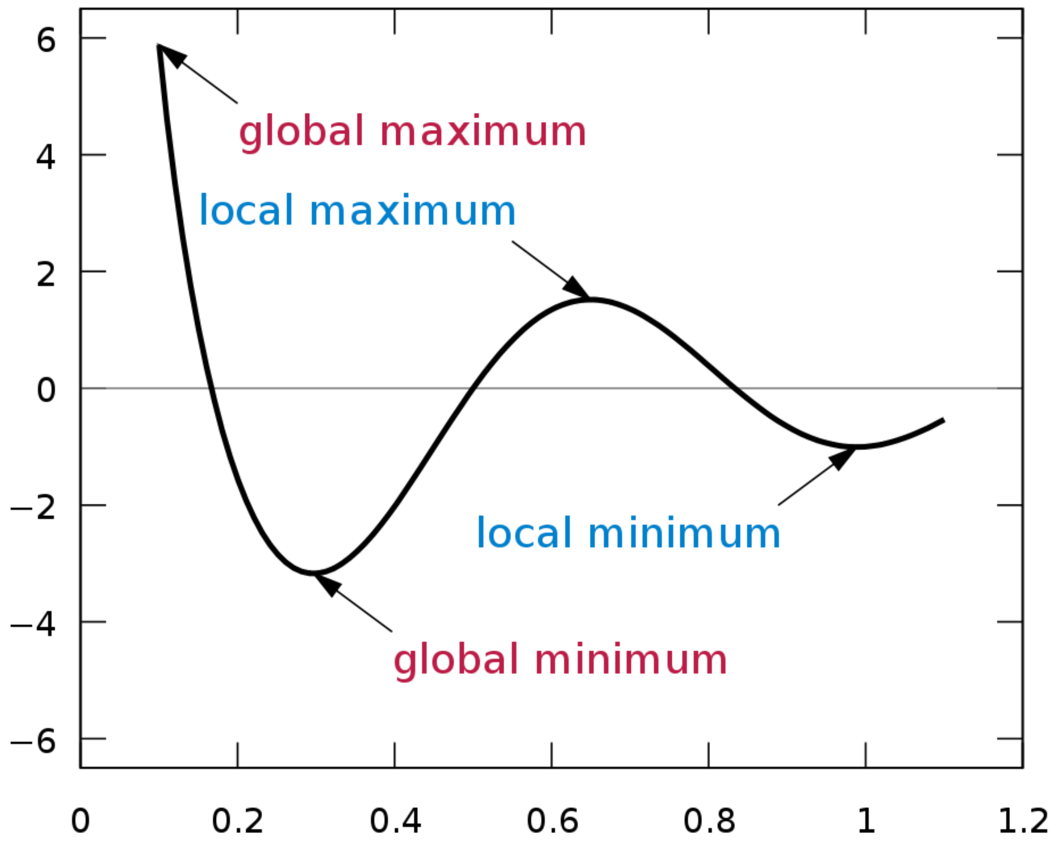
GEO 242: Numerical methods and modeling in the geosciences



Nonlinear inverse modeling

Outline

What do we mean by a ‘linear’ model?

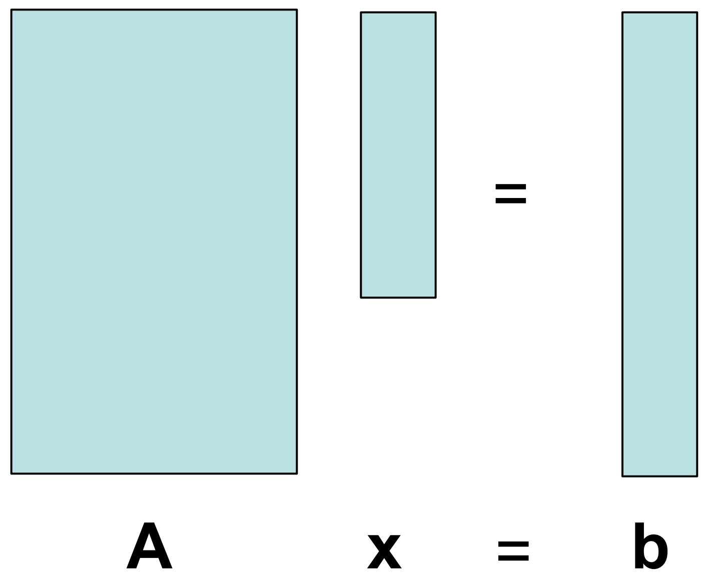
How to deal with a nonlinear model

- The grid search
- Monte Carlo methods
- Optimization (steepest descent, Powell, etc)

What is a linear model?

A linear model does not mean that it is a model that can be expressed as a straight line (at least, not exclusively that...)

Recall the ‘column view’ of a linear system...



b is a linear combination of the columns of **A** weighted by the rows of **x**

What is a linear model?

