MANCHESTER 1824

The University of Manchester

Cockcroft Institute Postgraduate Lectures
Numerical Methods and Lattice Design

Lecture 5: Optimisation

Hywel Owen

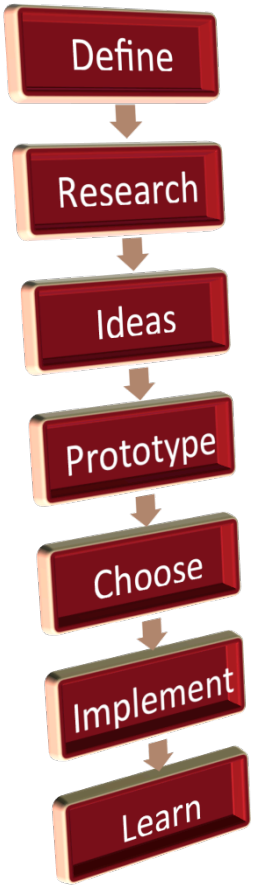
School of Physics and Astronomy, University of Manchester



The Course Syllabus and Projects

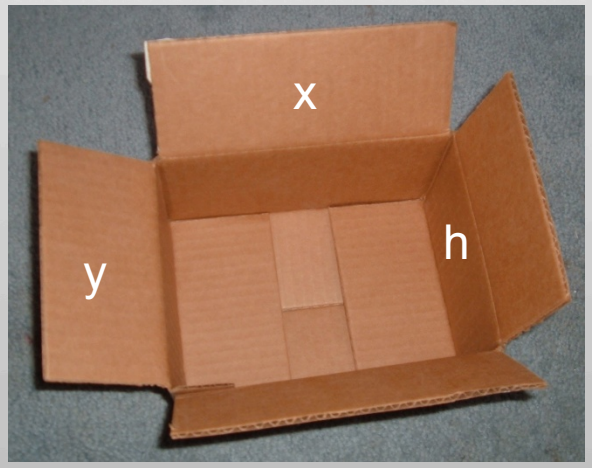
- Recap on programming languages for physics; MATLAB and Python; summary of commands;
- Introduction to numerical computing; errors in computer calculations;
- Numerical integration methods; Euler's method; higher-order methods;
- Precision vs. accuracy; validation;
- Phase space; conserved quantities;
- Introduction to mappings and nonlinear systems;
- *Example: Methods for solving the linear and non-linear simple harmonic oscillator.*
- Introduction to Monte Carlo methods; Monte Carlo integration; classical problems;
- Pseudorandom and quasirandom sampling; methods of sampling; generation of distributions;
- Particle transport simulation; nuclear cross sections; particle histories; applications of Monte Carlo transport;
- *Example: Simulation of penetration of neutrons through shielding.*
- From mappings to linear optics; the concept of lattices;
- Transfer matrices and periodic solutions; propagation of linear optics parameters;
- Classic optical systems: the FODO, the double-bend achromat;
- Matching and optimisation; penalty/objective functions;
- Hill-climbing methods: Cauchy's method, Nelder-Mead, simulated annealing;
- Variables and constraints; under- and over-constrained problems;
- *Example: MAD8 matching of FODO Twiss values;*
- Multiple-configuration methods; genetic algorithms and evolutionary algorithms;
- A bestiary of codes; choosing the right code;
- Common pitfalls;
- *Example: Particle tracking in MAD8;*

What is Design?



- Design is the process of creating something to fit a purpose - from toothbrushes to accelerators.
- A design is judged to be good by *quantifying* how good it is compared to other designs.
- The space of possible designs is termed the *Configuration Space*. The 'goodness' of the design is termed the *Objective Function*.
- *Optimisation* is the improving of a design. This means either maximising or minimising the Objective Function *F*.
- There is a strong link between optimisation, linear/nonlinear programming, and more 'mundane' activities like curve fitting; they are mathematically similar.

What is Configuration Space?: A Simple Example



A cardboard box

- Question:
  - What is the largest volume that can be enclosed by a given surface area of cardboard?
- Motivation:
  - We would like to minimise the amount of cardboard used!

- Of course, in this simple example we know the answer:

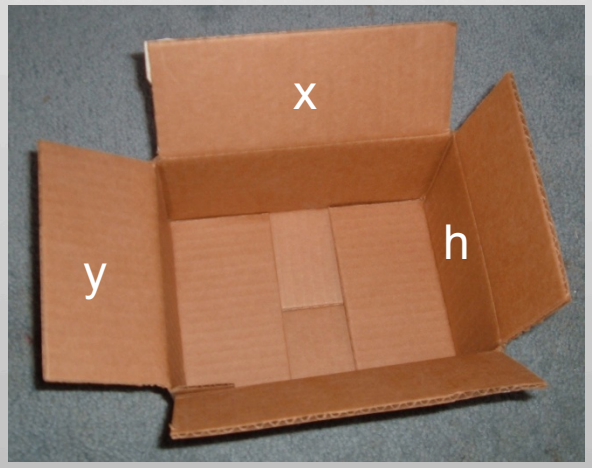
$$x = y = h$$

$$A = 2(x^2 + x^2 + x^2) = 6x^2$$

$$V = x^3$$

$$V = (\tfrac{A}{6})^{3/2}$$

Configuration Space: Varying the Independent Parameters



Our cardboard box

- For any dimensions we have

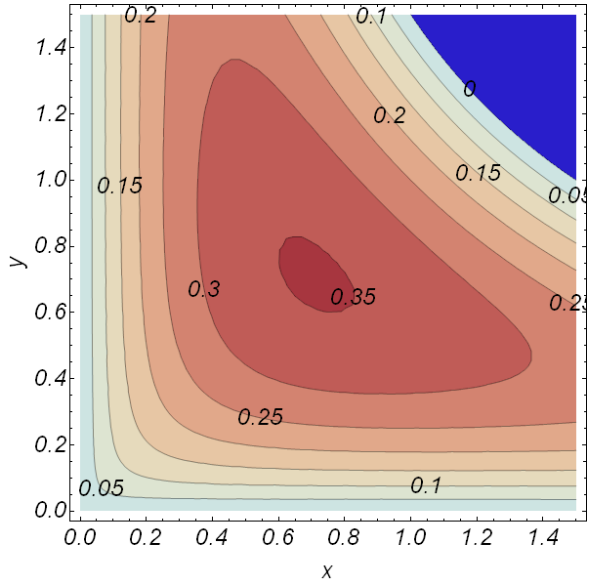
$$A = 2(xy + xh + yh)$$

$$V = xyh$$

- Eliminating dependent variable h we have

$$V = \frac{xy(A-2xy)}{2(x+y)}$$

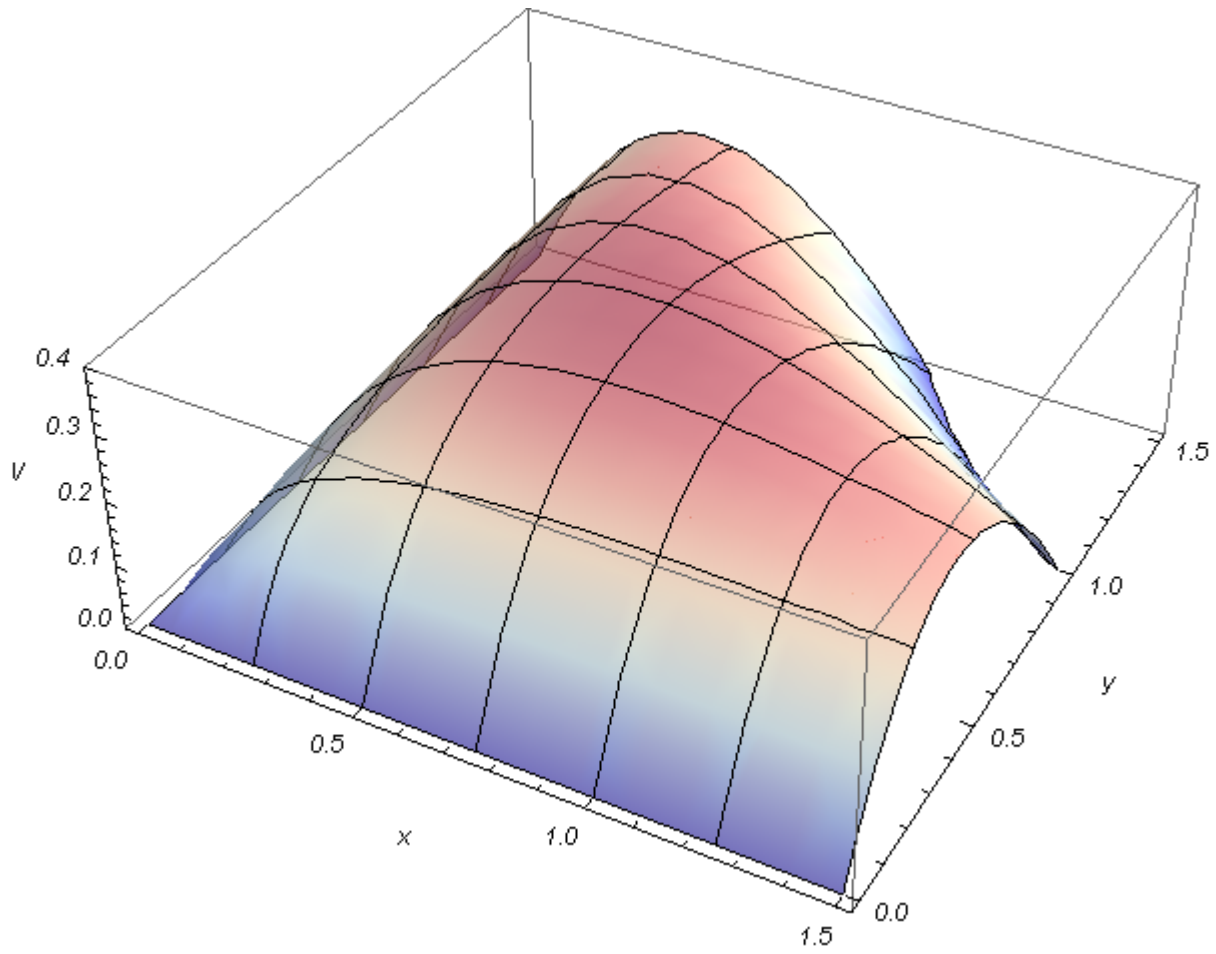
- Here, V is the *Objective Function*
- Example: $A = 3$



