Geometry and Optimization

Simon Flöry

These slides discuss the basics of geometry optimization in parametric modelling environments. They comprise edited course material of a lecture on *CAAD and Geometry* for students of Architecture, taught at Vienna University of Technology in winter 2012/2013.

RECHENRAUM

software and consulting

About the Author Simon Flöry holds a PhD in mathematics from Vienna University of Technology and has several years of research, working and teaching experience in Architectural Geometry, Applied Geometry and Geometry Processing. Having a great interest in various aspects of software development, he has been maintaining and contributing to open-source software projects for more than a decade. Since 2012, he is leading the Vienna-based software and consulting company Rechenraum e.U..

Optimization manages limited ressources.

Optimization guides complex processes.

Optimization imitates nature.

Functions

Optimization

Examples

Functions

We describe a close relation between parametric models and the mathematical notion of a function. We identify a special type of a parametric model (and a function) that is the base of any optimization.

Consider two points E and F.

Consider two points E and F. The expression

$$(1-x) \cdot E + x \cdot F$$

describes for varying x all points on the line connecting E and F.

Consider two points E and F. The expression

$$(1 - x) \cdot E + x \cdot F$$

describes for varying x all points on the line connecting E and F.

Consider two points E and F. The expression

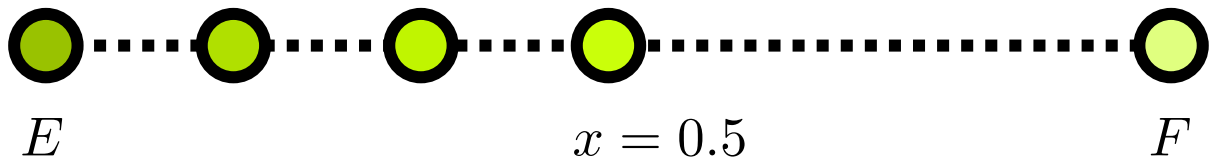
$$(1 - x) \cdot E + x \cdot F$$

describes for varying x all points on the line connecting E and F.

Consider two points E and F. The expression

$$(1 - x) \cdot E + x \cdot F$$

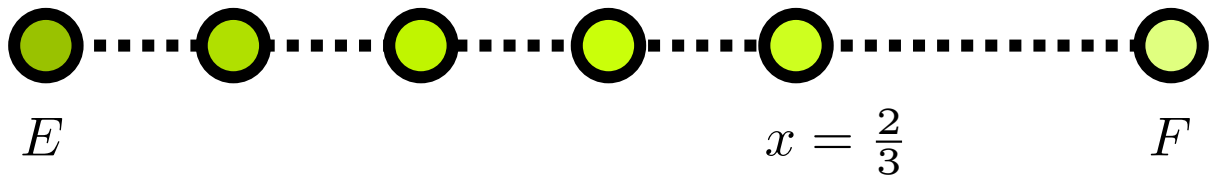
describes for varying x all points on the line connecting E and F.



Consider two points E and F. The expression

$$(1 - x) \cdot E + x \cdot F$$

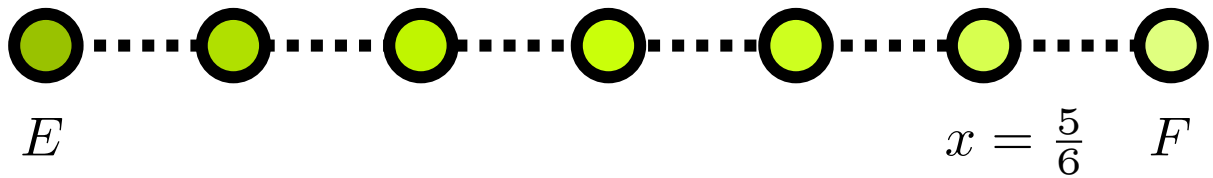
describes for varying x all points on the line connecting E and F.



Consider two points E and F. The expression

$$(1 - x) \cdot E + x \cdot F$$

describes for varying x all points on the line connecting E and F.



Consider two points E and F. The expression

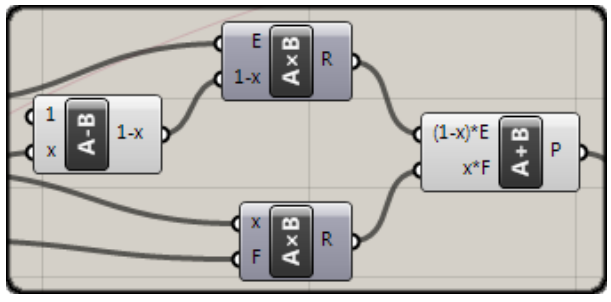
$$(1 - x) \cdot E + x \cdot F$$

describes for varying x all points on the line connecting E and F.

Consider two points E and F. The expression

$$(1-x) \cdot E + x \cdot F$$

describes for varying x all points on the line connecting E and F.



Example: affine_combination

For given x, let us write $P(x)$ for the point on the line connecting E and F.

The expression

$$P(x) = (1 - x) \cdot E + x \cdot F$$

is not only a formal description of a parametric model but a *function* in a mathematical sense.

We have

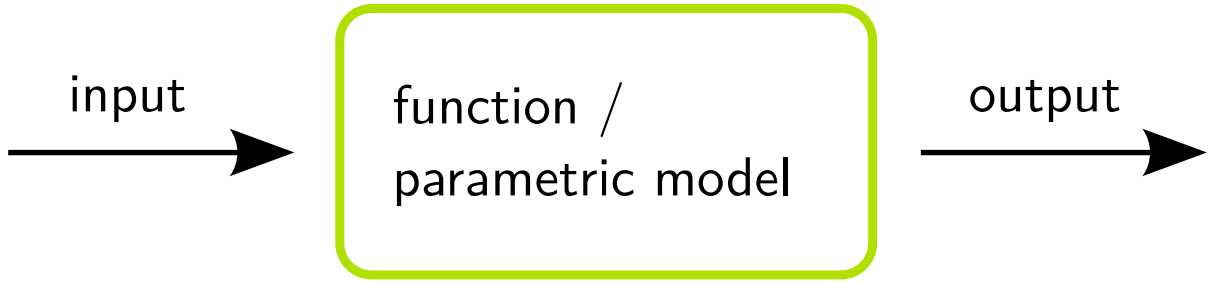
- two constant inputs E and F

We have

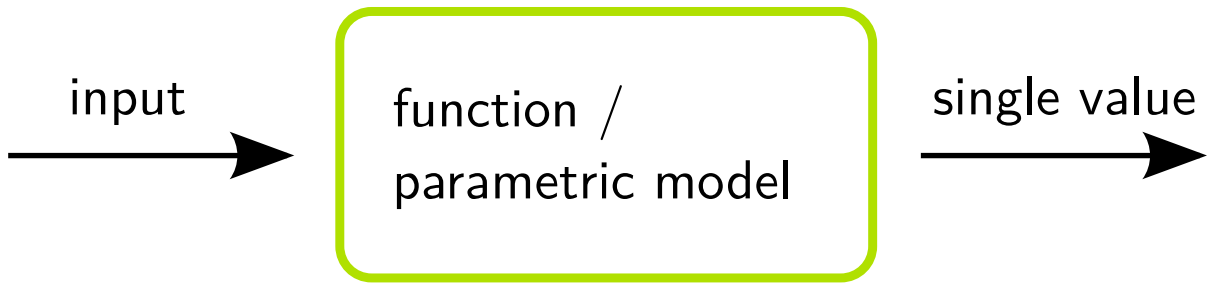
- two constant inputs E and F
- a single *variable* input x

We have

- two constant inputs E and F
- a single *variable* input x
- and the two-dimensional coordinates of $P(x)$ as output.



A function may have any kind and number of inputs and outputs: a single or several points, entire surfaces, single values, ...