A linear model responds predictably when it is perturbed... e.g. if you double the model parameters **x**, you double the predicted data (model output) **b**… or vice-versa

The effect of the perturbation is proportional to its cause

Fault slip models are linear. Double the measured displacements and your best-fitting model parameters will be doubled, and vice-versa. Similarly, the curves we were fitting to data last time are linear – double the data and you would double the coefficients used to fit our lines.

Solving a linear model

We have now seen how to solve linear inverse problems – by the method of least squares, implemented by matrix inversion

$$\text{e.g. } \mathbf{x} = (\mathbf{A}^{T}\mathbf{A})^{-1}\ \mathbf{A}^{T}\mathbf{b}$$

These models are quick to run, and have one best answer, that corresponds to the best possible fit to the data in a least squared sense

Nonlinear models

A nonlinear model is a model where a perturbation to the model parameters does not result in a predictable change to the model output

Examples of nonlinear models include models of fault geometry (changing strike, dip, rake, depth, location does not change displacement in a predictable way), earthquake location, numerical weather prediction…

So what’s the problem?

The methods used to solve linear inverse problems cannot be used to solve nonlinear inverse problems. This is a shame, since those methods are efficient and fast.

In order to find an inverse model of a nonlinear system, it is usually necessary to run a lot of forward calculations.

Nonlinear modeling

In general, we aim to find the combination of nonlinear model parameters which best fits the data.

This typically requires the calculation of a misfit statistic, typically referred to as the *penalty function*, a function of the model parameters and the given data.

The modeling process therefore becomes an attempt to find the values of the model parameters that minimize the penalty function, i.e.

- select values of the model parameters
- calculate a forward model of ‘predicted data’
- calculate the penalty function
- modify model parameter values and repeat
- identify the model parameters with the lowest misfit

Parameter space

You can think of nonlinear modeling as a process of evaluating a model’s penalty function over some kind of restricted ‘parameter space’.

Parameter space is fairly easy to visualize in one or two dimensions.

Imagine a graph whose X axis is model parameter values and its Y axis is penalty…

Or a plot where both the X and Y axes are model parameters and the plotted points are color coded or contoured by misfit.

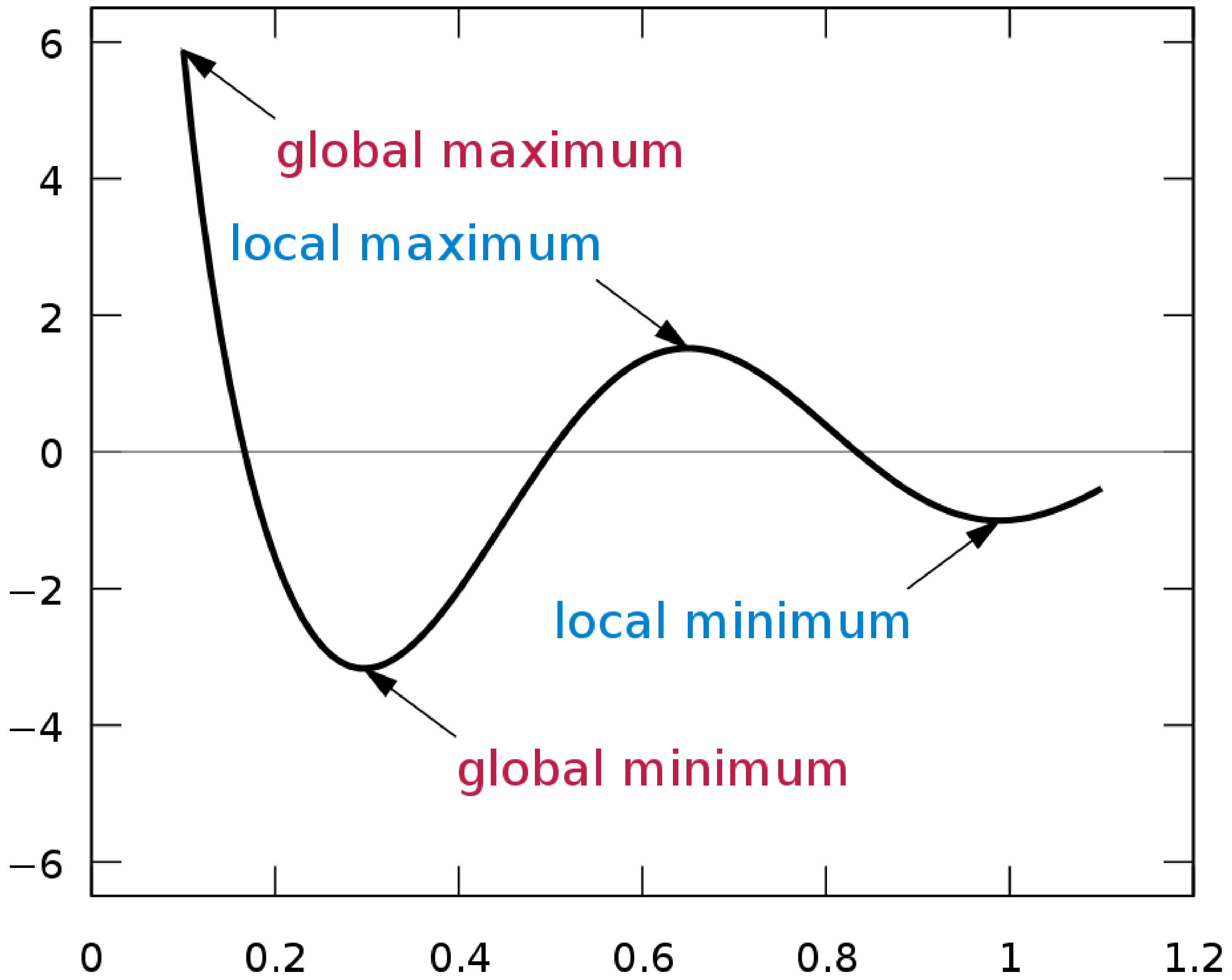
Of course, depending on how many model parameters you have, you might not be able to imagine that many dimensions, plus one…

