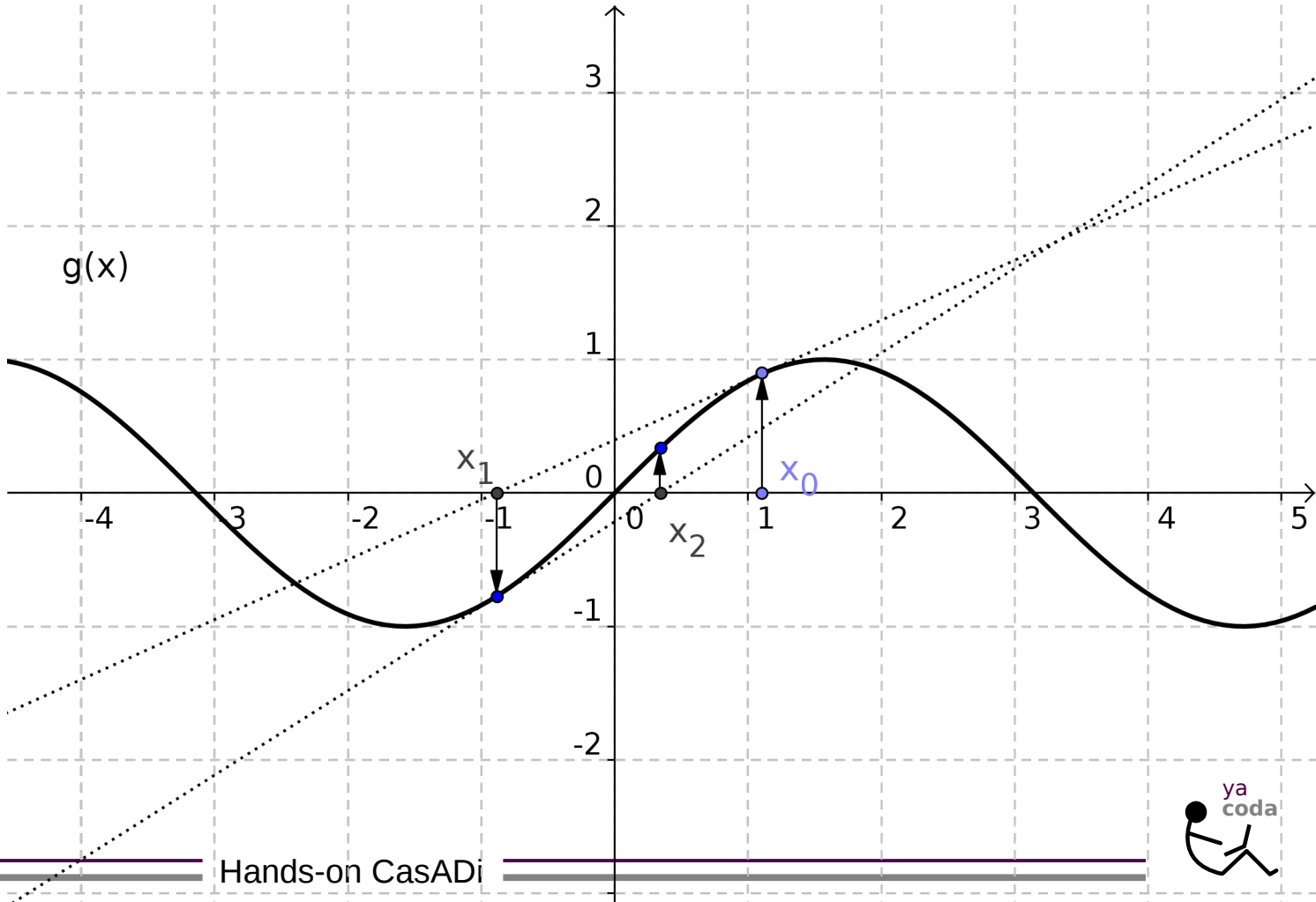
1.414213562373095048801688920475241871636128564197555660572087917336356907562561544251100 30

1.4142135623730950488016887242096980785696718753769616921111551470778848082909676831097873 4

1.41421356237309504880168872420969807856967187537694807317667973799073247846210703885038753

