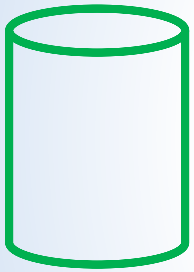
Minimizing in 3D

Method 4: Cyclic Coordinate Search

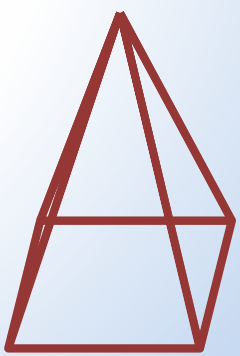
The Premise

With a 3D object, any cross-section forms a 2-dimensional object.



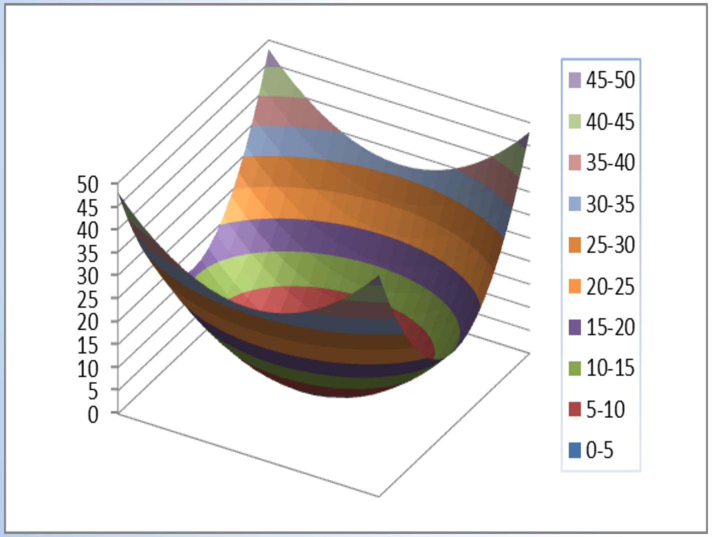
Vertical = rectangle
Horizontal = circle
Oblique = oval

Vertical = triangle/trapezoid
Horizontal = square
Oblique = triangle/trapezoid



The Premise

Similarly, with a 3D function any cross-section will form a 2D function.



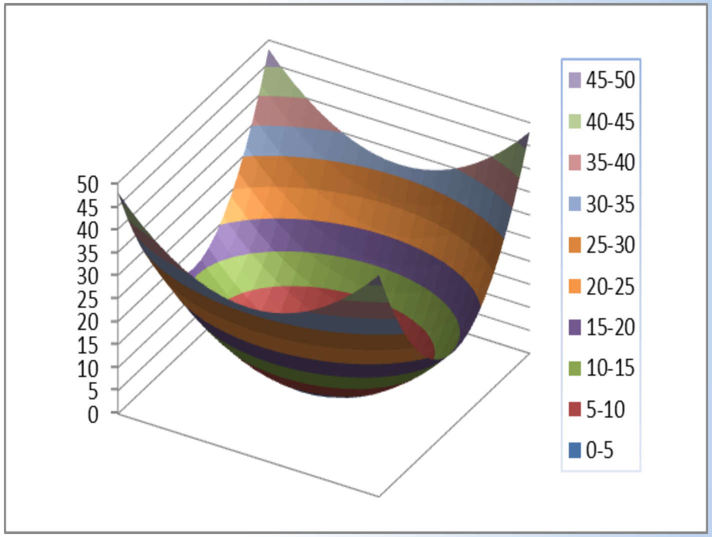
Looking at the cross-sections of this 3D graph, what shapes do you see?

The Premise

This is the graph of the function

$$f(x_1, x_2) = (x_1)^2 + 2(x_2)^2.$$

The reason you see parabolas is that, if we set x_2 to a constant, we get $f(x_1) = (x_1)^2 + c$, a parabola. You will also get a parabola if x_1 is constant.



The horizontal cross-sections, ellipses, are formed when f is constant: $(x_1)^2 + 2(x_2)^2 = c$.