Global vs local minima

The penalty function need not be ‘simple’ – it can have numerous small peaks (‘local maxima’) and troughs (‘local minima’); we want to find the smallest value overall (the ‘global minimum’), of course!

We need to be sure to sample parameter space carefully in order to be sure to find the global minimum, and not be confused by a local one.

Global vs local minima



Strategies for nonlinear inversion

Given a set of data, a model and a set of model parameters, how best, then, to find the set of model parameters that results in the best fit to the data?

Some common strategies:

1) Grid search – a regular sampling of parameter space

2) Monte Carlo – a random sampling of parameter space

3) ‘Optimization algorithms’ – automated methods for searching parameter space for minimum misfit

4) Markov Chain Monte Carlo – random walk through parameter space

Grid searches

This is a very simple method for finding the minimum of anything numerically – sample parameter space regularly, and find the combinations of parameters that give the minimum

Pros: simple to implement

Cons: you might ‘miss’ the minimum if your sampling is too infrequent; might need several repeat runs with more ‘focusing’ to nail down the minimum; it can waste time sampling areas of misfit space that are unlikely or useless; impractical for more than 3 parameters

Monte Carlo

Use randomly selected values of model parameters (perhaps chosen within defined ranges) to map out the penalty function

Pros: simple to implement, can be targeted (if you select parameter values via a normal distribution, say), recover a ‘posterior’ distribution of most likely models

Cons: can be slow, especially with multiple input parameters

The method of steepest descent

The ‘steepest descent’ method finds the direction in misfit space that has the steepest ‘downhill’ gradient at every step

This is the same idea as water finding its way to the bottom of a valley by flowing down the steepest slope available at every point (i.e. the most efficient path to reduce gravitational potential)

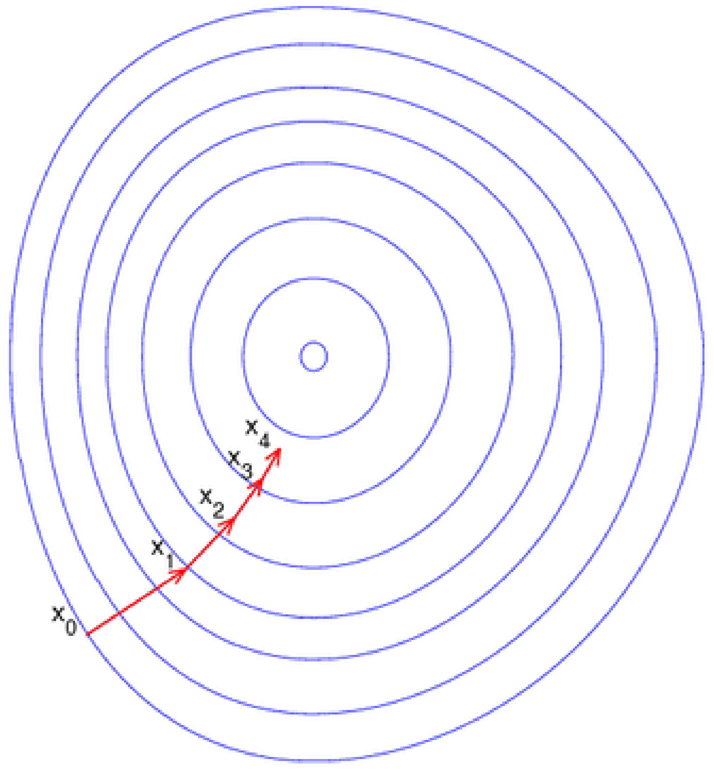
Such methods are often referred to as ‘downhill methods’ or 'gradient descent'