MANCHESTER 1824

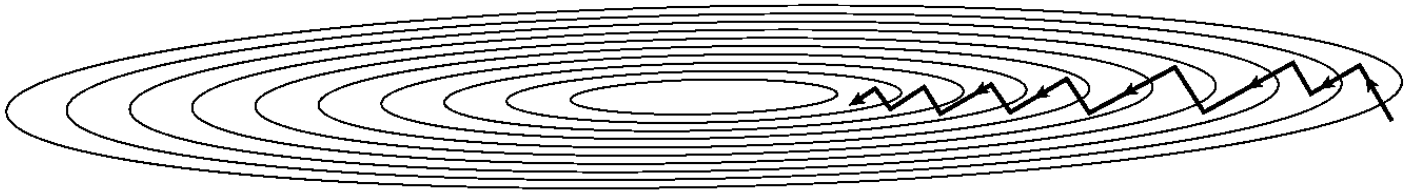
The University of Manchester

Variation of Objective Function over the Configuration Space

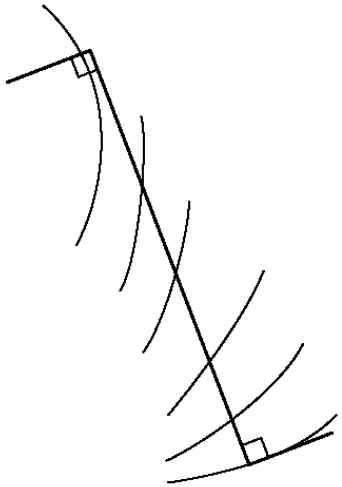


Method of Steepest Descent (Cauchy)

- Requires that the local gradient of the objective function *F* can be calculated in some way
- Choose point $\underline{P}_0$
- Move from $\underline{P}_i$ to $\underline{P}_{i+1}$ by minimising along the direction $-\nabla F$



(a)

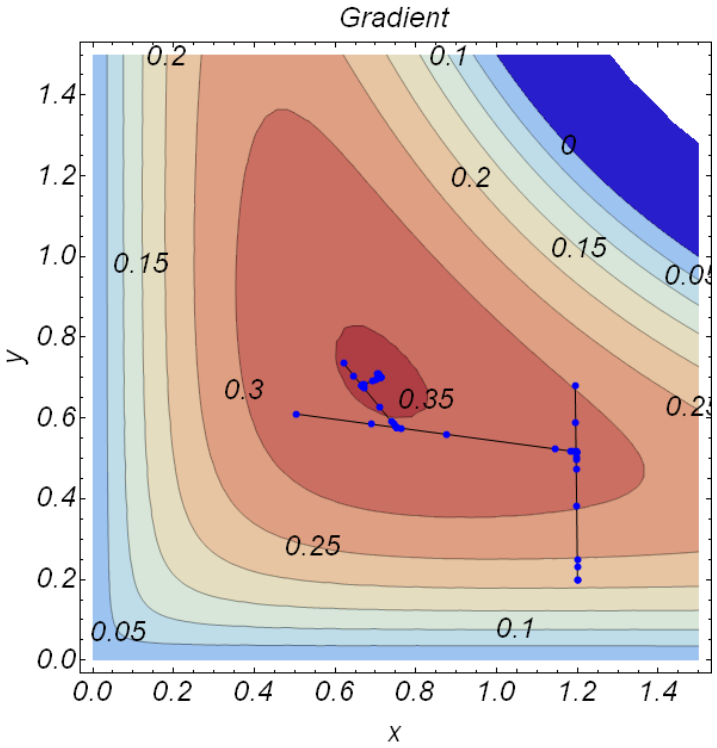


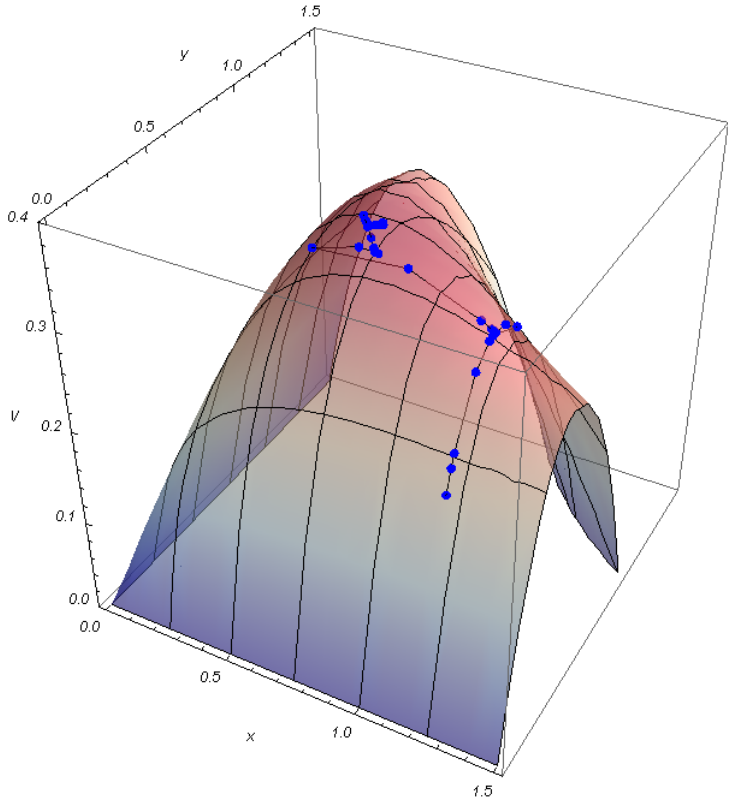
(b)

MANCHESTER 1824

The University of Manchester

Optimising the box problem numerically (Cauchy's method)



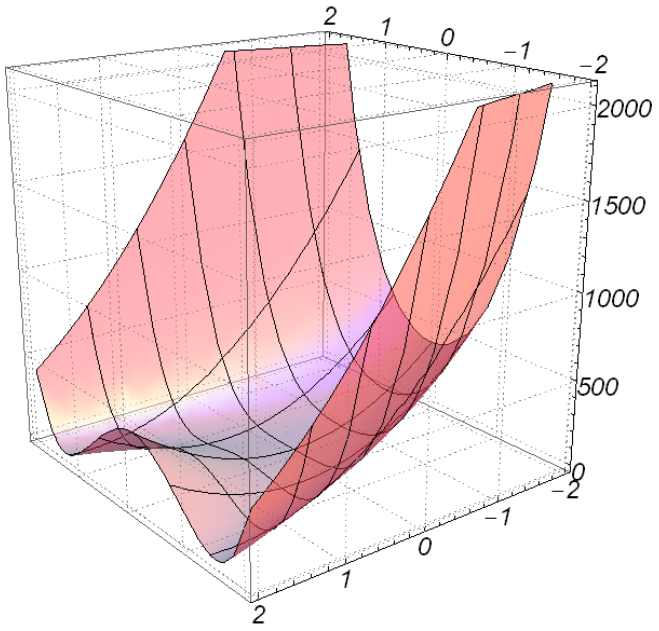


Note that you must implicitly define a tolerance for how close you are to the 'top'