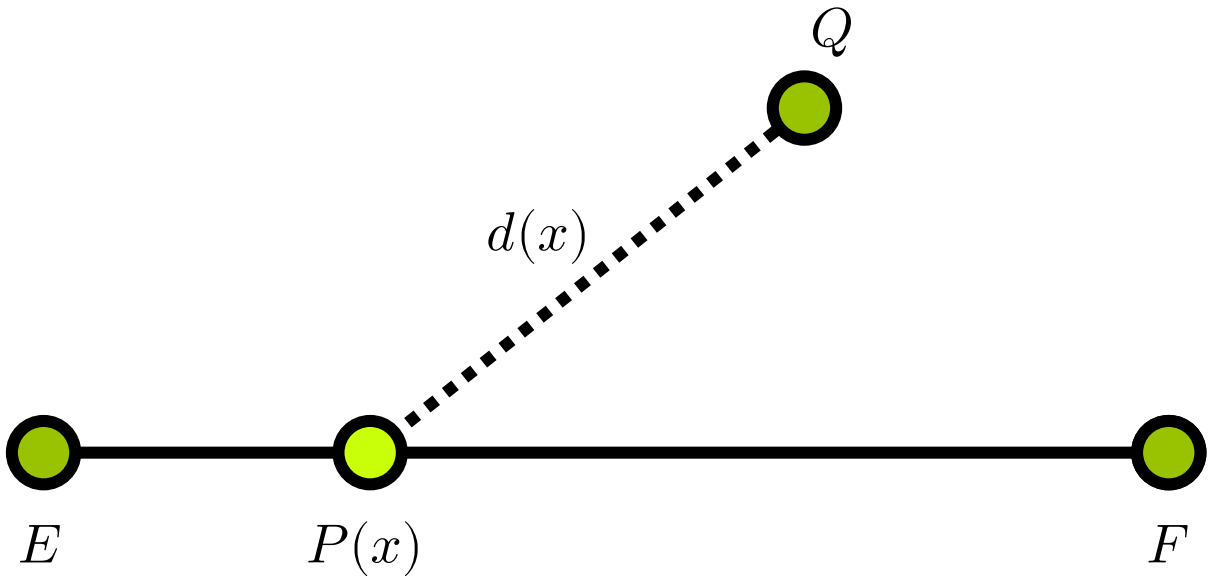


Functions with single valued output are special. We may think of the output as a *quality measure* that rates the input.

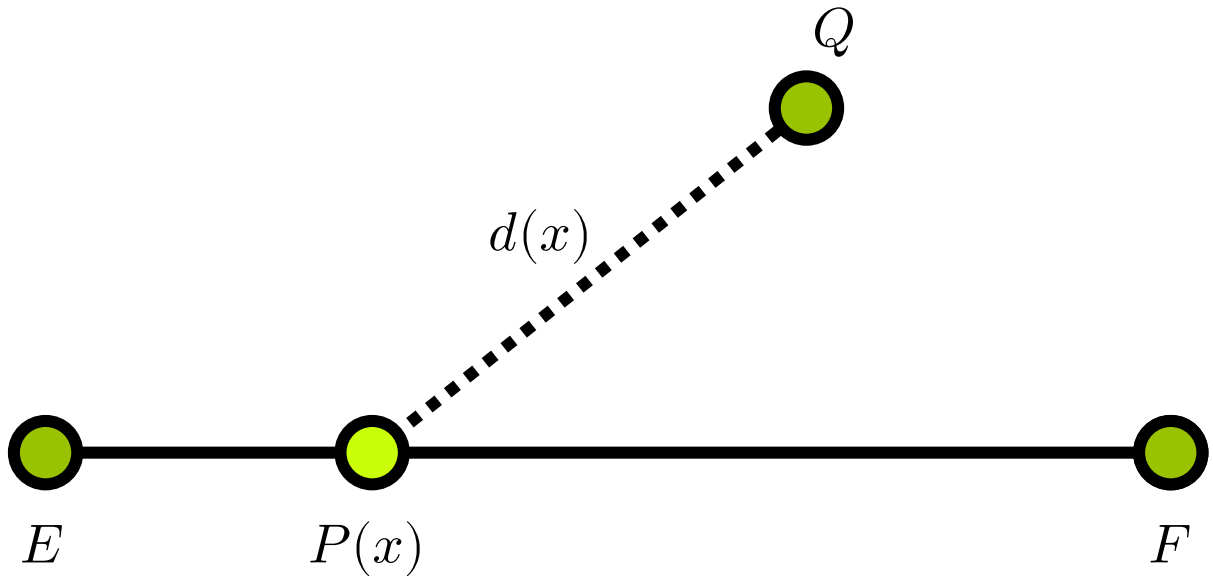
Welcome to the world of optimization: Here we deal with functions of arbitrary input with a single value as output. Our goal is to minimize or maximize this quality measure.

Optimization

We will see why derivatives of functions are important for optimization. As derivatives are not available in all cases (e.g. in a parametric modelling context), we discuss strategies to get around computing explicit derivatives.

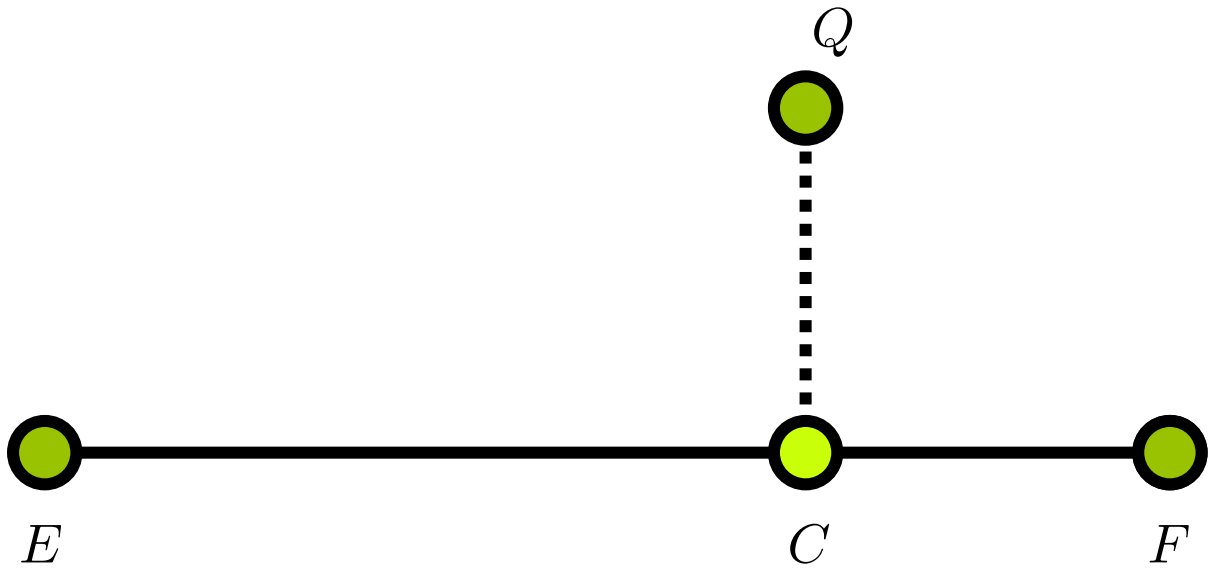


Let's find that point on the connecting line of E and F that is closest to a third point Q.

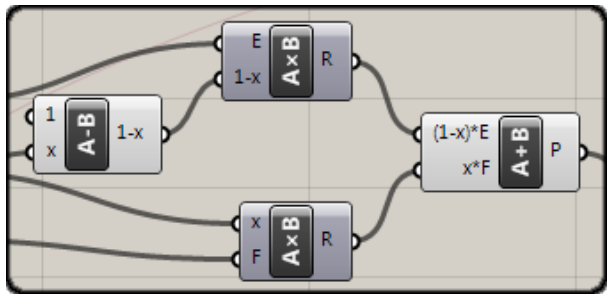


We build a parametric model suitable for optimization, that evaluates the (squared) distance of $P(x)$ to Q for varying x,