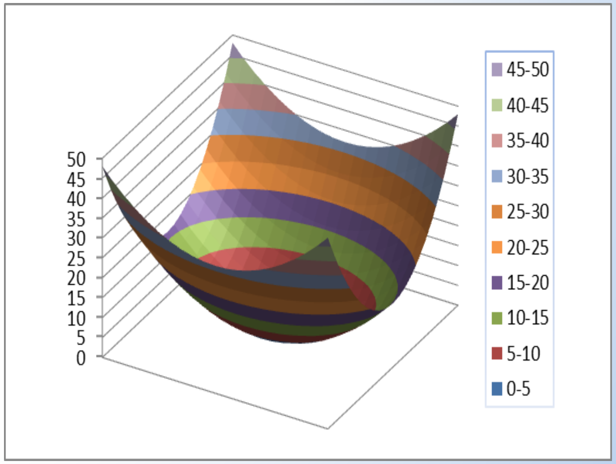
The Premise

The reason this is useful information is that you already have programs that minimize one-variable functions (which create 2-dimensional graphs). These include the golden section method and the slopes method.

The Premise

A 2-D minimum won't find the minimum of the entire graph, but it will find the minimum of a cross section.

Then if we repeat a few times, we can get the minimum of the graph.



Practice Problem 1

Use the function $f(x_1, x_2) = (x_1 - 3)^2 + (x_2 + 5)^2$ and starting point (-2, -1).

a) Rewrite the function substituting $x_2 = -1$.

b) Use any of your minimization programs to find the minimum value of the resulting function.

c) Rewrite the original function substituting x_1 with its minimized value from b, then minimize x_2.

Practice Problem 2

Using the function $f(x_1, x_2) = (x_1)^2 + 2(x_2)^2 + 2x_1x_2$ and starting point (0.5, 1), execute one cycle of the algorithm (meaning, minimize x_1 and then x_2 and write down the resulting values).

Calculate the value of $f(x_1, x_2)$ at the original point and at the new point. Also, calculate the vector from the original point to the new point.

Introducing a New Variable

The next step is *not* to just adopt the new point. Instead, we create a vector from the old point to the new point and minimize from the new point in that direction. The math looks like this:

new point: (p_1, p_2) = known

vector: $\langle v_1, v_2 \rangle$ = known   scalar: a = unknown

end point = new point + scalar · vector

end point = $(p_1 + a \cdot v_1,\ p_2 + a \cdot v_2)$

Practice Problem 3

Use the function $f(x_1, x_2) = (x_1)^2 + 2(x_2)^2 + 2x_1x_2$ with the new point (-1, .5) and vector <-1.5, -.5> as calculated in Practice Problem 2.

3a. Write the end point as $(p_1 + a \cdot v_1, p_2 + a \cdot v_2)$ with all the known numbers filled in. The only variable should be a.

3b. Plug the end point into the function; it is not necessary to simplify.

Introducing a New Variable

After forming the new point from the old point, known vector, and scalar a, plug the x_1 and x_2 of the end point into the equation. This gives an equation in one variable.

The next step is to minimize that equation. This is known as the “acceleration step”.

Practice Problem 4

4a.Use one of your 1-variable minimization programs to minimize the equation from 3b.

4b. Plug the minimizing value of a into the coordinates for the end point to get numerical values for x_1 and x_2. This is the actual end point.

4c. Find the value of the function at this point.

The Last Step

The last step is to repeat the process until the change in the function value with each step is very small (below a given tolerance).

All the steps, in order:

1. Minimize x_1 with x_2 constant, then minimize x_2 with x_1 constant. This is the new point.
2. Find the vector from the original point to the new point, then form the end point using scalar a.
3. Minimize the function formed by plugging the end point into the equation.
4. Solve for the end point and repeat until the change in f is below tolerance.

Practice Problem 5

Write a program that will perform a cyclic coordinate search around a given function $f(x_1, x_2)$ with a given starting point (a, b).

One way to do this is to copy and paste your minimization code three times, with slight modifications; there might be a better way.

Write your program to loop until the change in the value of f is below 0.0001.

Save this program! Document your code!