Newton's method is local

- Solution depends on initial guess



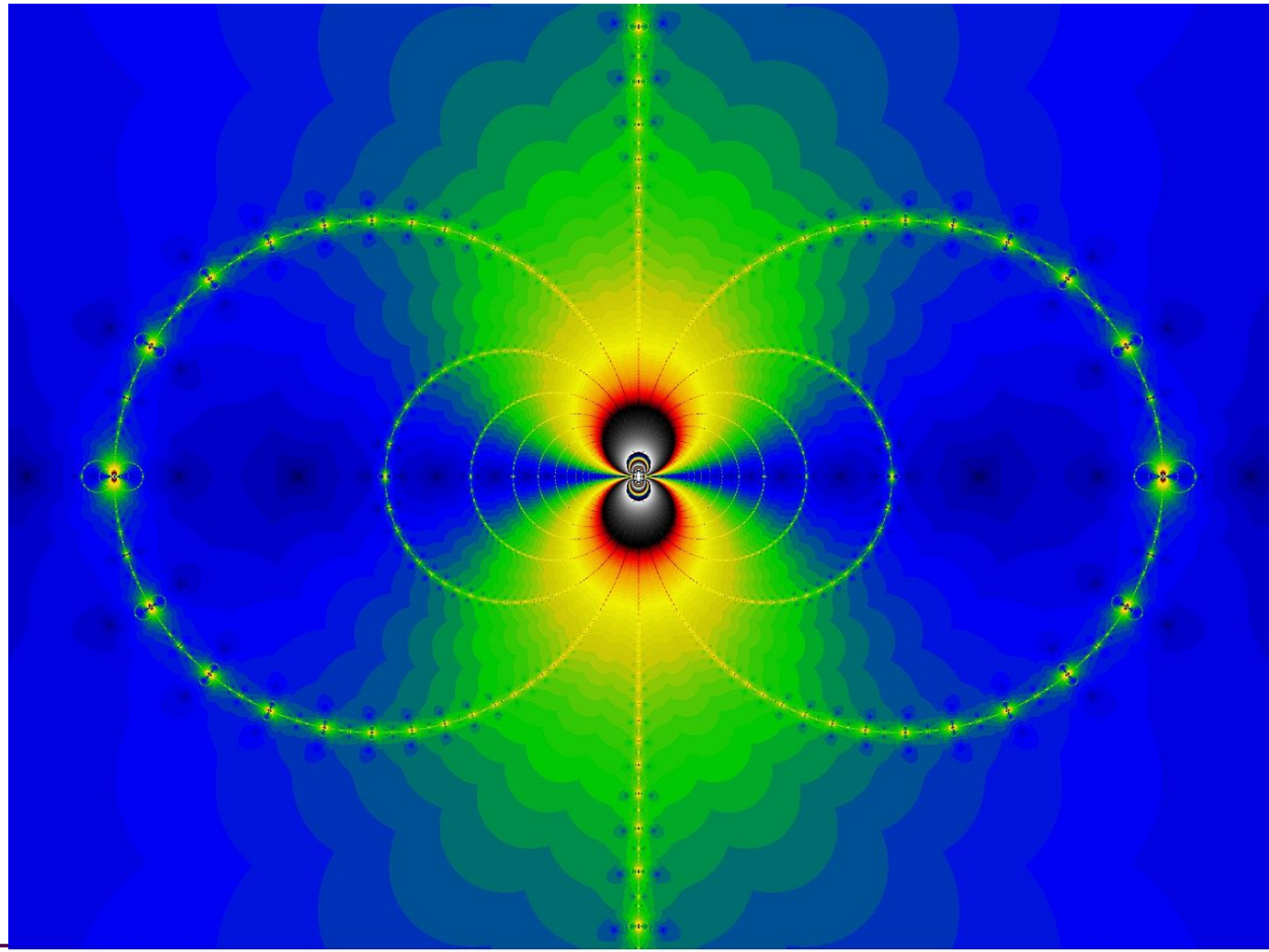
Newton's method is local