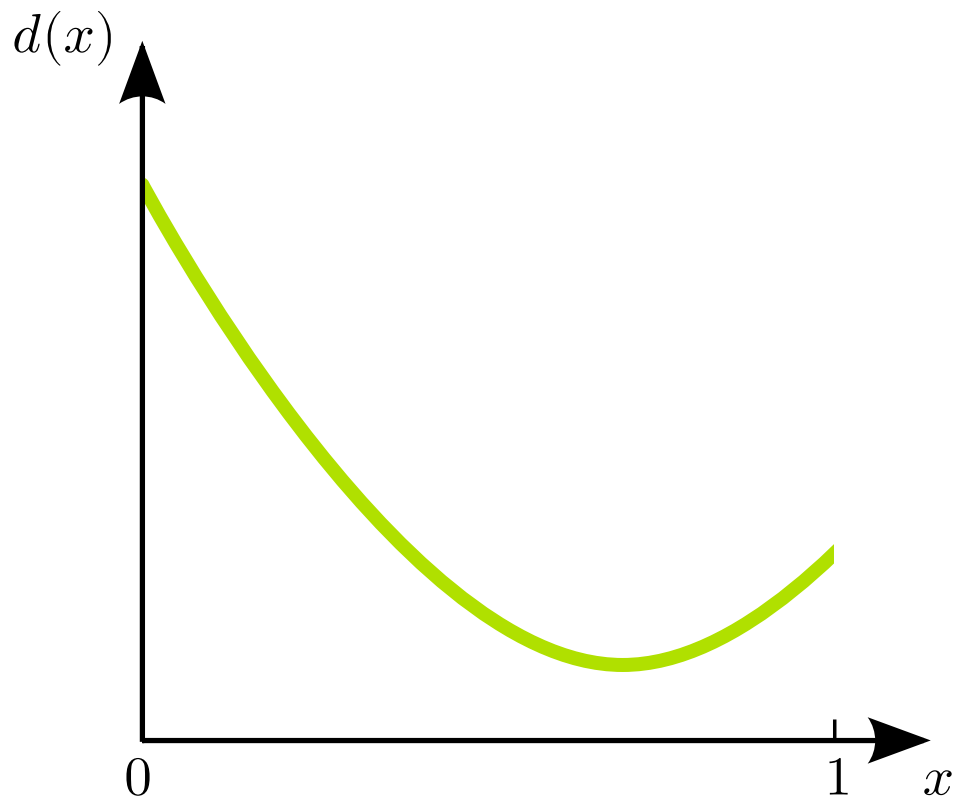
$$d(x) = (P(x) - Q)^2.$$



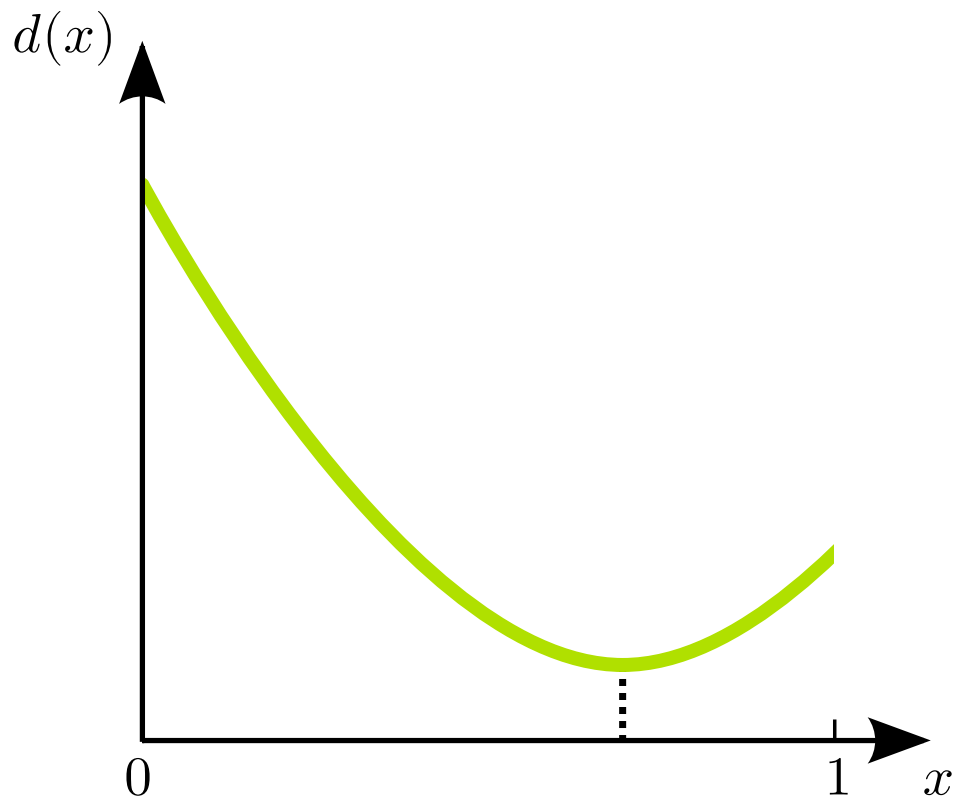
As $P(x)$ moves from left to right, the distance decreases until the closest point C. Then, the distance grows again.