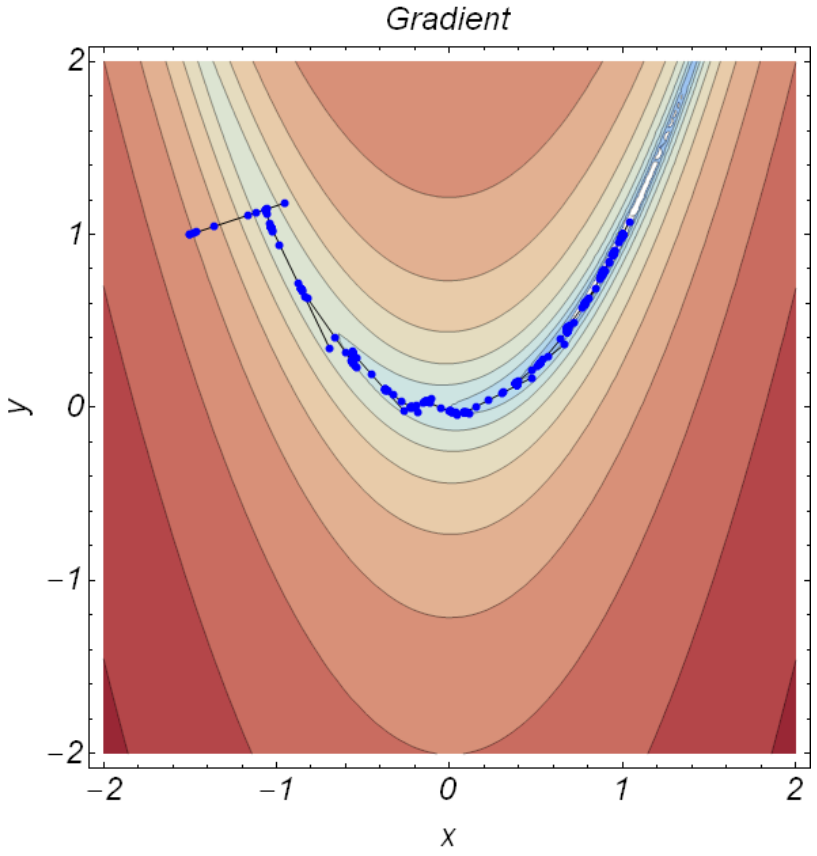
When the method of steepest descent has problems: Rosenbrock's function

Rosenbrock's function defines a curved narrow valley with a shallow-sloped bottom:

$$f(x,y) = (1-x)^2 + 100(y-x^2)^2$$



The method of steepest descent can take many steps….

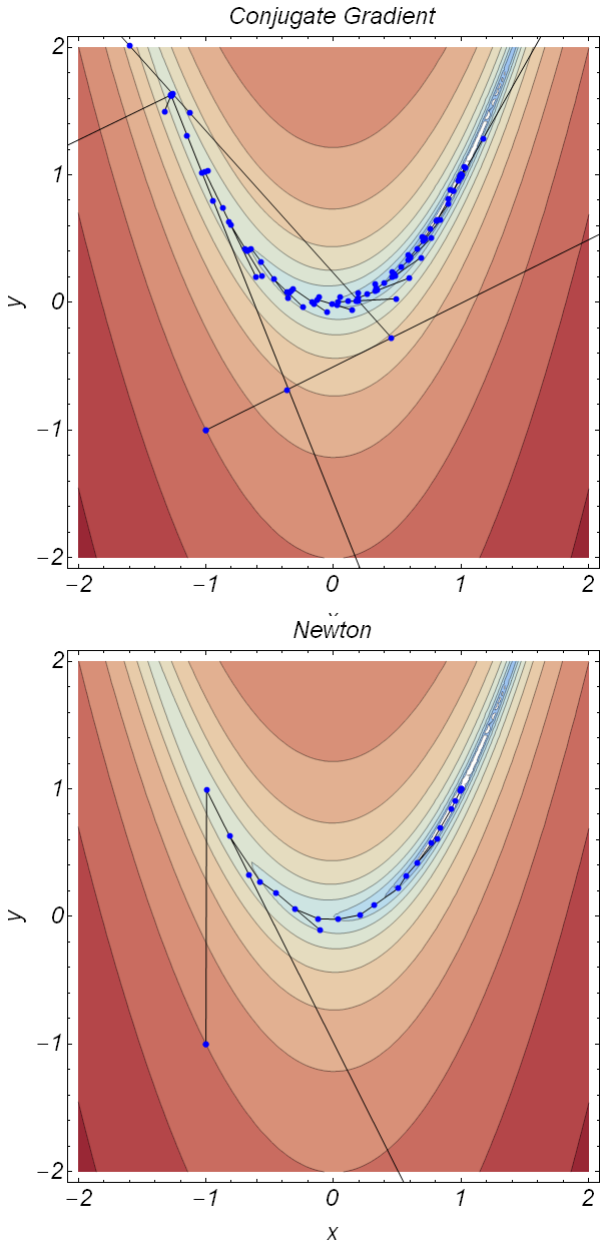


Variations of Hill-Climbing Strategies

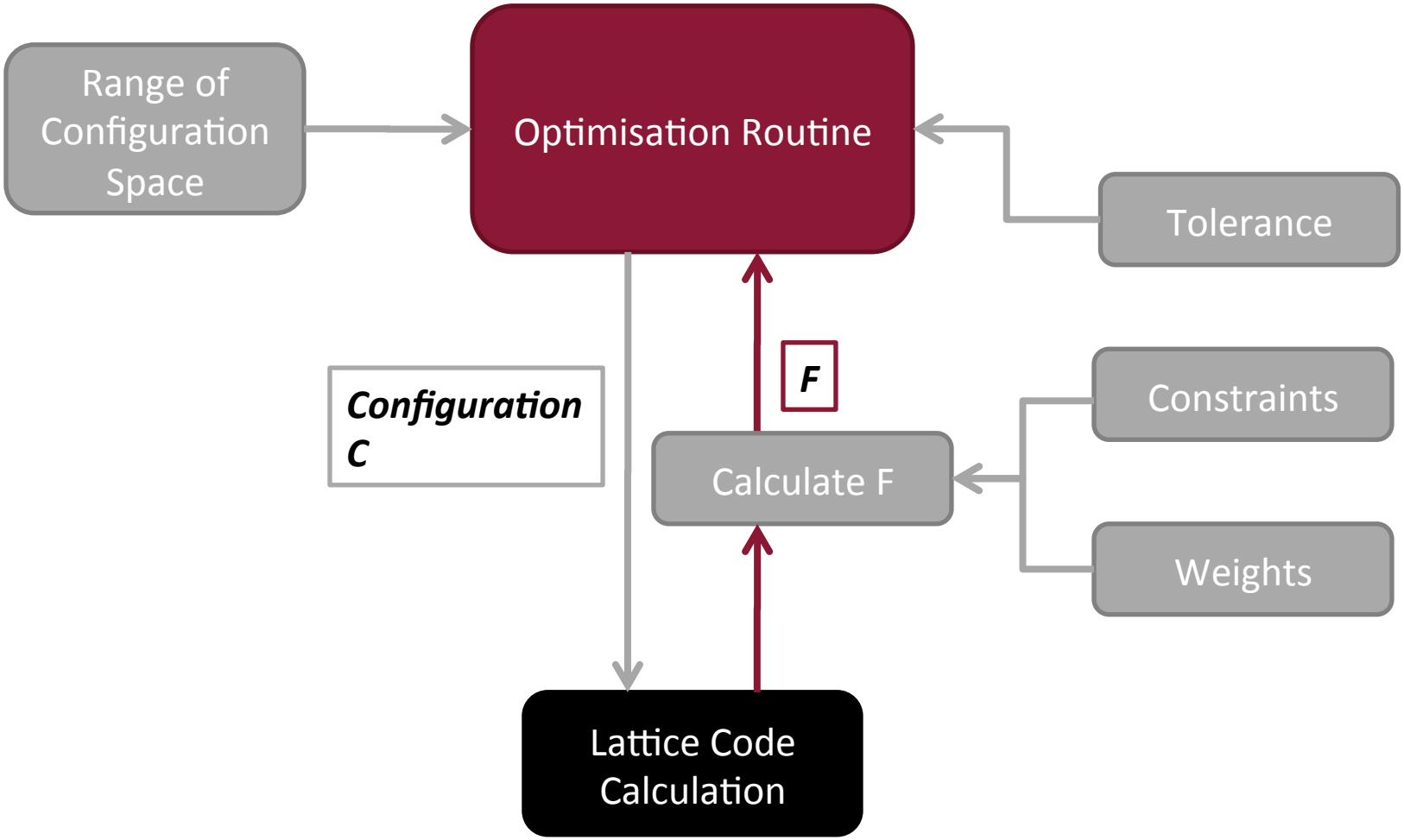
- There are variations on a theme, but they all share the same features:

1. Have to choose an initial start point
2. Need to calculate derivative $-\nabla F$

- Calculating a derivative can be done with ‘functions, but what about general codes?



General Structure of an Optimisation Routine (e.g. in a Lattice Code)



Example – MAD Matching Module

- Objective Function is called Penalty Function, which is minimised. Weighting is accomplished by multiplying the constraint by the weight in the penalty function calculation.
- Three methods used - LMDIF, MIGRAD, and SIMPLEX. MIGRAD and LMDIF calculate numerical derivatives of either the penalty function as a whole or of eac es the Simplex algorithm.