The method of steepest descent

In practice, there will need to be a specified threshold below which gradient will be treated as effectively horizontal

Again, this method is very good at finding minima, but can be 'distracted' by local minima

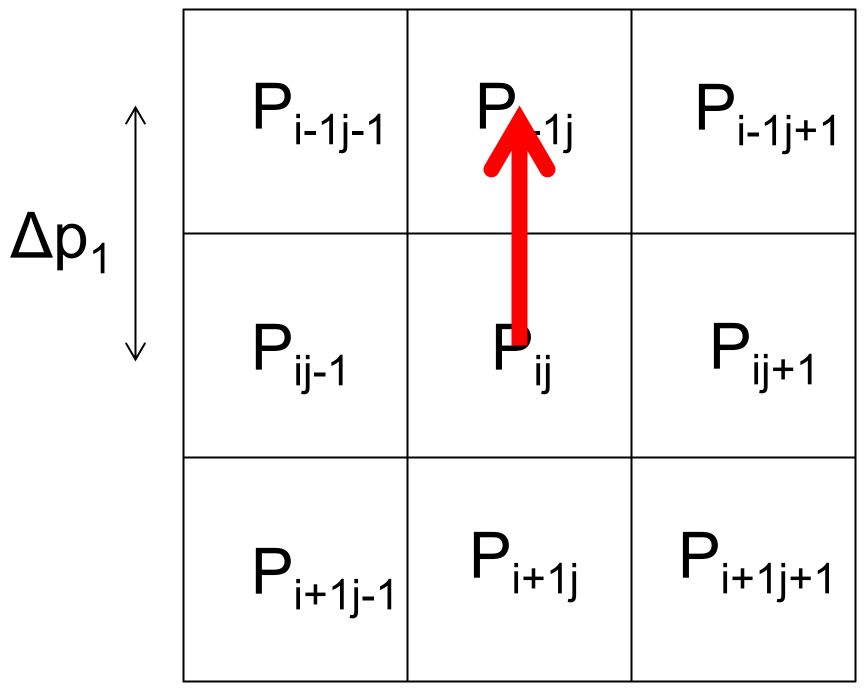
(Solution: Monte Carlo restarts)



Steepest descent in practice

To implement a steepest descent algorithm, you need to be able to estimate local gradients of the misfit function – this can be achieved by finite differencing

You may need to normalize 'distance' between cells

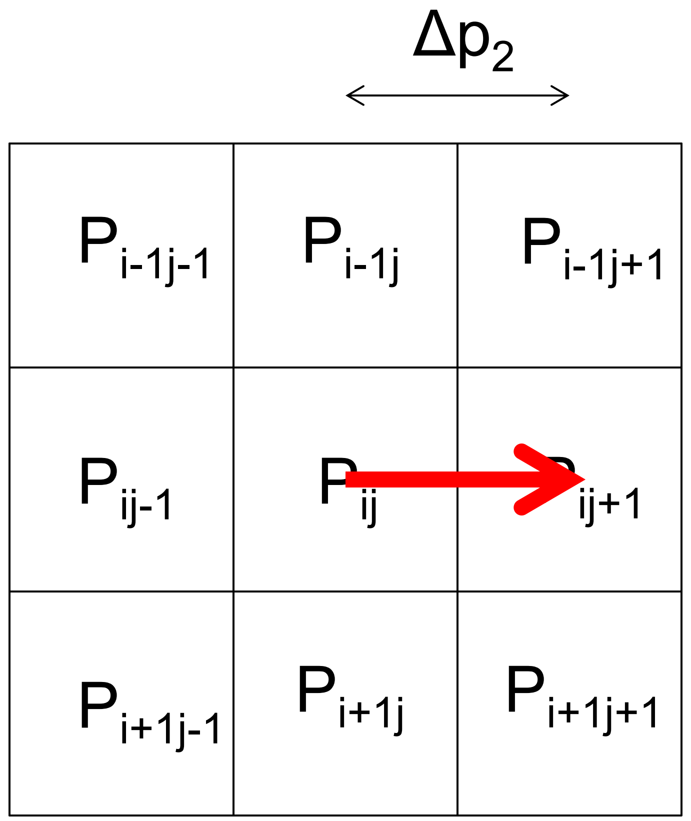


$$\text{grad} = (P_{ij} - P_{i-1j}) / \Delta p_1$$

Steepest descent in practice

To implement a steepest descent algorithm, you need to be able to estimate local gradients of the misfit function – this can be achieved by finite differencing