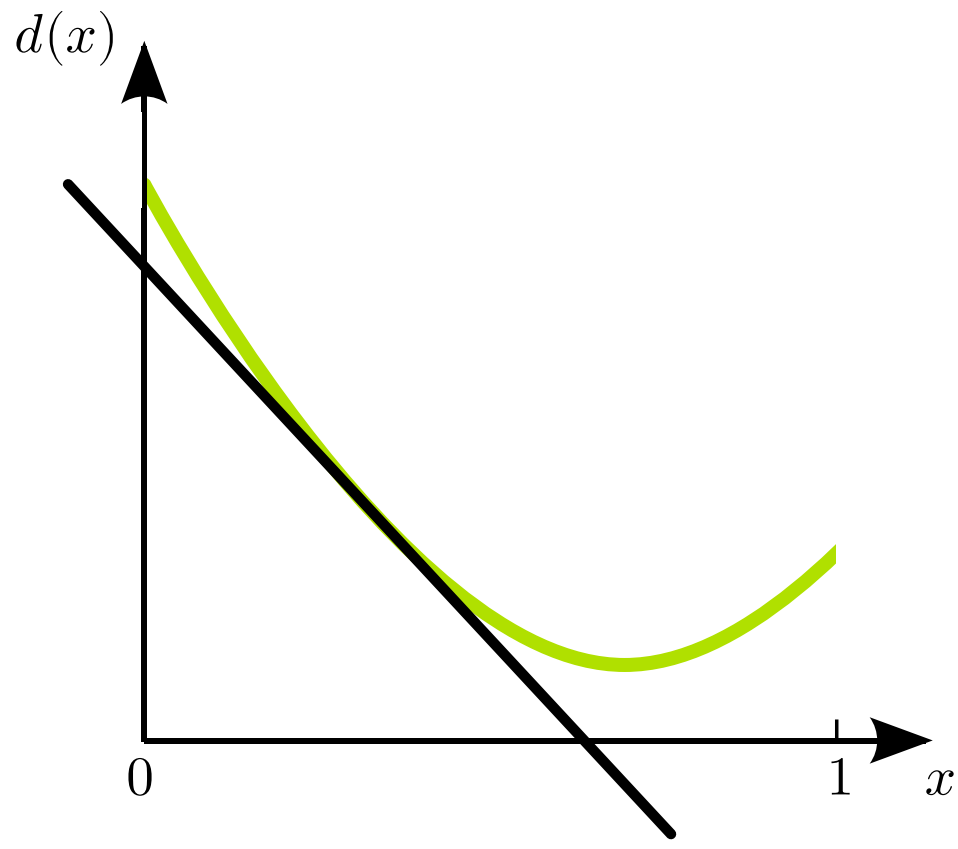
Example: closest_point



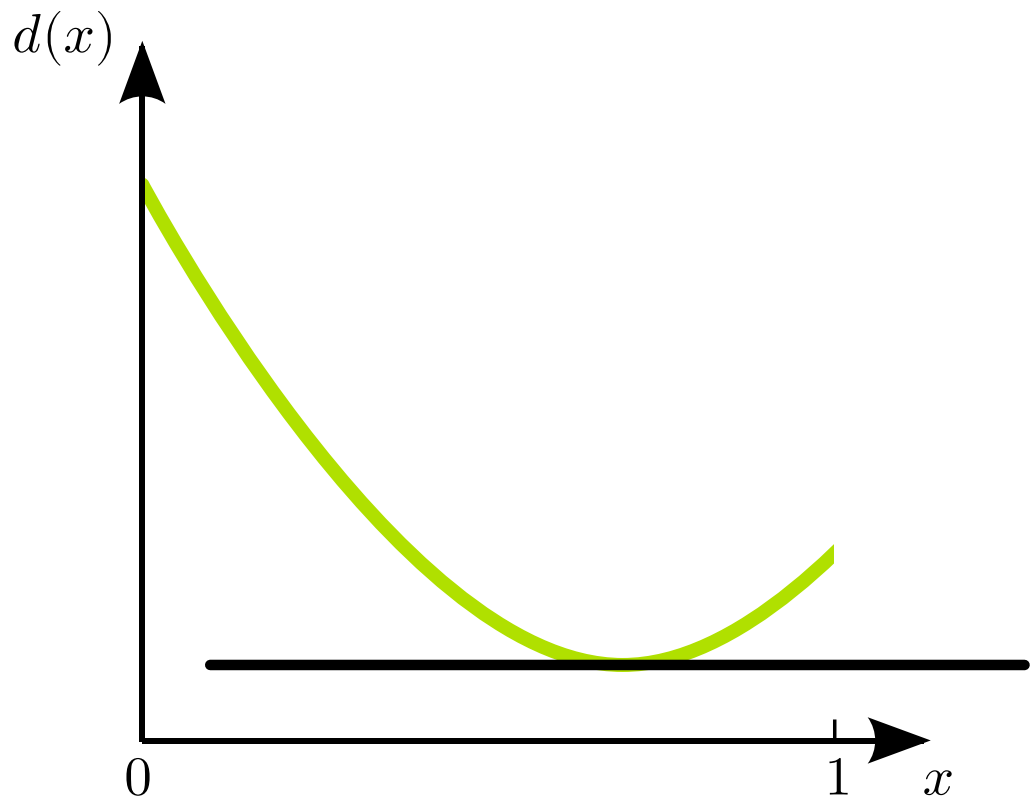
We visualize the changing distance in a chart. For each value of the variable input x, we write down the single valued output of our model.



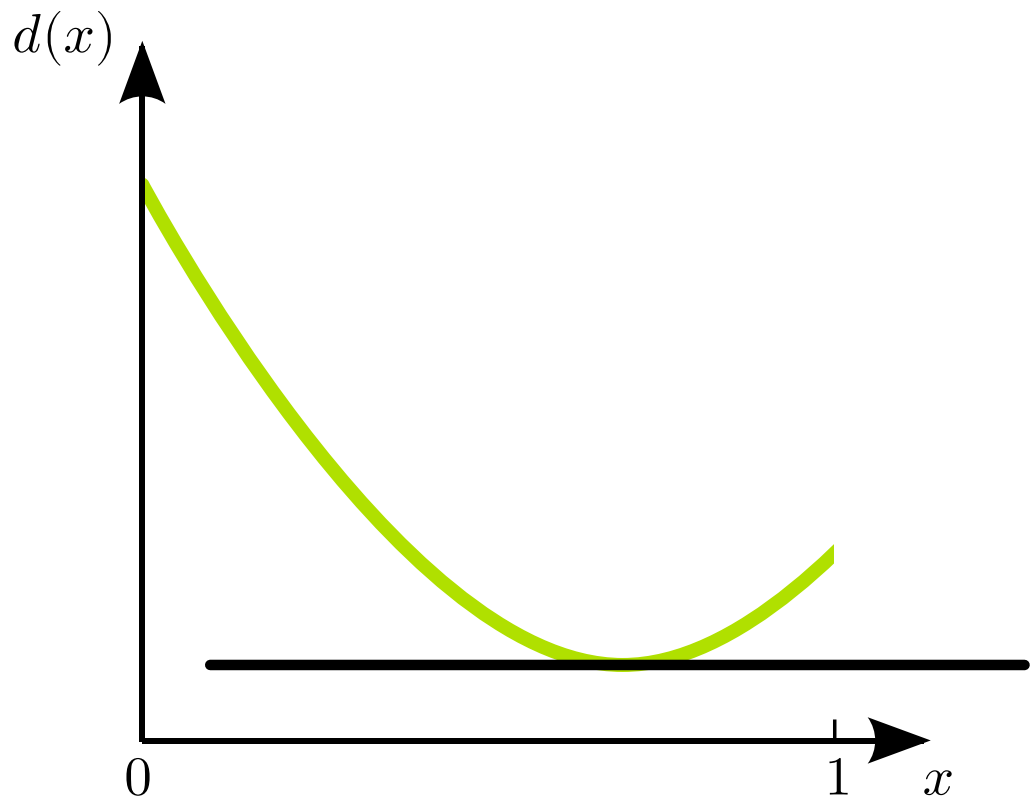
We easily spot the closest point configuration in the chart.
How may we describe the solution formally?



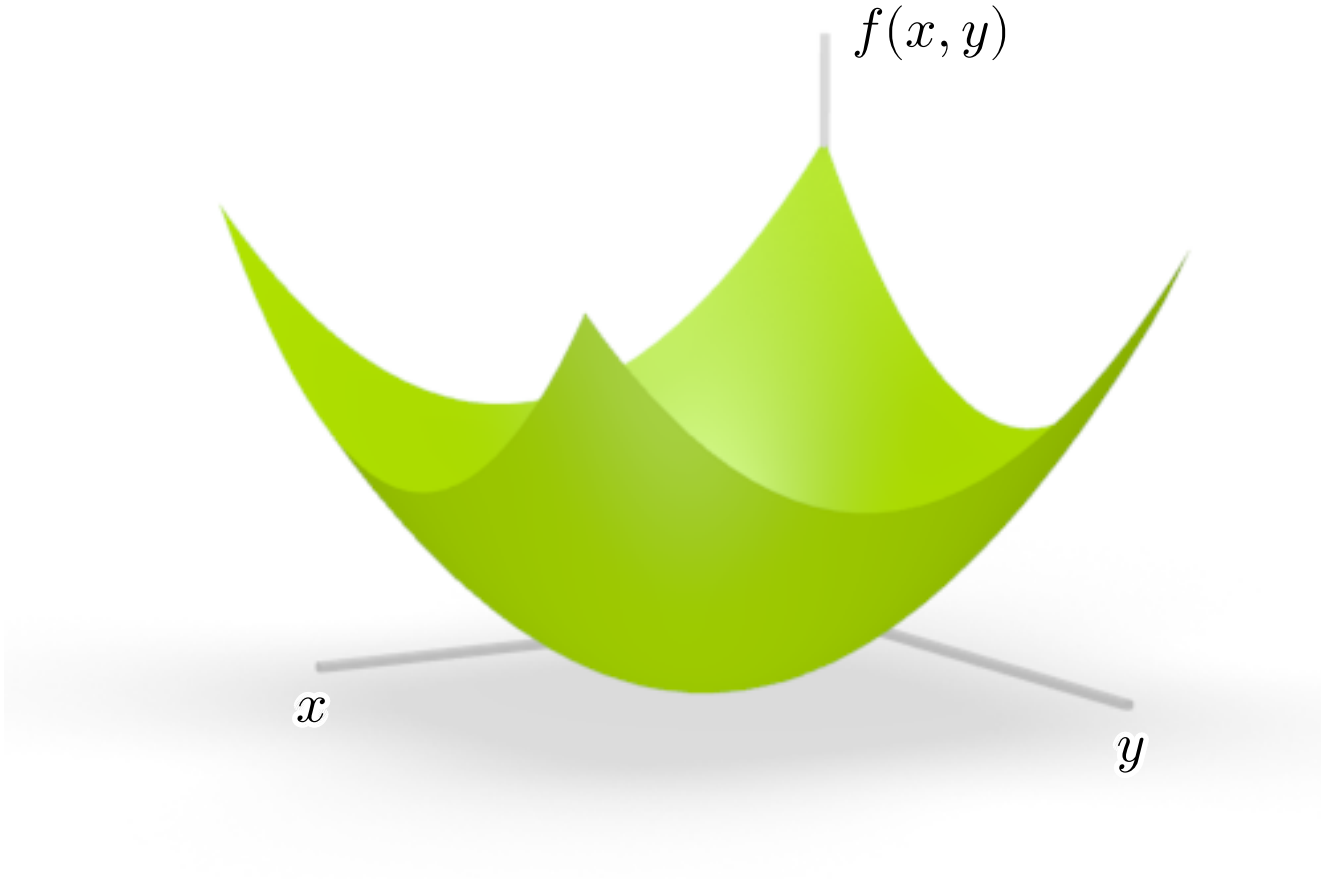
Recall, that a *tangent* is that line approximating a given curve best in a point, and that the tangent's slope is given by a function's derivative.