12.6 Matching Examples

12.6.1 Simple Periodic Beam Line

Match a simple cell with given phase advances:

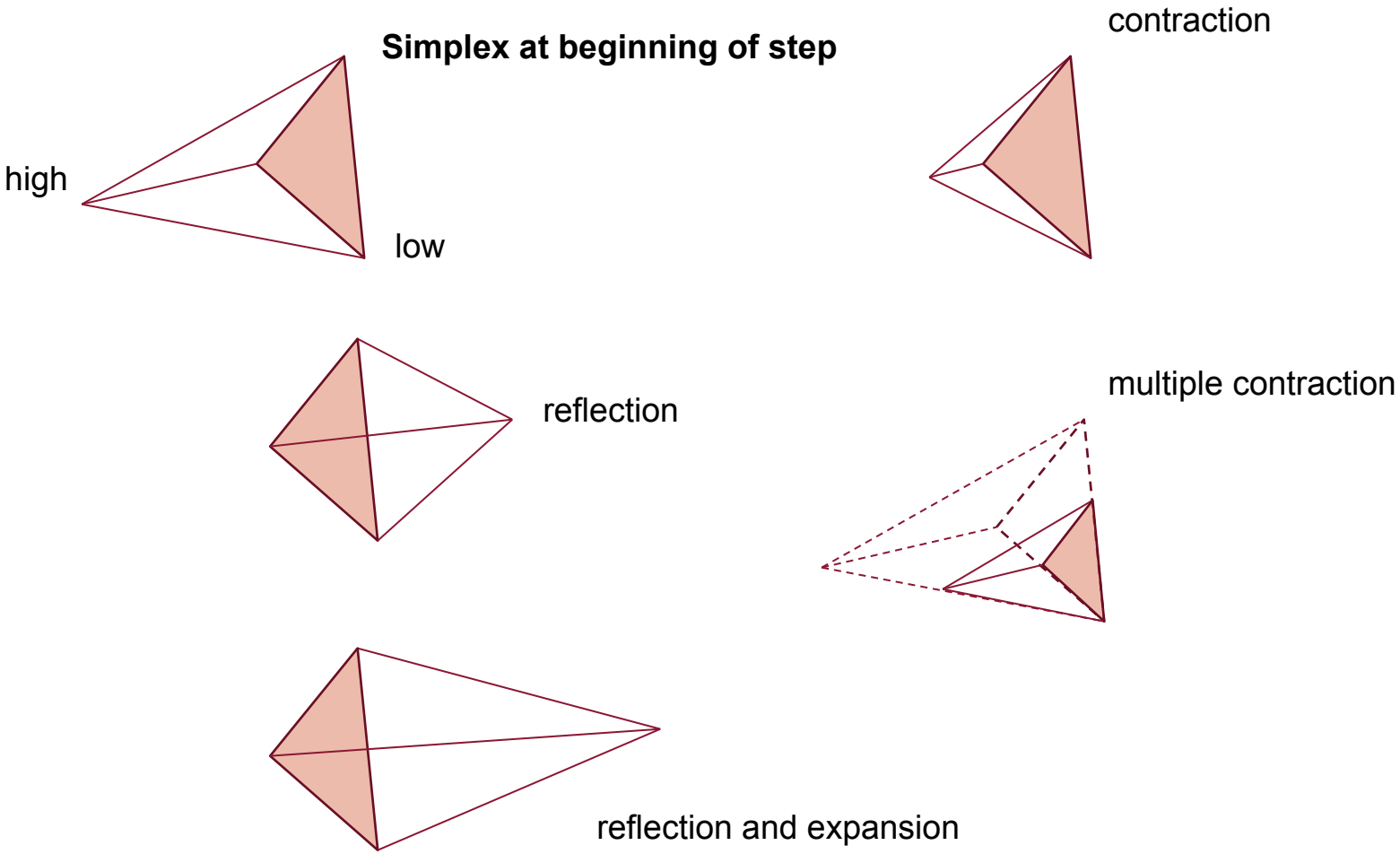
```
QF:     QUADRUPOLE,...
QD:     QUADRUPOLE,...
CELL1:  LINE=(...,QF,...,QD,...)
        USE,CELL1
        CELL
        VARY,NAME=QD[K1],STEP=0.01
        VARY,NAME=QF[K1],STEP=0.01
        CONSTRAINT,PLACE=#E,MUX=0.25,MUY=1/6
        MIGRAD,CALLS=2000
        ENDMATCH
```

The Downhill Simplex Method (Nelder & Mead, 1965)

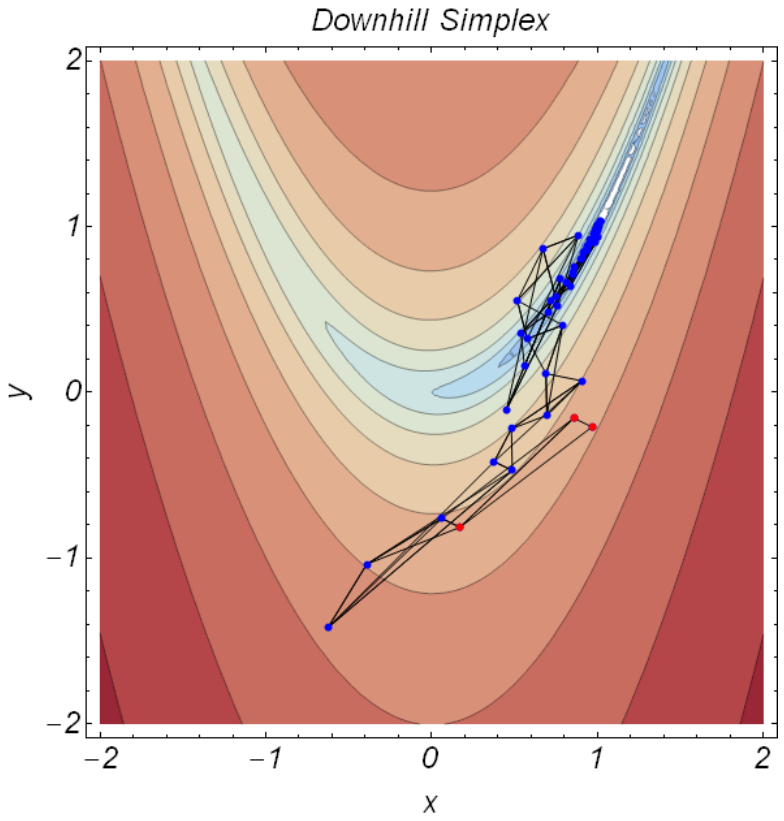
- A way of getting round the derivative problem – use multiple starting points.

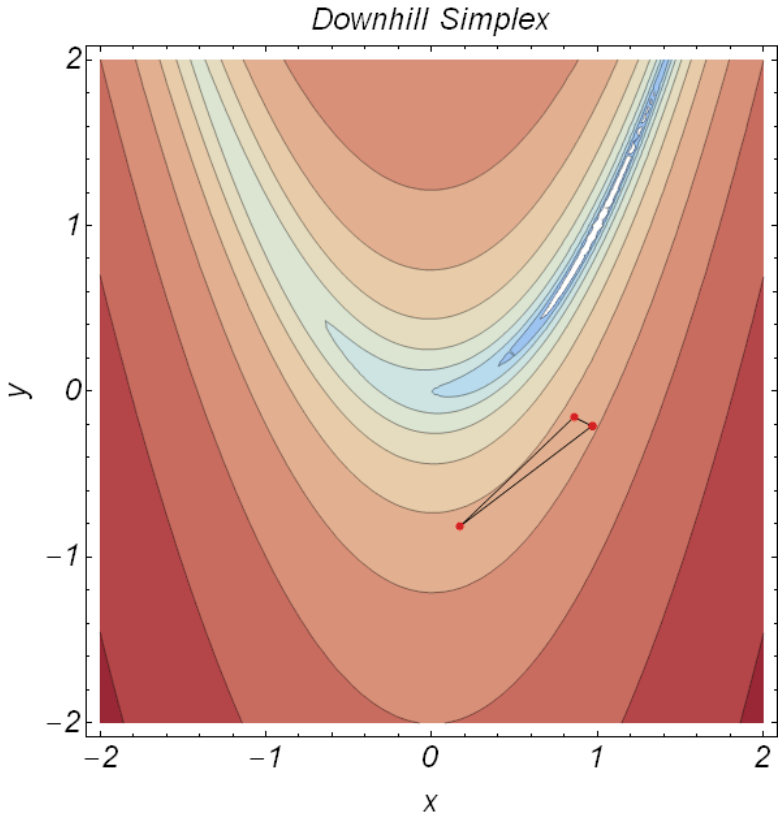
- Simplex - geometrical figure in *n* dimensions, with *n+1* vertices.
  - Triangle in 2 dimensions, tetrahedron in 3 dimensions…
- Choose starting point $\underline{P}_0$, and create simplex by adding each of the unit vectors $\underline{e}_i$ for each vertex.
- Evaluate *F* for each vertex. Choose new simplex.