You may need to normalize 'distance' between cells

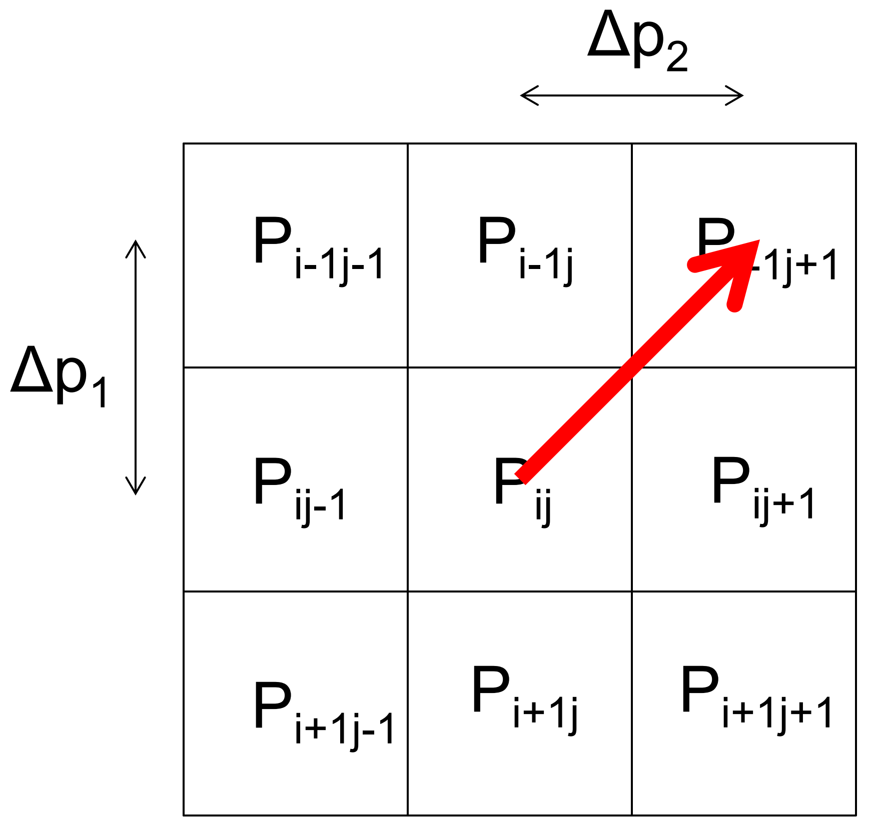


grad = $(P_{ij} - P_{ij+1}) / \Delta p_2$

Steepest descent in practice

To implement a steepest descent algorithm, you need to be able to estimate local gradients of the misfit function – this can be achieved by finite differencing

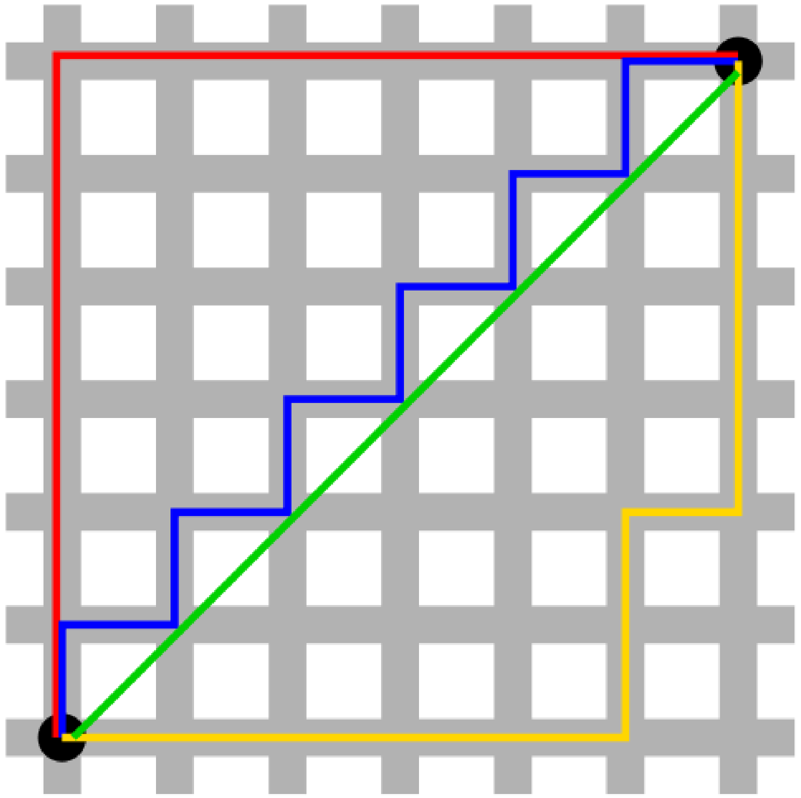
You may need to normalize 'distance' between cells