$$g(x) = \sin(x) - 1, x \in \mathbb{C}$$

$$\updownarrow$$

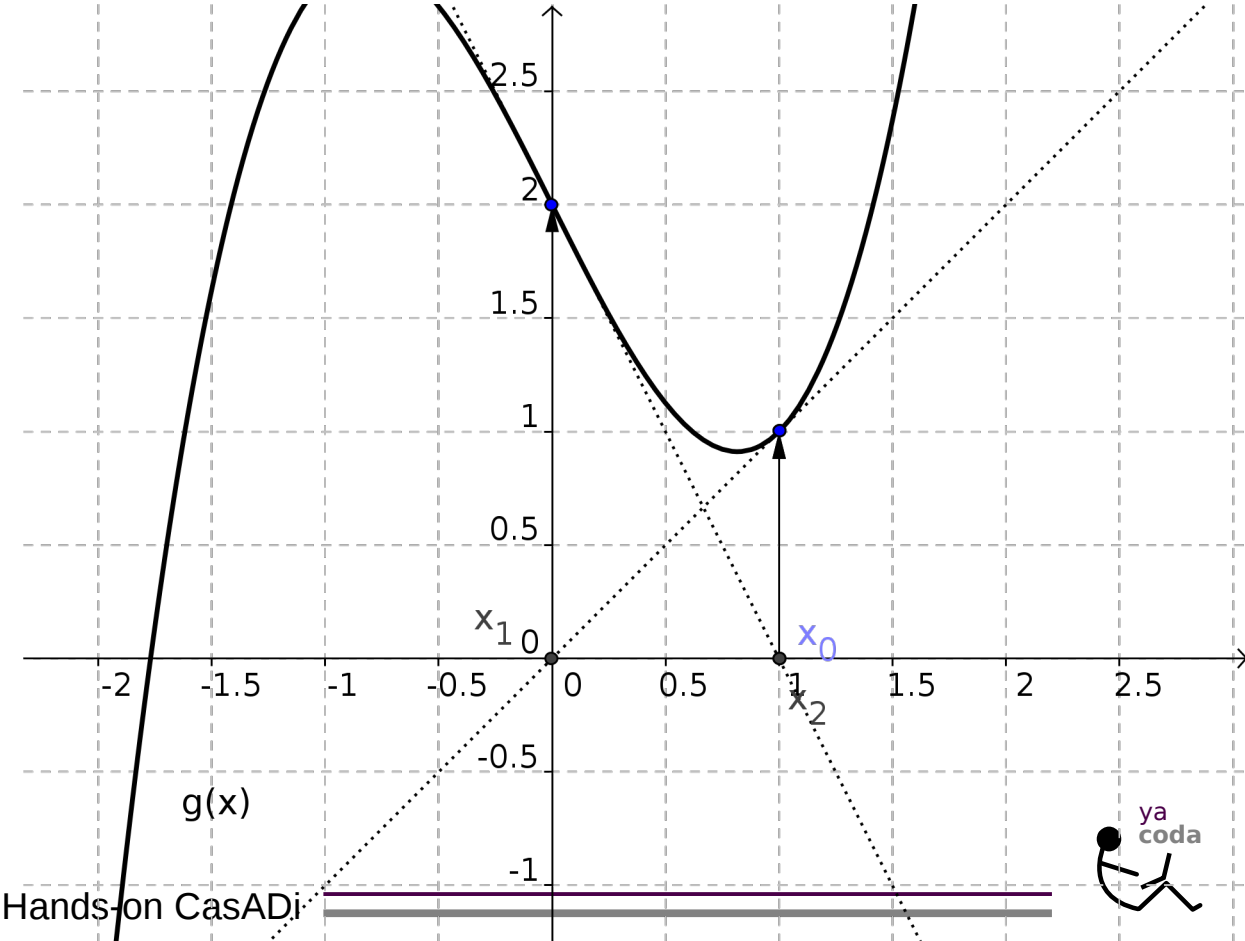
$$g(x) = [...], x \in \mathbb{R}^2$$