Defining and moving the simplex



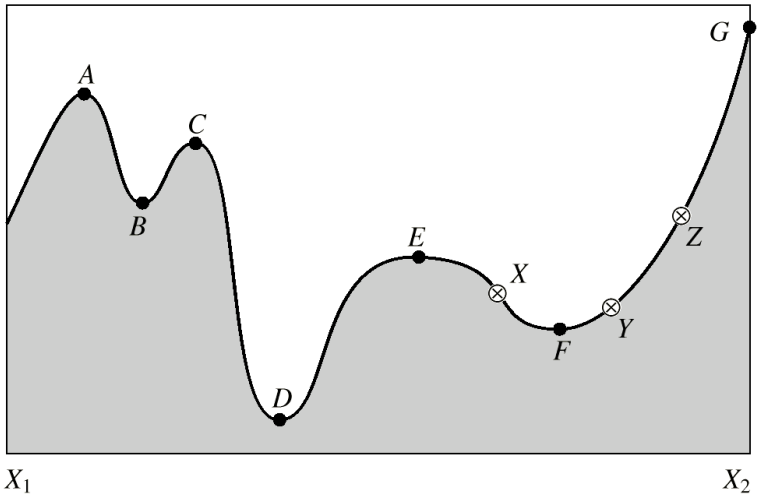
Downhill Simplex on Rosenbrock's Function





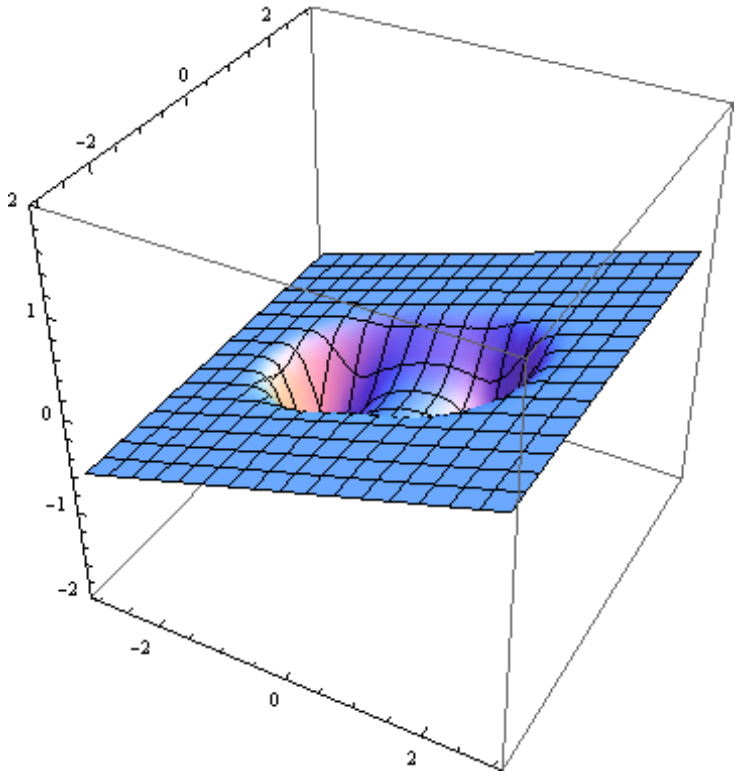
Hill-Climbing

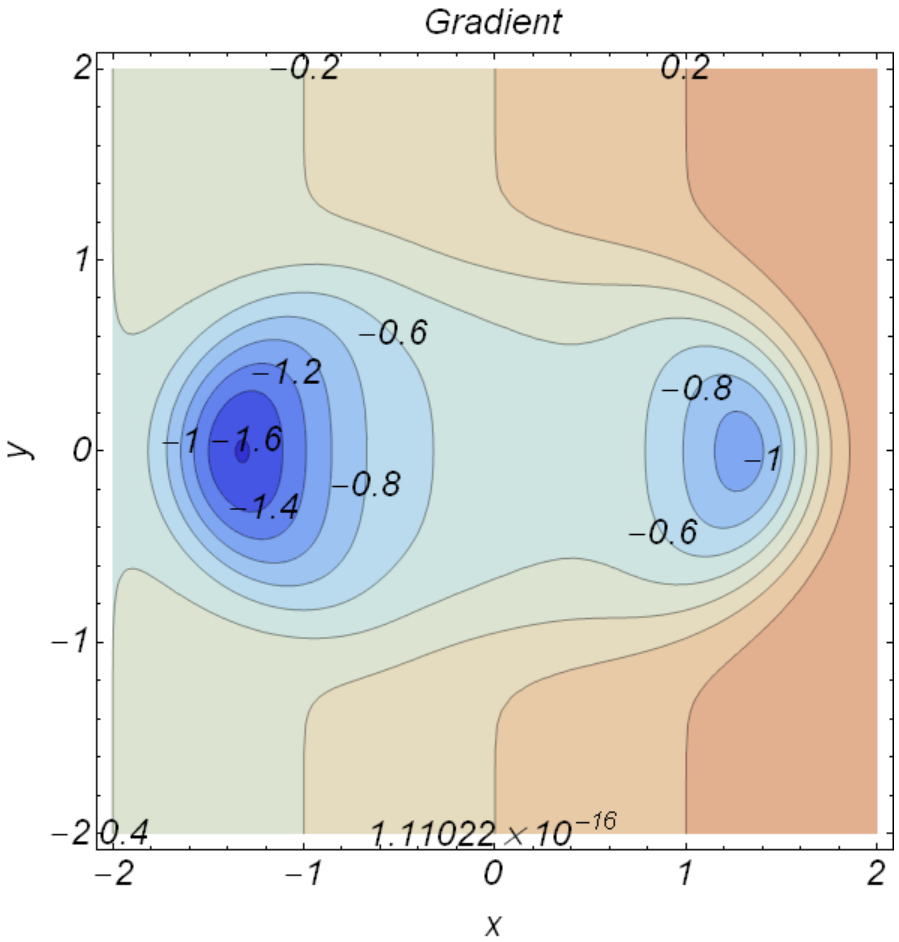
- All of the previous methods are *Hill-Climbing* strategies. Once you're on the top of the *nearest* hill, you can't get any higher.
- How do you find the highest point?



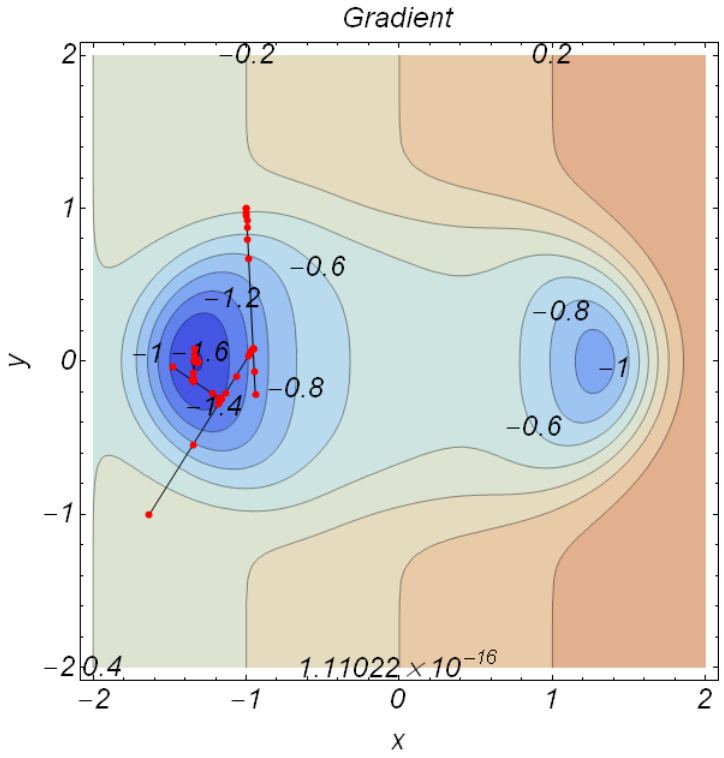
- (hint: this is also an example of a *greedy* strategy)

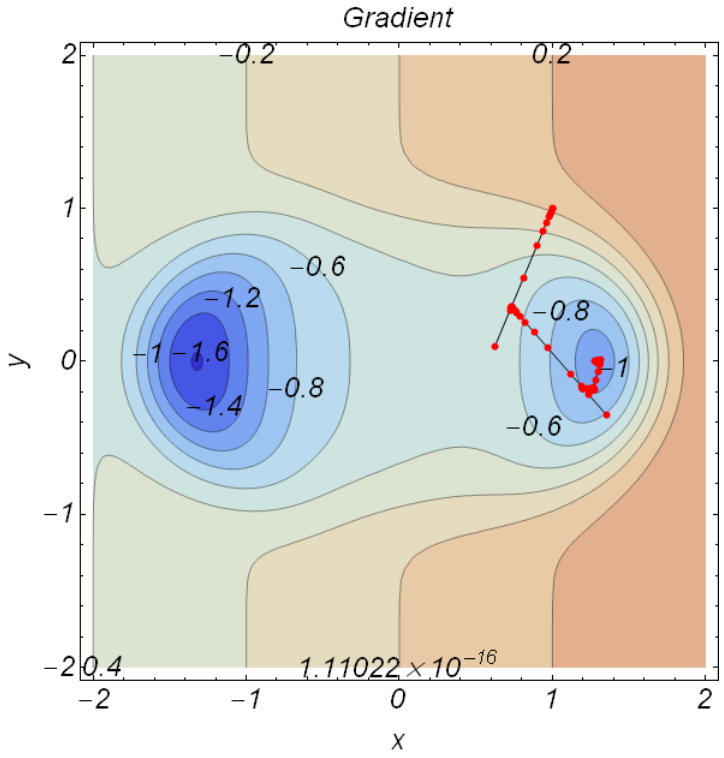
Multiple-Minima Systems Example: Sloped Double Gaussian