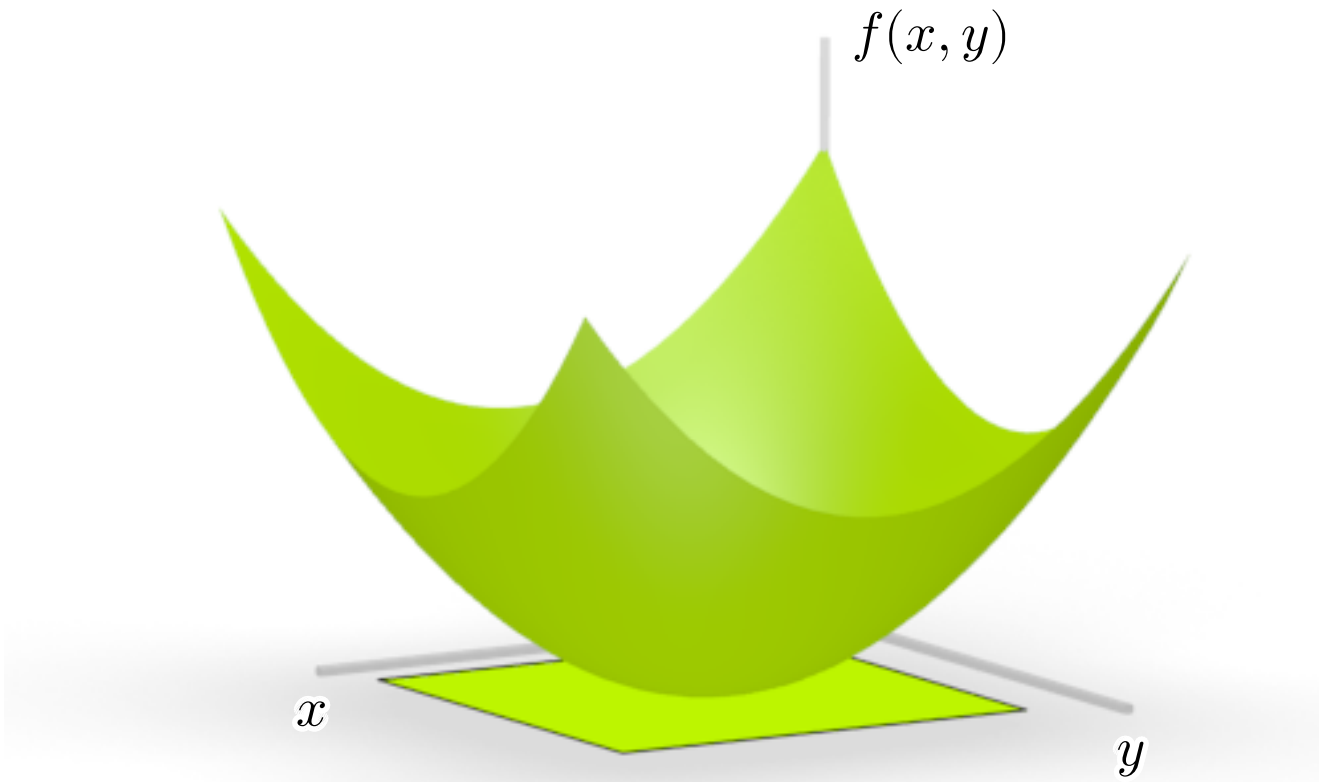
Obviously, the solution to our optimization problem is that point, in which the tangent is horizontal, e.g. has no slope as the function's derivative is 0.



This leads to an important first conclusion: For parametric models with single variable input and single valued output, the derivative must vanish in an optimal point.



What about parametric models with multiple variable inputs? As their output is always single valued, we may still generate charts (over multi-dimensional domains).



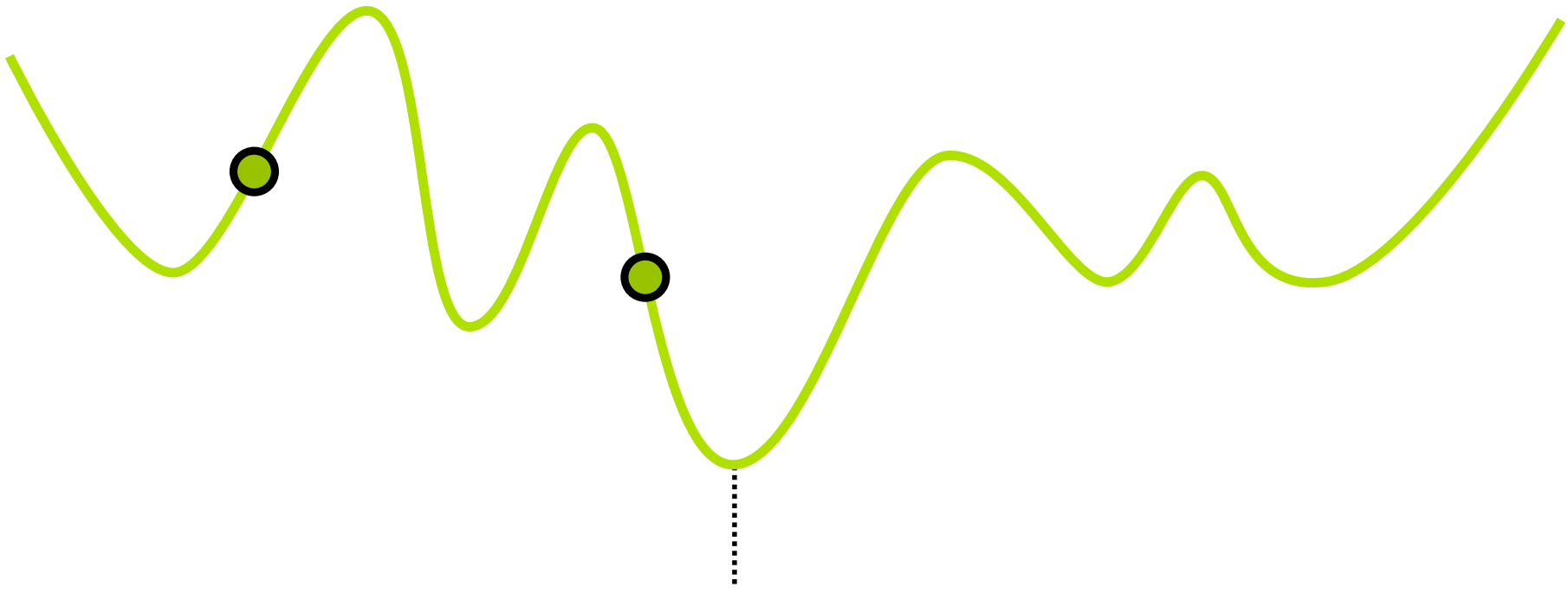
In an optimal point, the tangent plane to the chart is horizontal. The gradient, a generalization of derivatives to functions with multiple variable inputs, must be zero.

$$f'(x)!$$

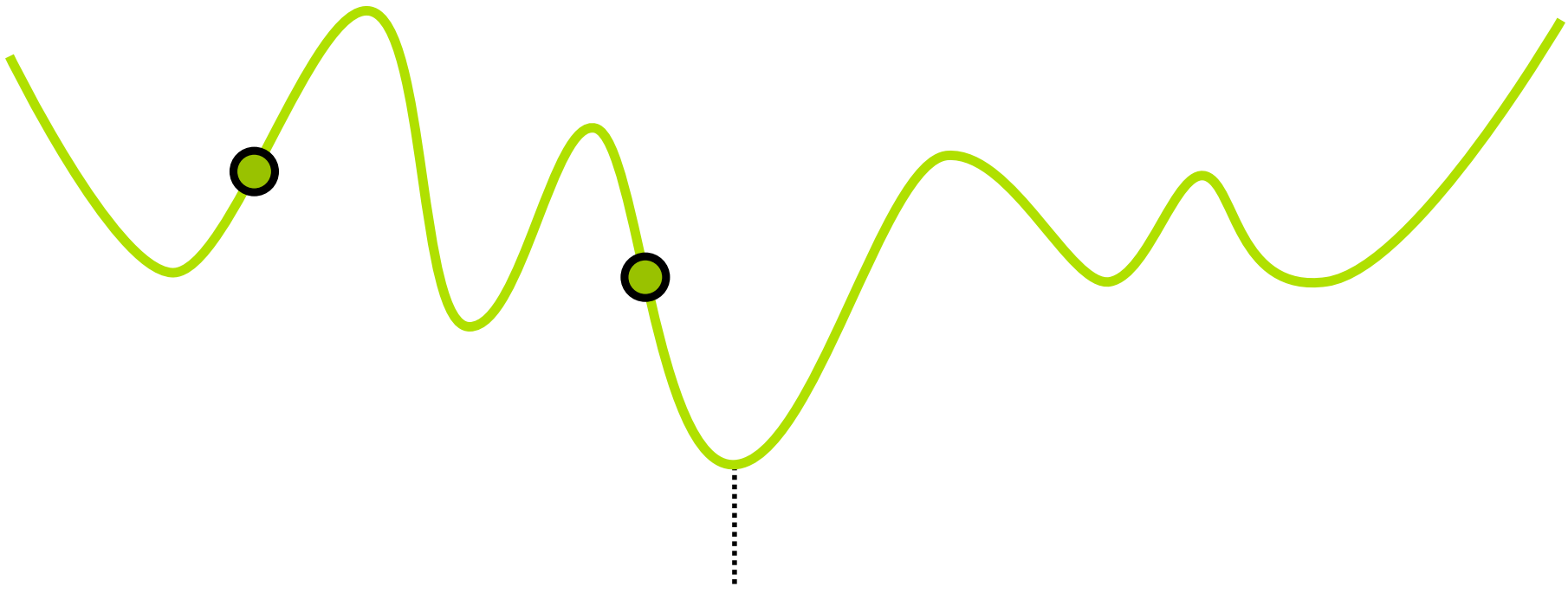
Nearly all local optimization algorithms build on gradient information to compute a point of vanishing gradient.