$$\text{grad} = (P_{ij} - P_{i-1j+1}) / \sqrt{(\Delta p_1^2 + \Delta p_2^2)}$$

The ‘taxi cab’ method

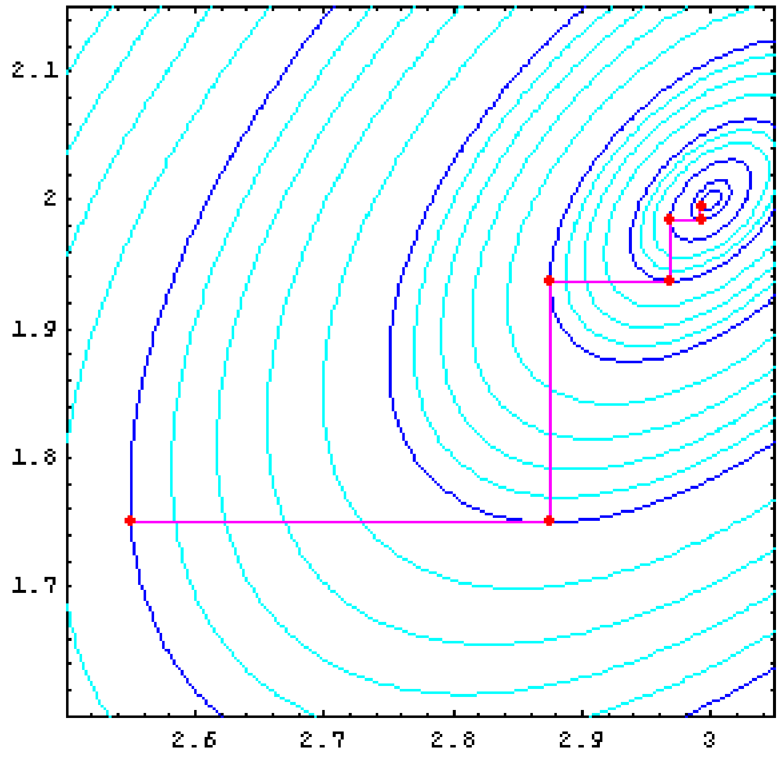
A variation on the steepest descent method uses ‘taxi cab geometry’ – the idea that distances can be represented as a series of steps in each direction, rather than by a simple ‘Euclidean’ straight line



The ‘taxi cab’ method

In the case of optimization, the ‘directions’ we are taking are movements through parameter space

In the taxi cab method, this means we can optimize using each parameter in turn, until a minimum is found

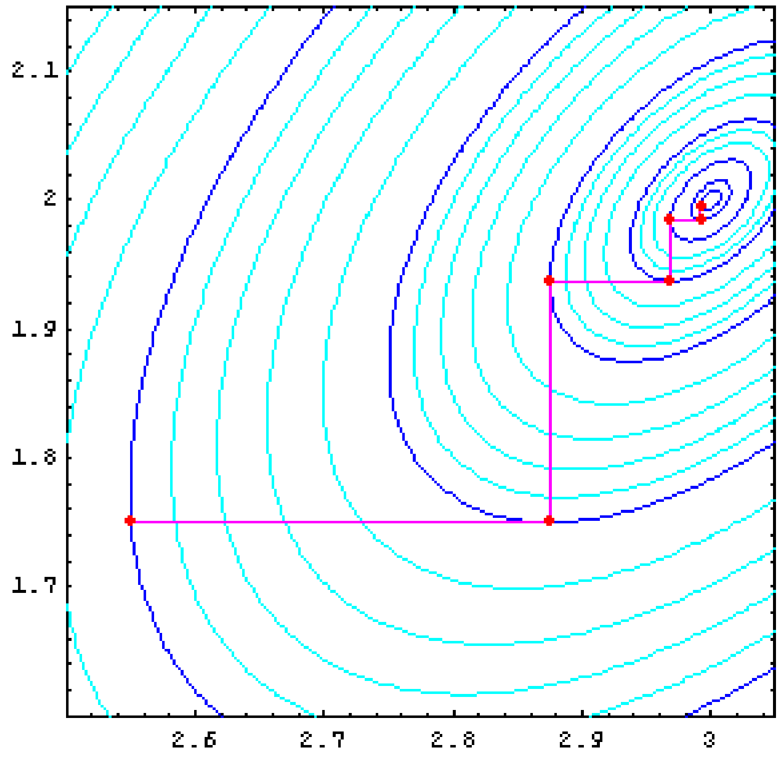


The ‘taxi cab’ method

A ‘taxi cab’ algorithm would hold all but one parameter fixed at a time, and vary one using steepest descent until a ‘flat spot’ is found…