The end point depends on the chosen starting point

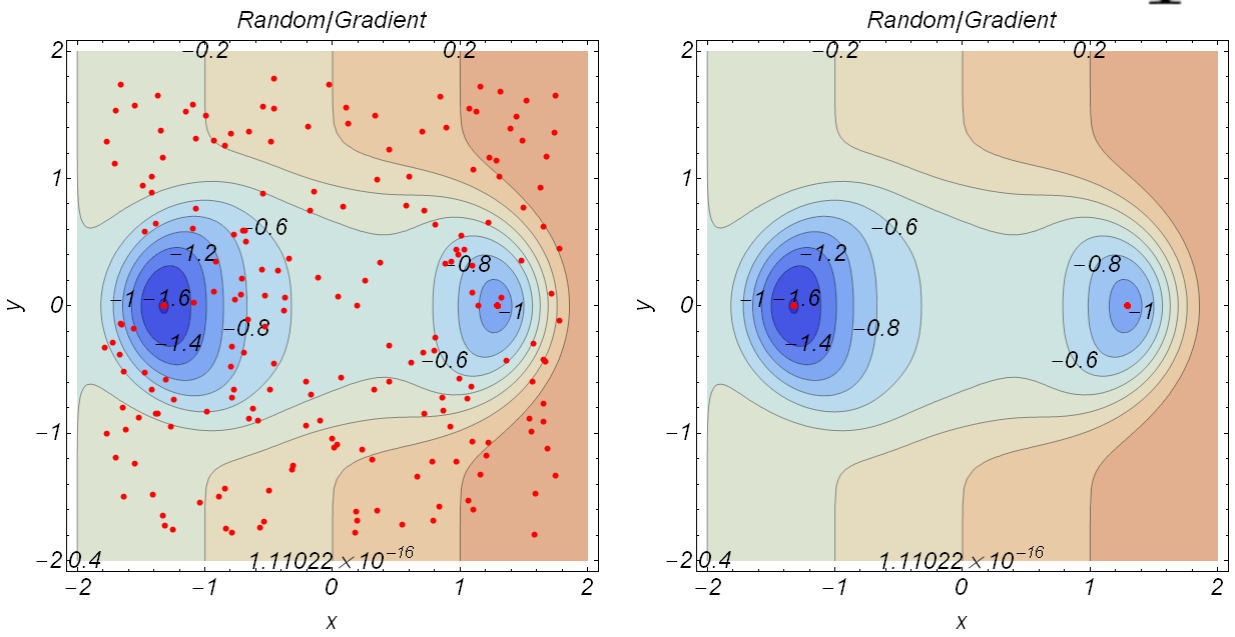




Random Search

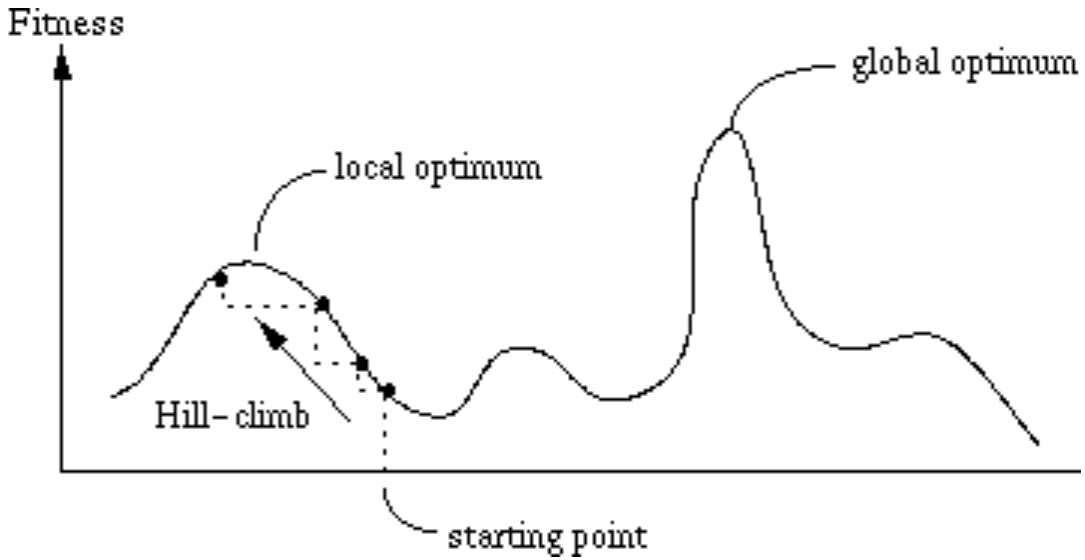
- Choose points randomly in the configuration space. Unintelligent, and rarely used by itself.
- Can be combined by doing single-point optimisation of each random point.
- Useful for comparing with other methods to see if they're working!
- Of course, with enough points you will eventually find the optimum – but just imagine how many points you need with many dimensions of configuration space.

$$T \sim n^p$$



Stochastic Hill Climbing

- Instead of just climbing up the nearest hill and you can also make random steps, retaining the move if the fitness is improved.
- Easy to implement and fast, but is 'noisy' if there are many small peaks.



Simulated Annealing (Metropolis, 1953)

- Analogy with thermodynamics - a liquid cooled slowly forms a large crystal where the atoms are nearly at their minimum (optimum) energy state.
- Key to optimisation process is *slow* cooling, where there is time for movement to the lowest energy state - this is *annealing*.
- The previous methods correspond to *quenching*.
- Boltzmann distribution gives probability of system being in a state of energy *E,*

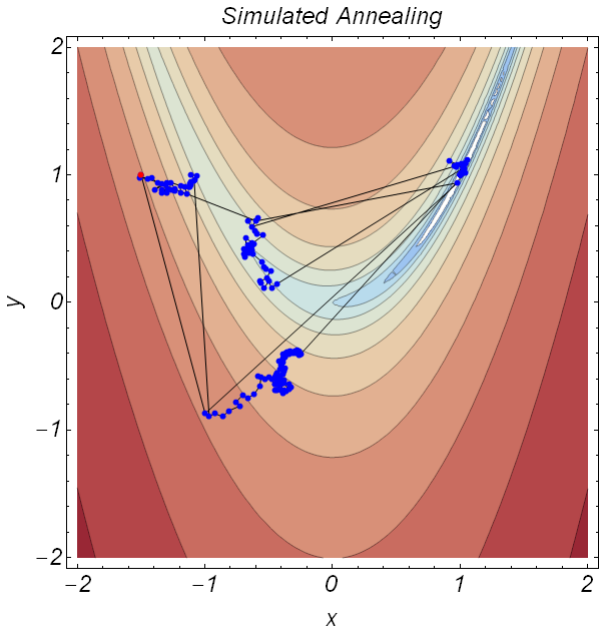
$$P(E) \sim \exp\left(\frac{-E}{kT}\right)$$

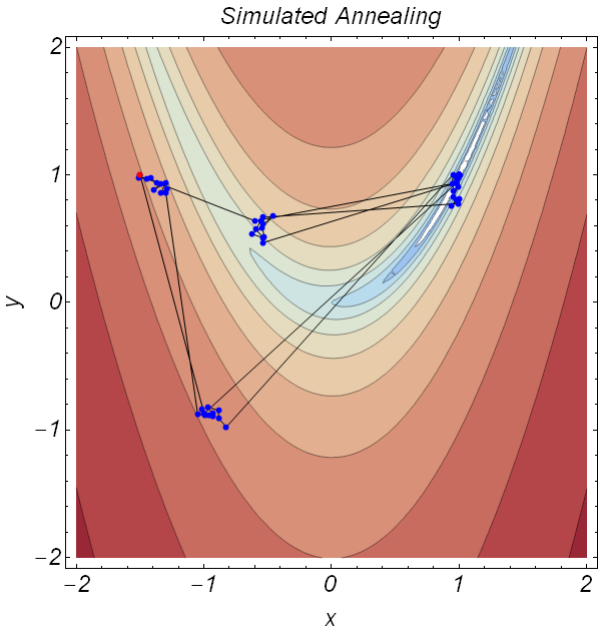
- Simulated annealing gives probability of transition from energy E_1 to E_2 with probability

$$p = \exp\left[\frac{-(E_2 - E_1)}{kT}\right]$$

Simulated Annealing (Metropolis, 1953): Implementation

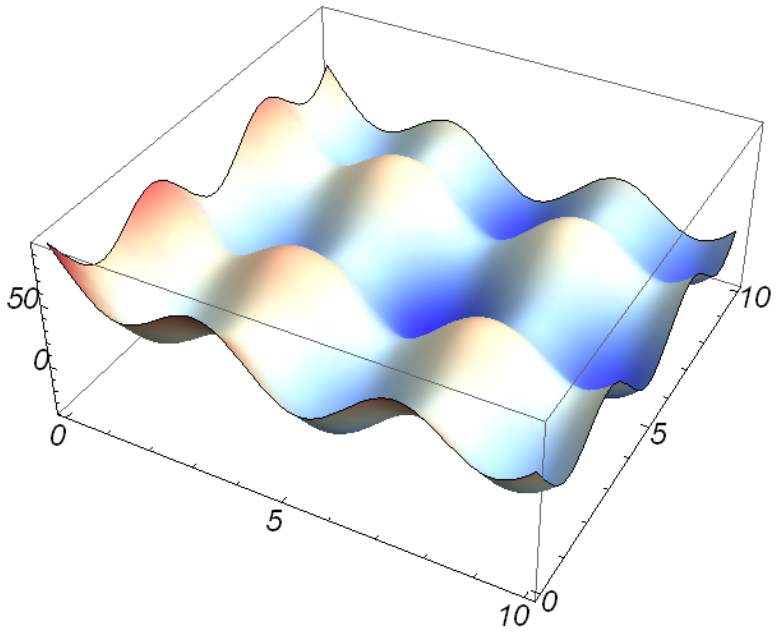
- The algorithm uses the following elements:
  - *1. A generator of random changes in the configuration.*
  - *2. An objective function **E** (analog of energy) to minimise.*
  - *3. A control parameter **T** (analog of temperature) and an **annealing schedule**.*
- High T gives high P of moving to a worse state - explores configuration space.
- Low T gives settling to final optimum.
- Infinitely slow cooling guarantees finding the global minimum.