$$f'(x)?$$

If derivatives are not available (e.g. in a parametric modelling environment), one estimates gradients numerically.



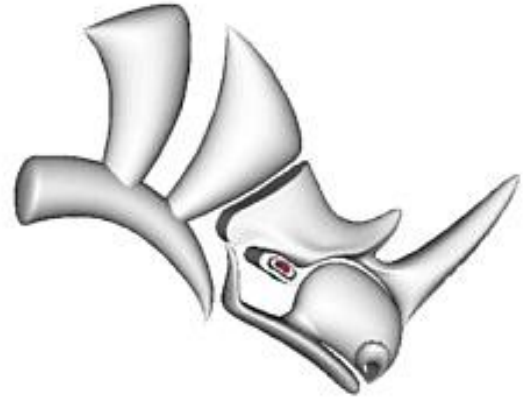
A word of warning: Relying on derivatives for optimization works only if we start in the vicinity of the optimal solution.



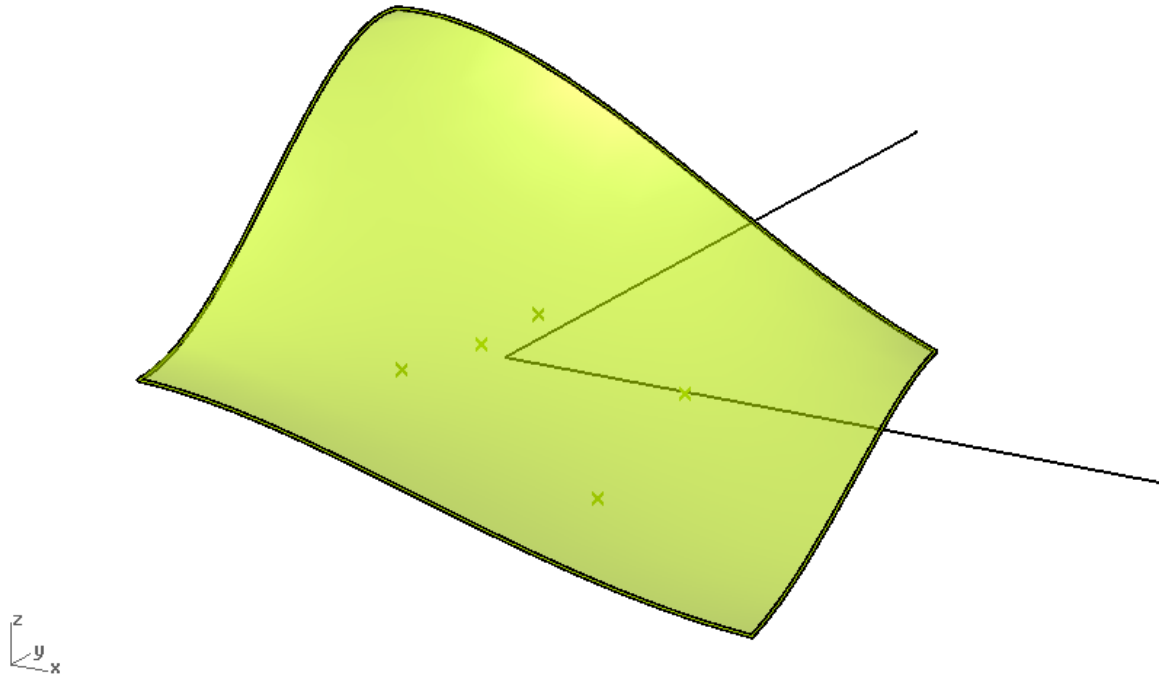
If we don't have a good initial guess available, we employ a two-phase approach: Global optimization brings us close to the optimal solution. Local optimization computes the final result efficiently.

Examples

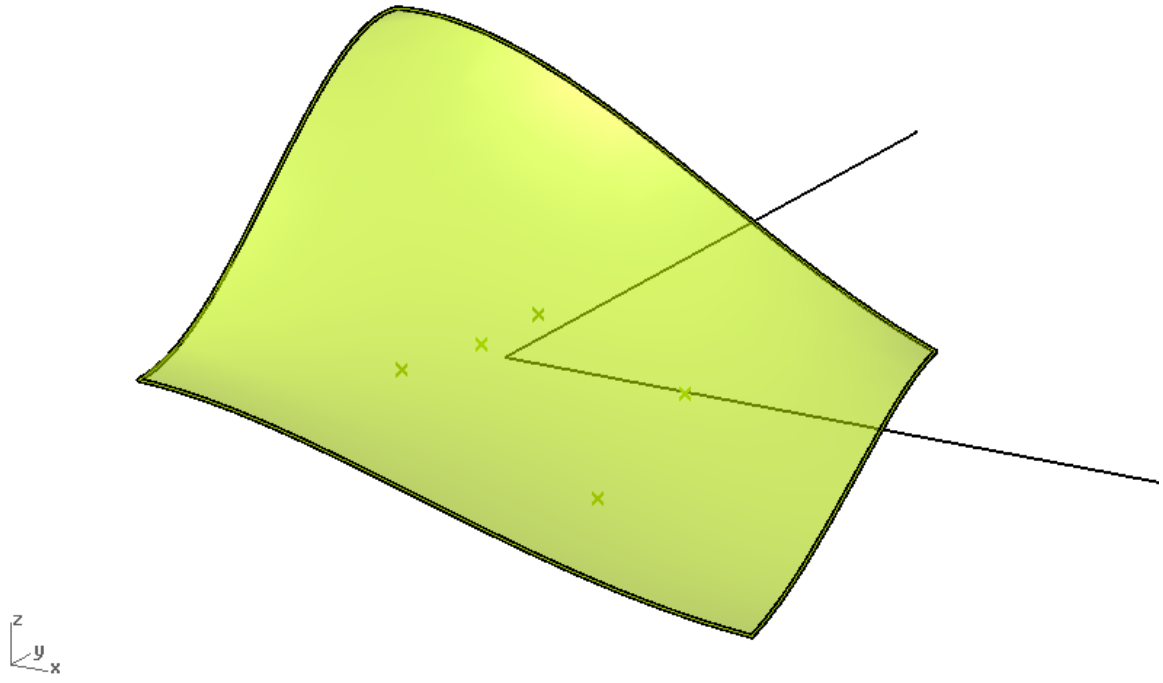
In a handful of examples we make use of optimization principles to generate geometry.



The following examples rely on goat, a free component for local optimization (numerically estimating gradients) and global optimization (among others, an evolutionary solver) for the parametric modelling environment Grasshopper, itself a plugin to the 3D modelling system Rhino. All examples are available for download at http://www.rechenraum.com/goat. For all optimization examples, double-click the goat component and start the optimization with default settings.