- Regions of attraction can be fractal



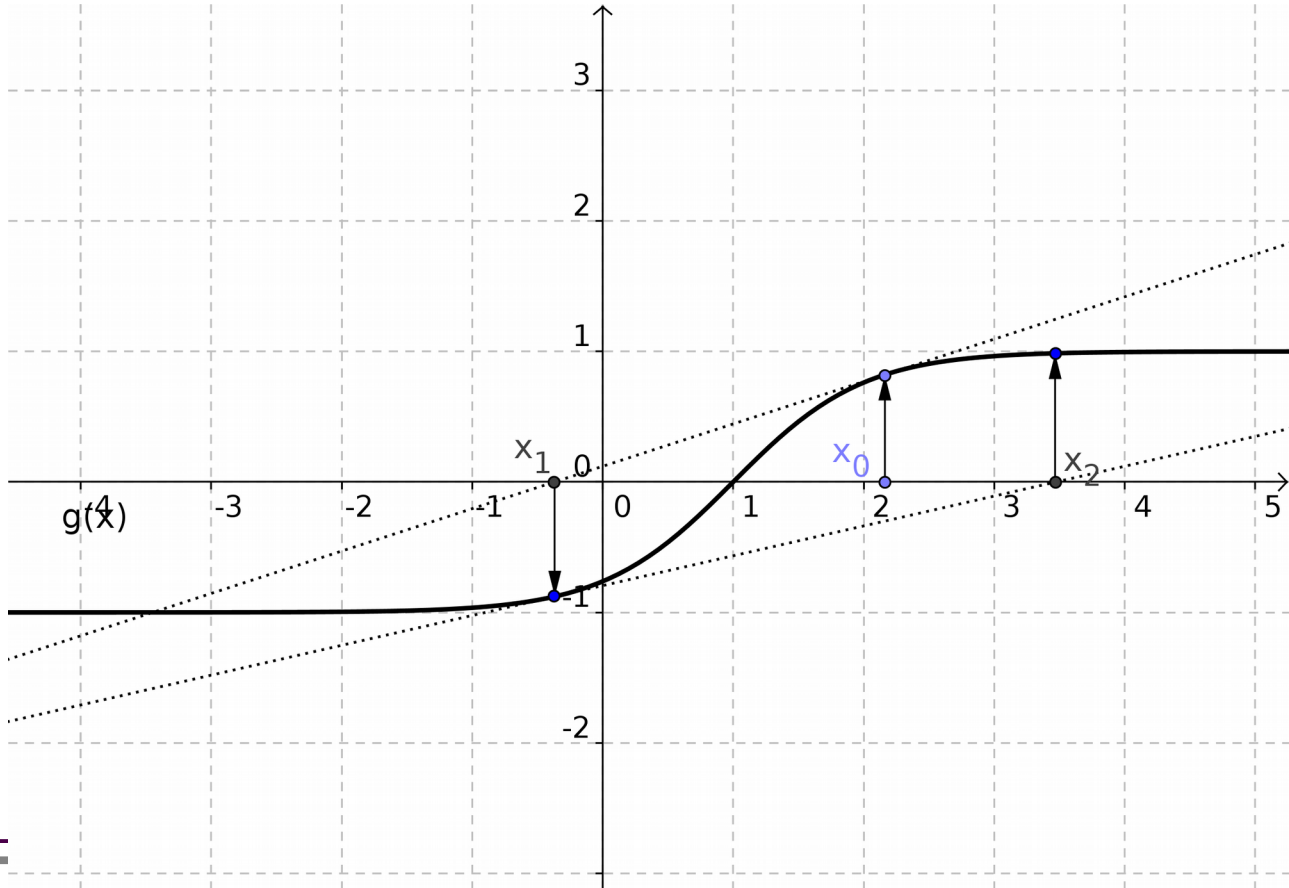
Mètode de Newton. SIN(Z) - 1 = 0 Zn+1 = (Zn * COS(Zn) - SIN(Zn) + 1) / COS(Zn)