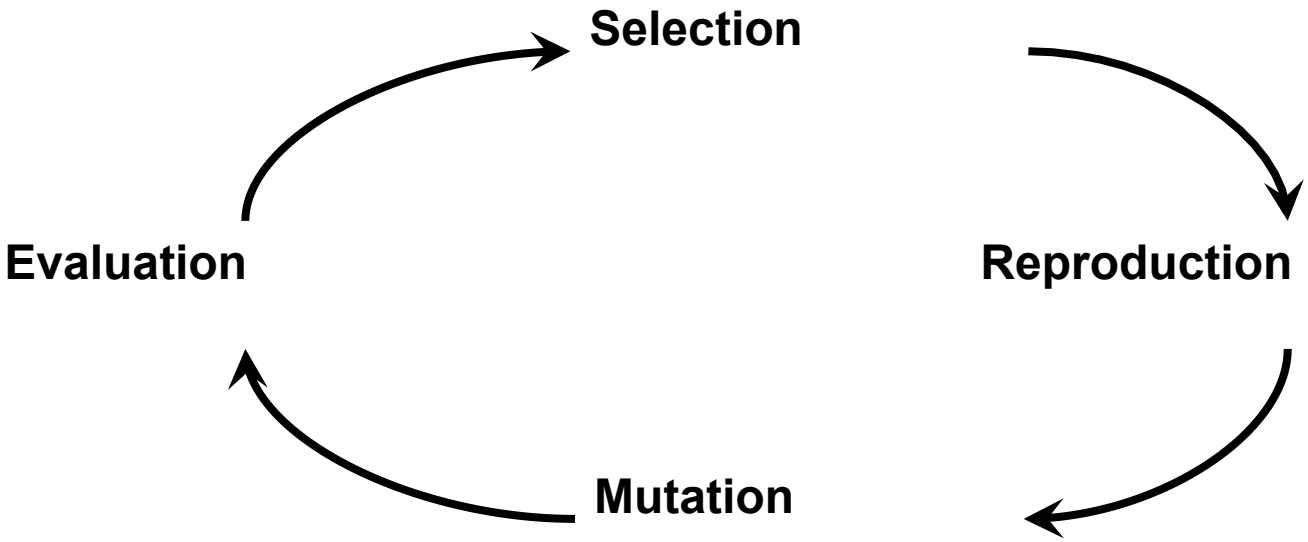
Multiple-Minima Functions

- In real life (i.e. accelerators), your system will be very ‘messy’, with multiple minima.



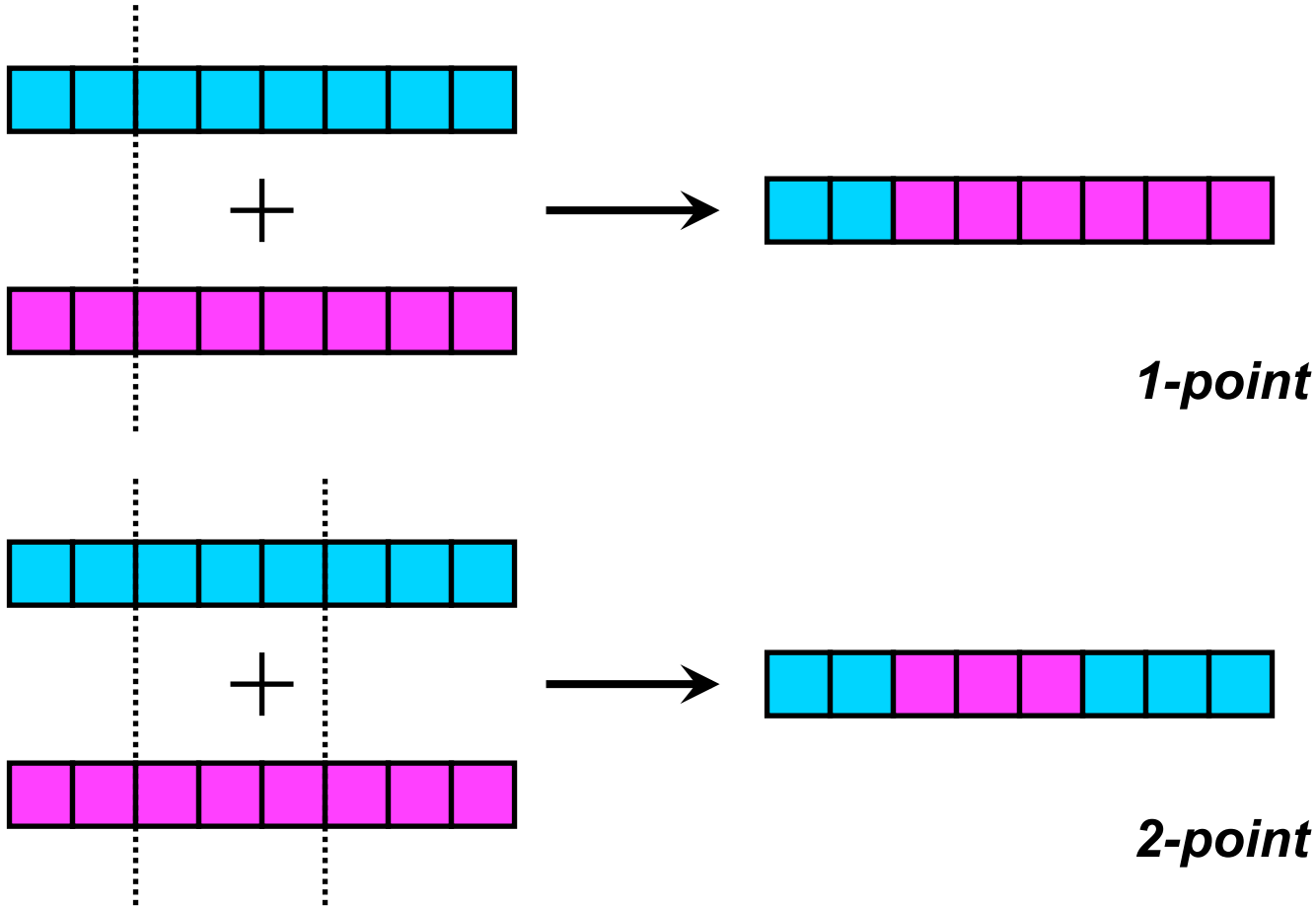
Genetic Algorithms (Holland, 1975)

- Concept is *Population* of points in configuration space. Each point *P* is represented by a *Gene* - a binary representation which can be decoded to give the *Phenotype* - the position in configuration space/particular design.
- The Population is allowed to *Evolve* through interaction between the individuals. Eventually the population will *Converge* to a fitter region of the configuration space.



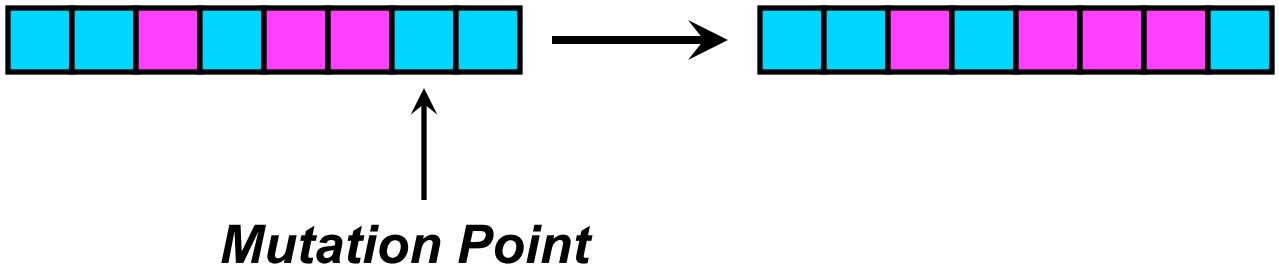
Genetic Algorithms - Reproduction

- Reproduction proceeds through crossover:



Genetic Algorithms - Mutations

- Mutations are characterised by a *Mutation Rate*.