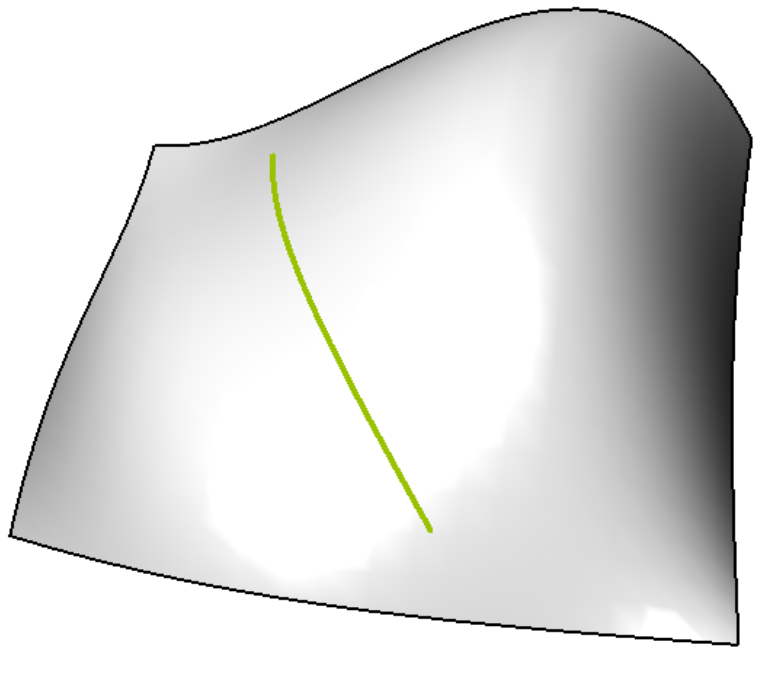


Problem Statement	Given four boundary curves, construct a surface of minimal area.
Constant Input	Four curves defining the boundary conditions.
Variable Input	Four central control points.
Function	Construct surface from input and compute area.
Quality Measure	Area of constructed surface.
Filename	*minimal_surface*

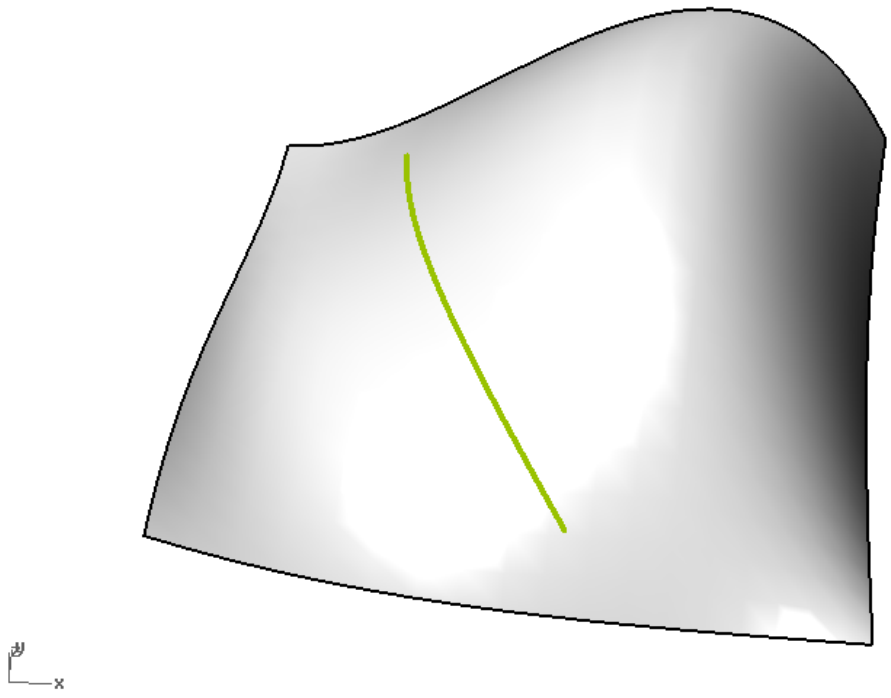


Tasks:

- Unlink one or several variable(s) from the optimization component. Explore the solutions for different configurations of the new constant parameter(s).
- Try different local and global solvers with different settings. Which one performs best?
- Modify the parametric model: remove one boundary curve and add two new control points as new variables.
- What surface will you obtain when the boundary curves are replaced by straight line segments?

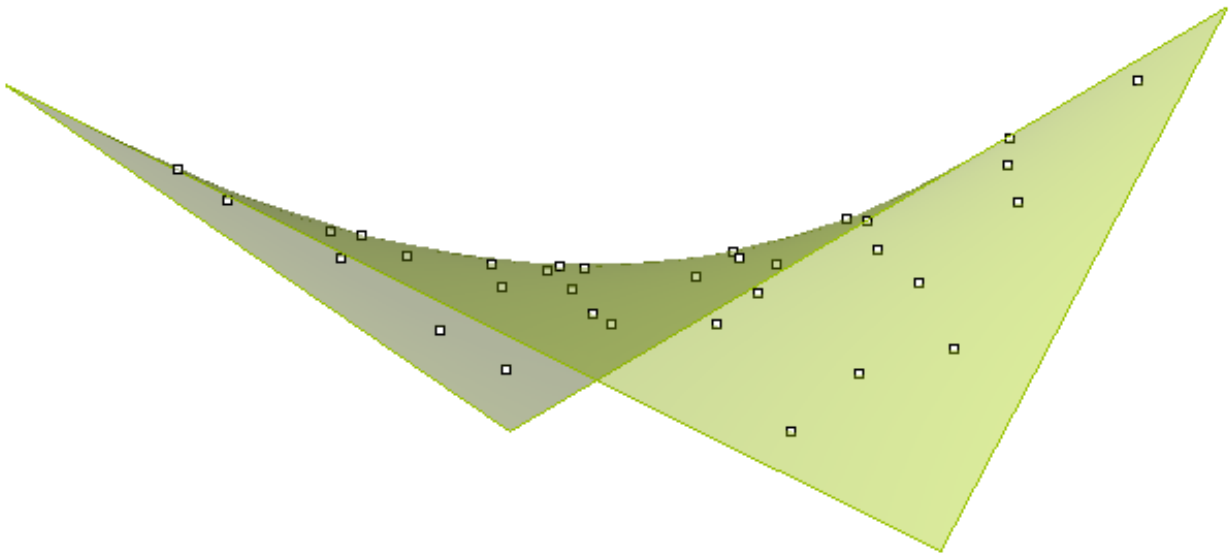


Problem Statement	Compute the shortest path between two fixed points on a surface, a so called *geodesic*. Wooden strips follow geodesics when bent onto a surface.
Constant Input	A surface and the two end points.
Variable Input	Two further interpolation points.
Function	Construct curve on surface and measure its length.
Quality Measure	Length of generated surface curve.
Filename	*geodesic*

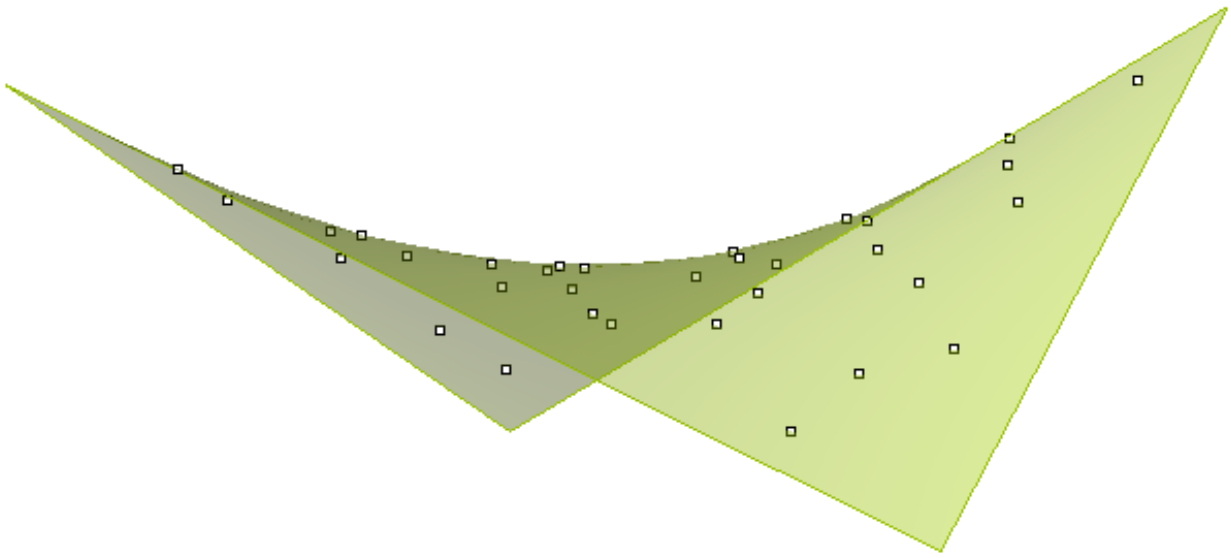


Tasks:

- Modify the shape of the surface and observe, how the path changes.
- Observe how different starting positions for the variable control points yield different solutions on the same surface. Discuss the reasons!
- Employ a two-phase approach combining global and local optimization to obtain the same optimal result for all starting curves.
- Add additional control points to get more accurate results and compare with Grasshopper's *Geodesic* component.