Newton's method may cycle



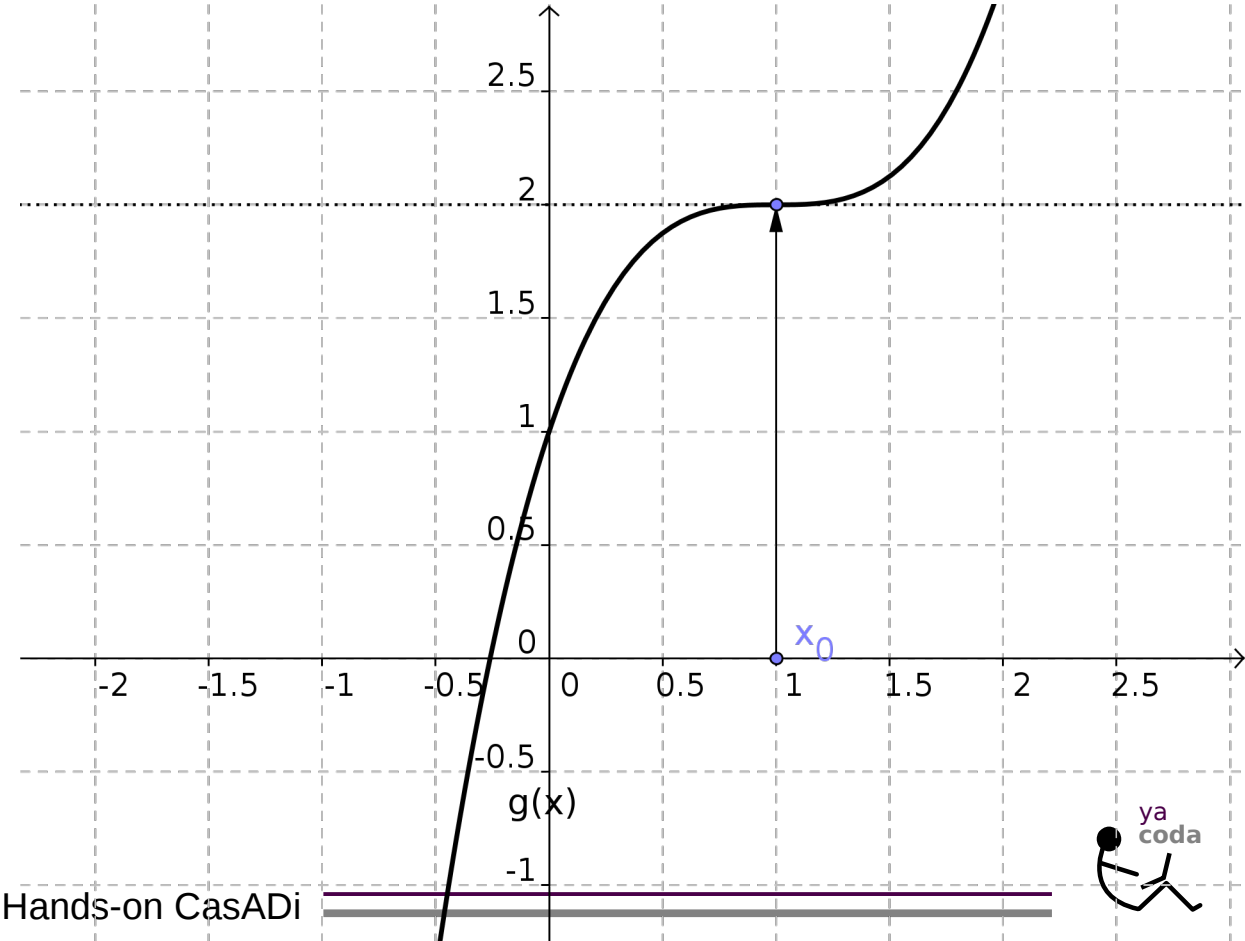
Newton's method may diverge

- "globalisation strategy": broaden the region of convergence
- Decrease Δx if not sufficient progress made