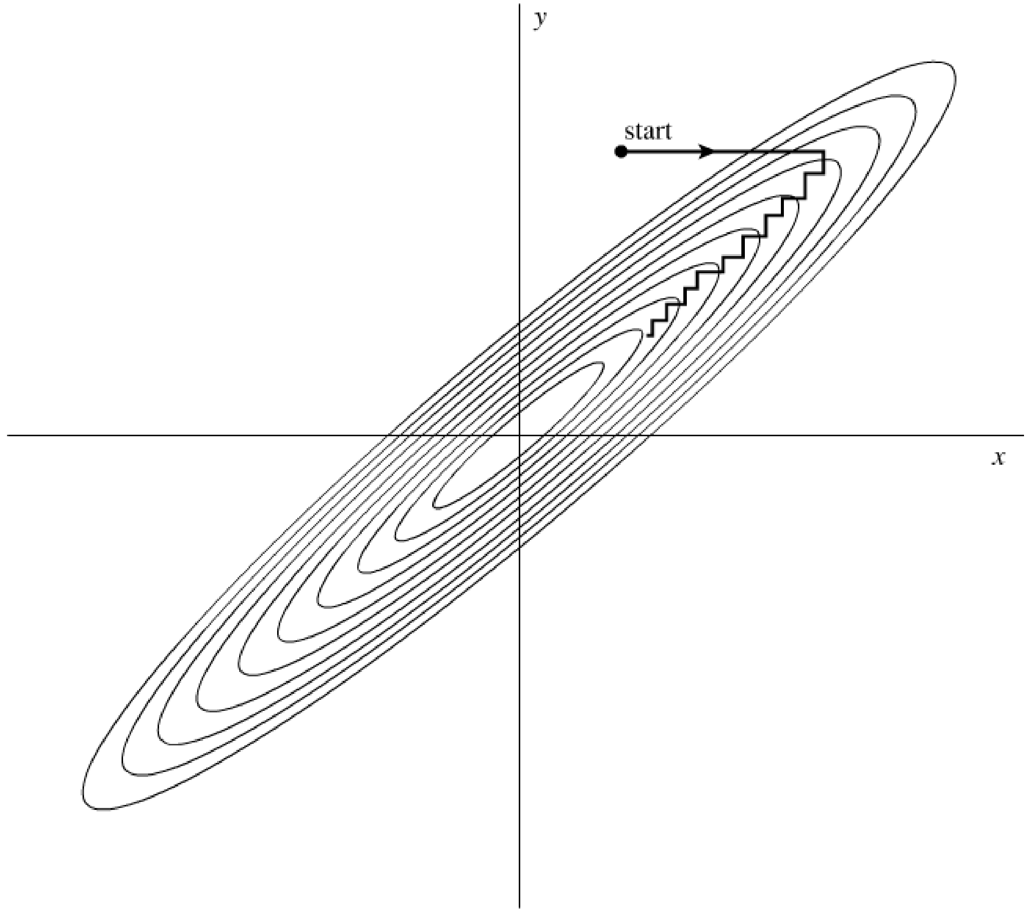
…and then fix that one and vary another until another ‘flat spot’ is found…

…and so on



When the taxi cab is slow…

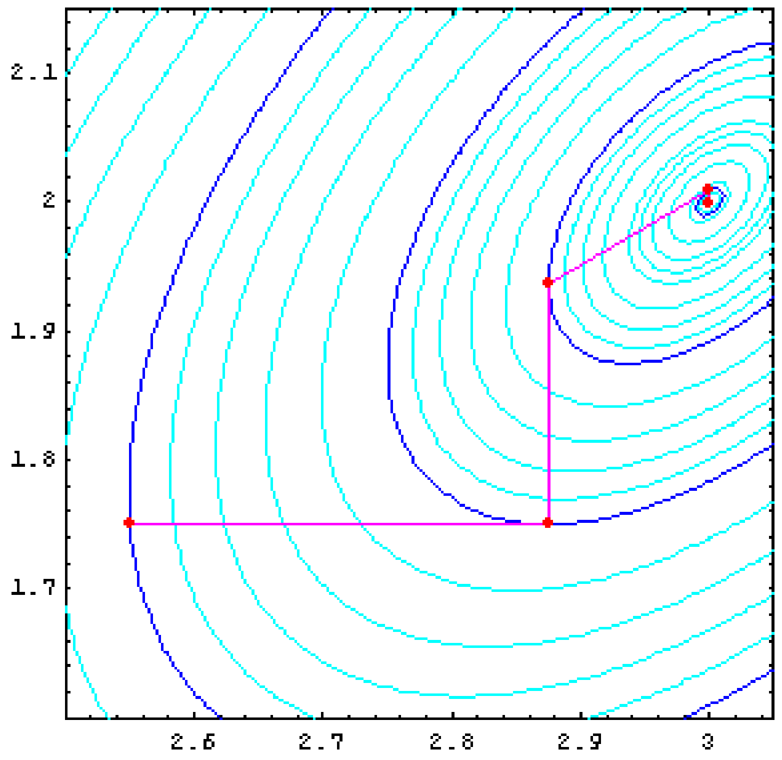
The taxi cab method works pretty well in practice most of the time, but there are definitely situations where it is inefficient…