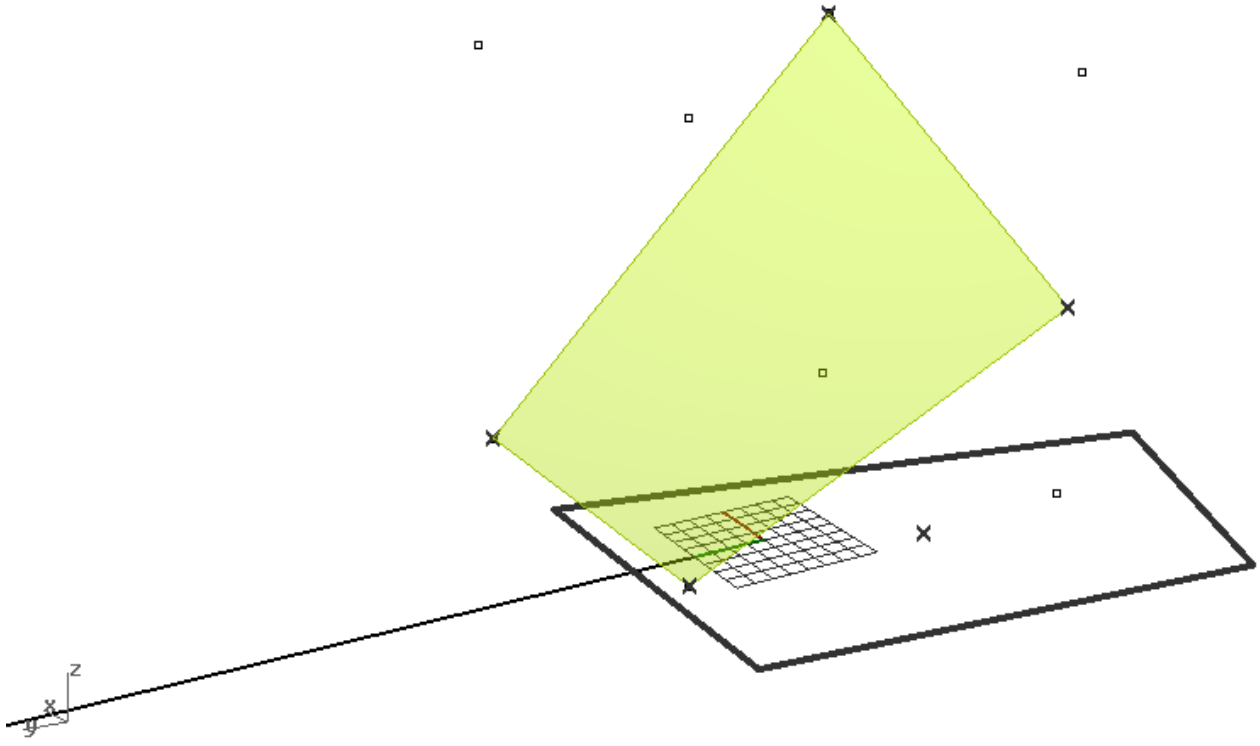


Problem Statement	Find that hyperbolic paraboloid (*HP-surface*, *hypar*) approximating a given set of points best.
Constant Input	A set of data points.
Variable Input	The four corners of the HP-surface.
Function	Construct hyperbolic paraboloid and measure (squared) distance to data points.
Quality Measure	Sum of (squared) distance from data points to surface.
Filename	*hypar_fitting*

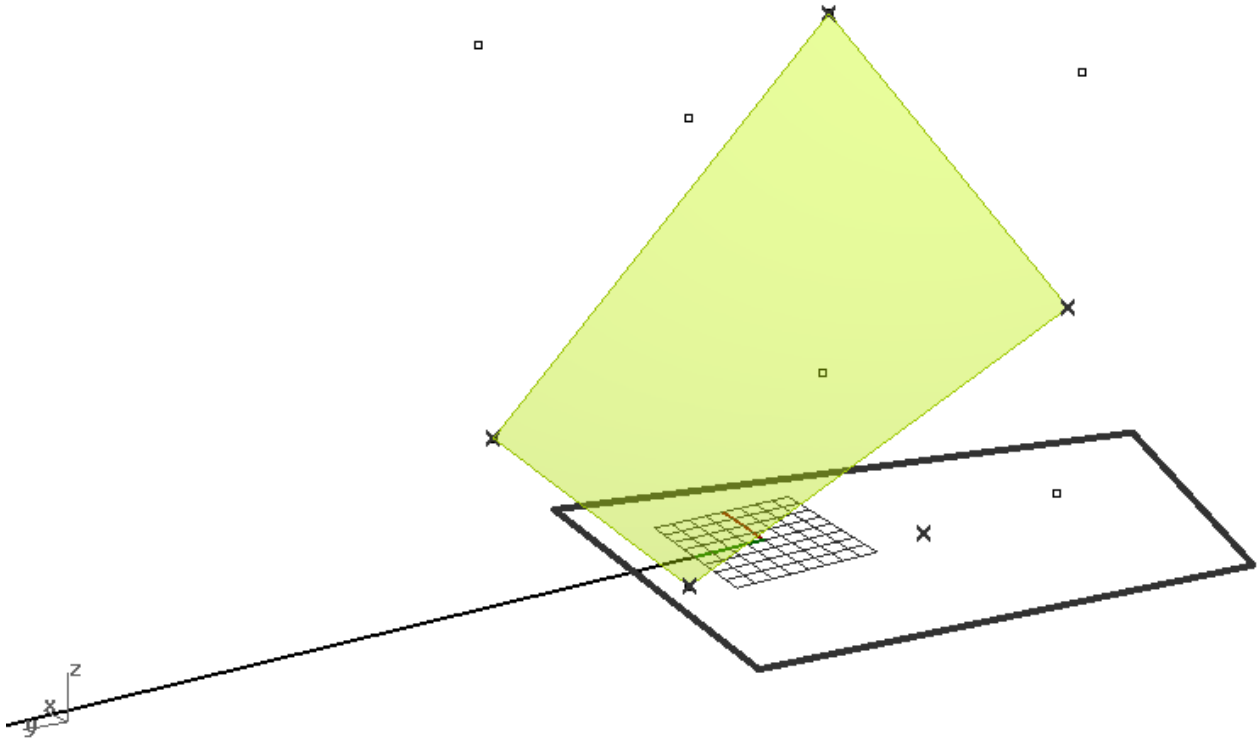


Tasks:

- Bypass the multiplication component to minimize the sum of distances instead of the sum of squared distances. Try different solvers!
- Displace one point significantly along the normal direction of the optimized surface and optimize again. How does this outlier change the result for unsquared and squared distances?
- Fit other surfaces such as cylinders, cones, etc.
- Fix one or several of the variables and observe, how the optimization behaves.



Problem Statement	Find that roof configuration maximizing the shadow for a given sun position.
Constant Input	Direction of sun rays and ground plane.
Variable Input	The four corners of a HP-surface.
Function	Construct hyperbolic paraboloid and its shadow.
Quality Measure	Area of shadow region.
Filename	*roof_shadow*



Tasks:

- Fix one or several corners of the roof and observe, how the result changes.
- Change the direction of the incoming light.
- With a light direction of $l = (1, 0, 1)$ and all roof corners variable, what will the optimized result be like? Verify your assumption.
- Instead of maximizing the shadow, maximize the amount of incoming light (e.g. minimize the shadow).

Appendix

License and Copyright Notices

Version March 26, 2013