Newton step may be invalid

- 1D: Slope 0



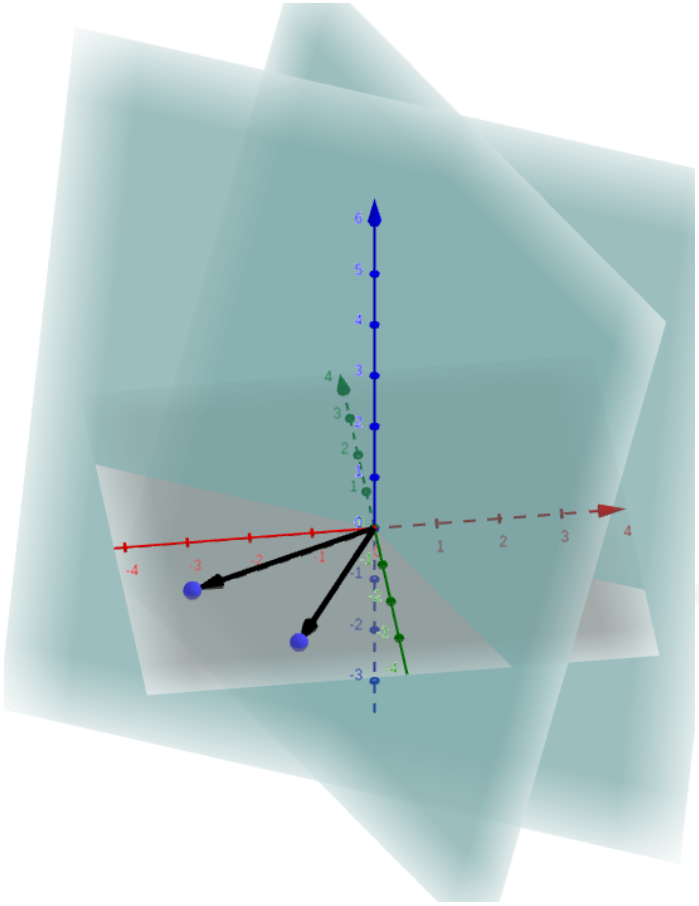
Newton step may be invalid

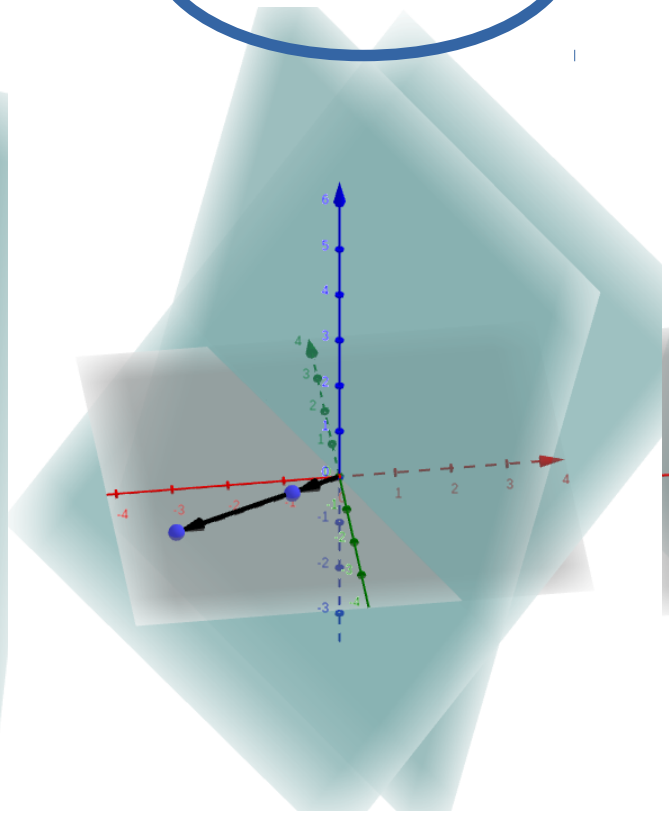
- 1D: Slope 0
- $\frac{\partial g}{\partial x}(x_k)$ not invertible

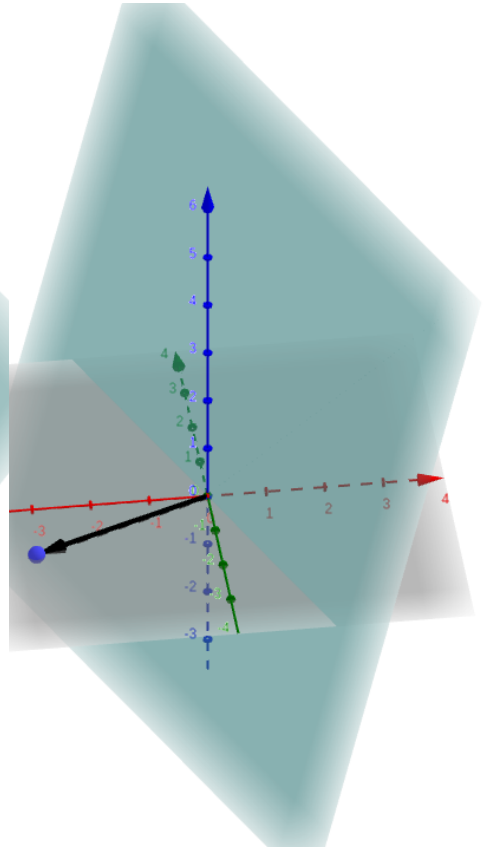
$$\begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} \end{bmatrix}$$

Transpose($\nabla_x g_1$)

Transpose($\nabla_x g_2$)







Newton's method is robust in practice

- In practice, no special precautions needed
- Jacobian $\frac{\partial g}{\partial x}(x_k)$ need not be exact
  - Approximations work
  - Updating for each iteration may not be needed

Tricks for hard real-time performance

How to compute that Jacobian?