Powell's method

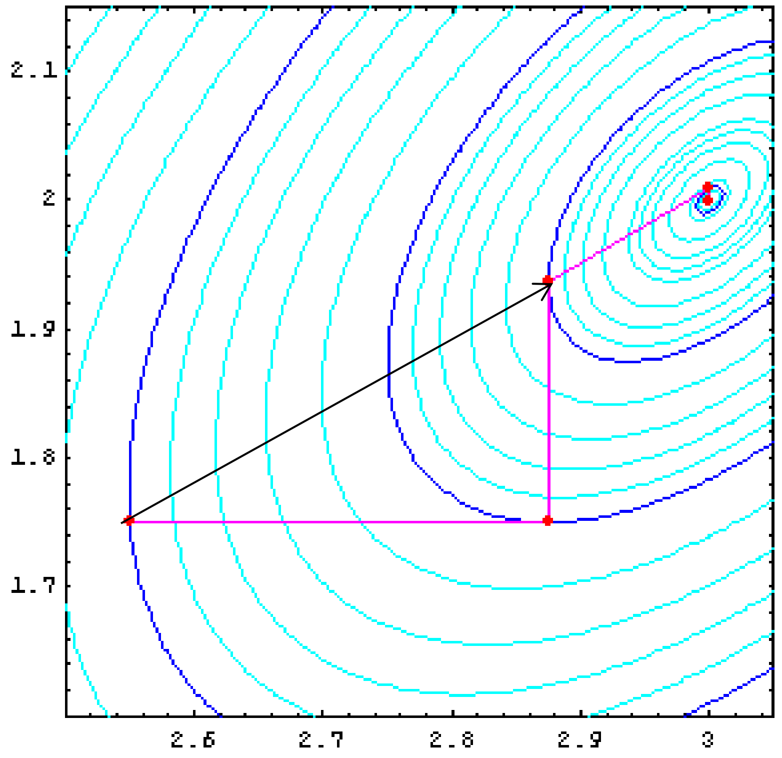
Powell's method is essentially an update of the taxi cab method that tries to estimate more efficient directions to search along

After cycling through all of the different parameters, you should have moved to an improved position in misfit space…



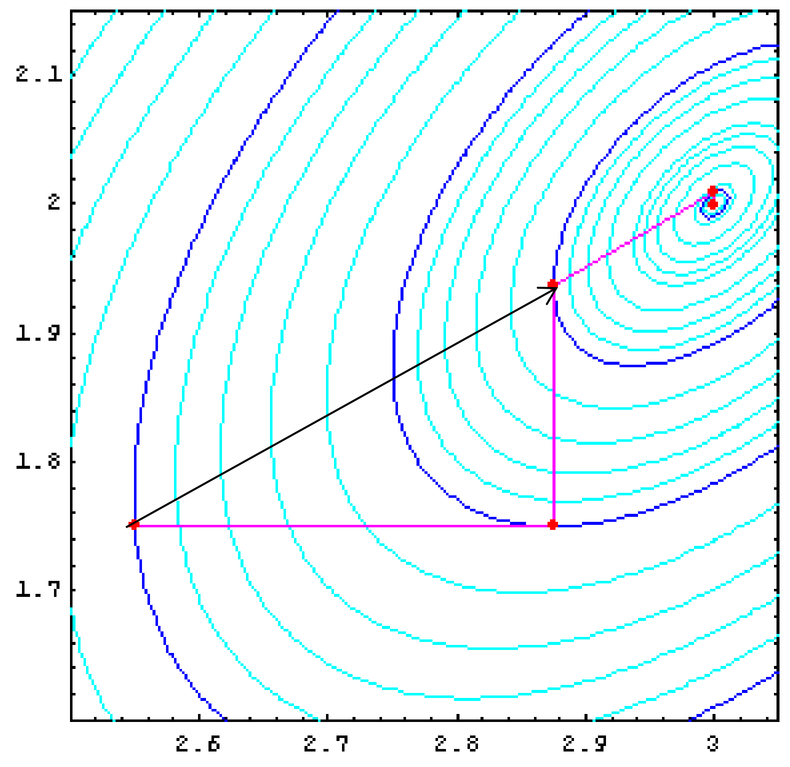
Powell’s method

The vector between the starting point and the end point of the first iteration is used as a new search direction, replacing one of the original search directions



Powell’s method

Powell’s method is referred to as a ‘direction set’ method, as it involves finding an optimal direction for searching parameter space



Optimization for models

So far, we have been looking at finding minima of relatively simple mathematical functions. When we optimize for the best model, the function we are optimizing is the penalty function

The penalty function, is, of course, a function of both the model parameters and the data, and is therefore more unpredictable than a smooth mathematical function (e.g. because of errors in the data)