- Derive by hand
- Finite differences
- Symbolic differentation
- Algorithmic differentiation (AD)

Re-use of subexpressions

variables

Jacobian $\frac{\partial g}{\partial x}$:

functions

$$\begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \frac{\partial g_1}{\partial x_3} & \frac{\partial g_1}{\partial x_4} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \frac{\partial g_2}{\partial x_3} & \frac{\partial g_2}{\partial x_4} \\ \frac{\partial g_3}{\partial x_1} & \frac{\partial g_3}{\partial x_2} & \frac{\partial g_3}{\partial x_3} & \frac{\partial g_3}{\partial x_4} \end{bmatrix}$$

$x \in \mathbb{R}^4, g : \mathbb{R}^4 \rightarrow \mathbb{R}^3$

Finite differences

- Cost of one column ~ cost(g)
- Limited precision
- Scaling issues

$$\frac{\partial g}{\partial x_1} \approx \frac{g\left(\begin{bmatrix} x_1+\varepsilon \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}\right) - g\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}\right)}{\varepsilon}$$

$$\begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \frac{\partial g_1}{\partial x_3} & \frac{\partial g_1}{\partial x_4} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \frac{\partial g_2}{\partial x_3} & \frac{\partial g_2}{\partial x_4} \\ \frac{\partial g_3}{\partial x_1} & \frac{\partial g_3}{\partial x_2} & \frac{\partial g_3}{\partial x_3} & \frac{\partial g_3}{\partial x_4} \end{bmatrix}$$

Finite differences

- Cost of one forward sensitivity ~ cost(g)
- Limited precision
- Scaling issues

$$\frac{\partial g}{\partial x} s \approx \frac{g(x + s\varepsilon) - g(x)}{\varepsilon}$$

Forward seed $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

Forward sensitivity

$$\begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \frac{\partial g_1}{\partial x_3} & \frac{\partial g_1}{\partial x_4} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \frac{\partial g_2}{\partial x_3} & \frac{\partial g_2}{\partial x_4} \\ \frac{\partial g_3}{\partial x_1} & \frac{\partial g_3}{\partial x_2} & \frac{\partial g_3}{\partial x_3} & \frac{\partial g_3}{\partial x_4} \end{bmatrix}$$

Algorithmic differentation (AD)

- Cost of one forward sensitivity ~ cost(g)
- Machine precision (no worries about scaling)
- Given an alorithm, $g(x)$, can construct a forward algorithm

$$\frac{\partial g}{\partial x} s = g^{forward}(x, s)$$

Forward seed $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

$$\begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \frac{\partial g_1}{\partial x_3} & \frac{\partial g_1}{\partial x_4} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \frac{\partial g_2}{\partial x_3} & \frac{\partial g_2}{\partial x_4} \\ \frac{\partial g_3}{\partial x_1} & \frac{\partial g_3}{\partial x_2} & \frac{\partial g_3}{\partial x_3} & \frac{\partial g_3}{\partial x_4} \end{bmatrix}$$

Algorithmic differentation (AD)

- Cost of one forward sensitivity ~ cost(g)
- Machine precision (no worries about scaling)
- Given an alorithm, $g(x)$, can construct a reverse algorithm

$$\left[\frac{\partial g}{\partial x}\right]^T s = g^{reverse}(x, s)$$

Reverse seed $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

$$\begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \frac{\partial g_1}{\partial x_3} & \frac{\partial g_1}{\partial x_4} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \frac{\partial g_2}{\partial x_3} & \frac{\partial g_2}{\partial x_4} \\ \frac{\partial g_3}{\partial x_1} & \frac{\partial g_3}{\partial x_2} & \frac{\partial g_3}{\partial x_3} & \frac{\partial g_3}{\partial x_4} \end{bmatrix}$$

Algorithmic differentation (AD)

- Jacobian can be composed from forward/reverse sweeps
- Reverse sweeps can be a real game-changer
  - Unconstrained optimization: deep learning
- Sparsity can be exploited
  - Can combine seeds when rows/cols structurally independent

yacoda
Let's do some coding...

1. rootfinding

Slides/solutions:
ocp2020.yacoda.com