Genetic Algorithms – Selection and Convergence

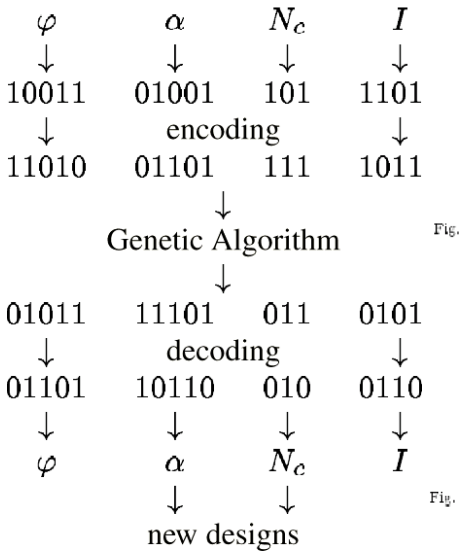
- Selection can proceed in various ways:
  - 1. Only the best children are kept (no parents kept).
  - 2. Parents and children are ranked together, and only the best are kept.
  - 3. Each child is compared to the parent most like it (using the *Hamming Distance*), which it replaces if it is better - This method is called *Niching*.
- The method of selection is important as it is obviously non-stochastic. Selection gives pressure toward fitter regions of configuration space.
- The selection procedure and the mutation rate are important for determining how fast the population converges to a particular region of configuration space.
- The convergence rate determines how much 'variety' is tried.
- Strong analogy with Simulated Annealing technique, and with damping and excitation in phase space.
- Selection is analogous to damping, mutation is analogous to noisy excitation.

Genetic Algorithms (GAs) and Evolutionary Programs (EPs)

- There are a HUGE number of implementations of GAs and EPs. However, what you need to know is:

- Genetic Algorithms *quantise* each variable: $n = 2^b$
- There is a formal proof that GAs work

- Evolutionary Programs allow a variety of continuous variables.
- There is no formal proof that they work, but they are used a lot because they provide good optimisation.



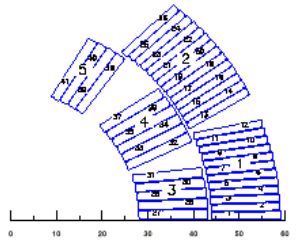


Fig. 4. Coil cross-section for the 5 block (41 turns) design (V5)

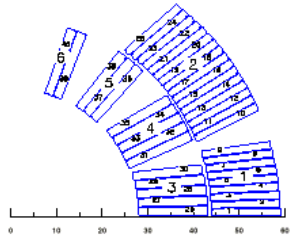


Fig. 5. Coil cross-section for the 6 block (40 turns) design (V6-1)

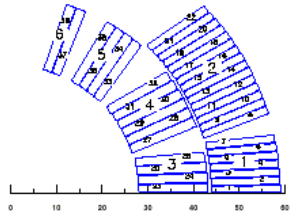
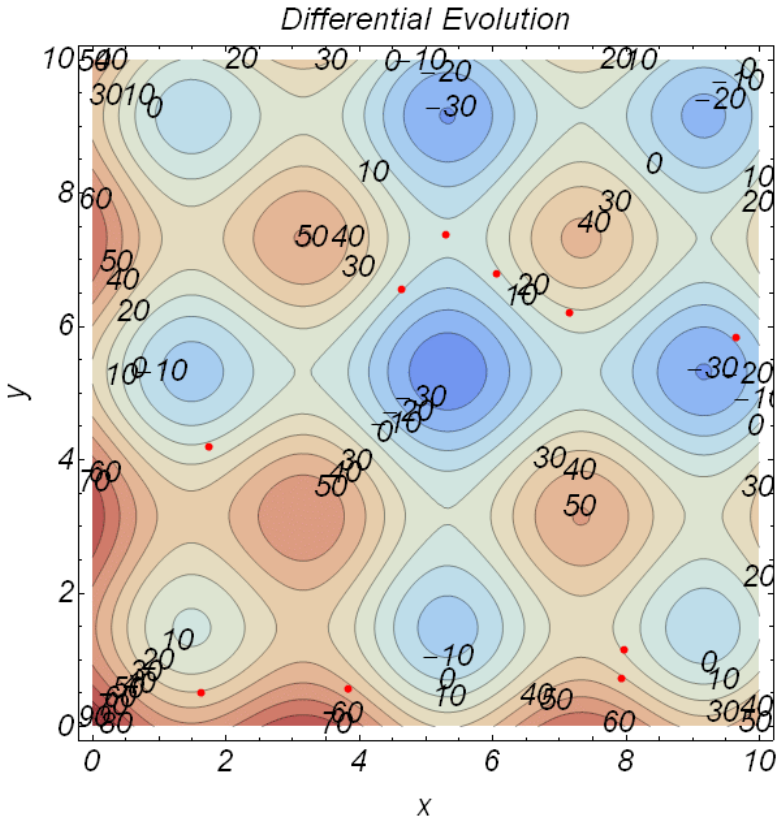
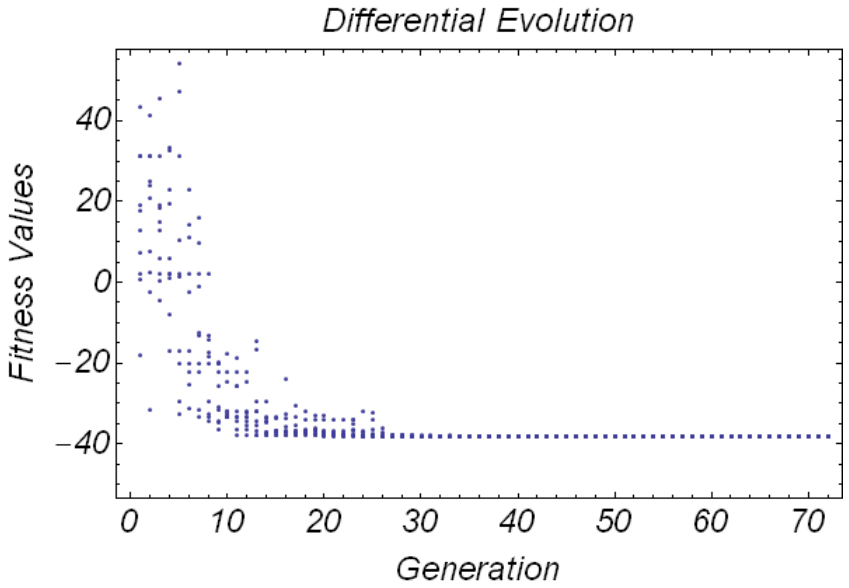


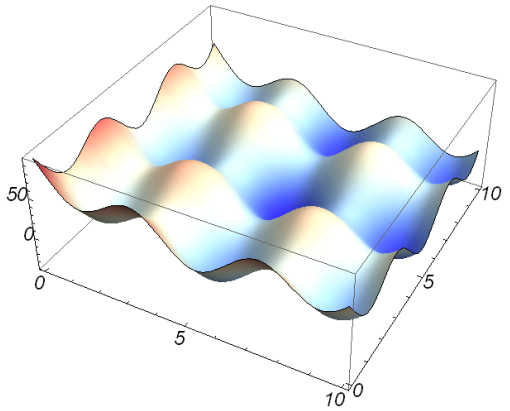
Fig. 6. Coil cross-section for the 6 block (38 turns) design (V6-3)

LHC dipole optimisation (Russenshuck, 1998)

Evolutionary Program with Population Size of 10







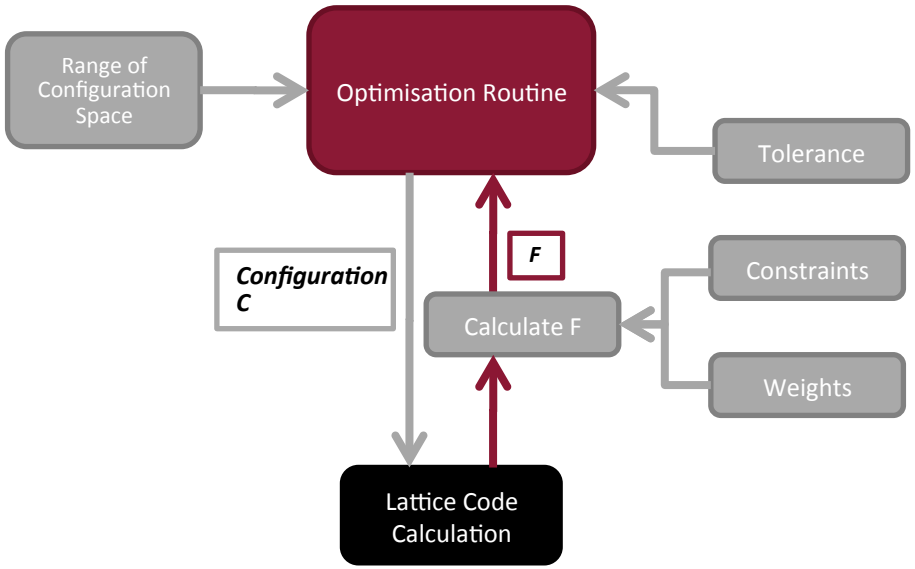
GAs vs. Single-Point methods

- Genetic algorithms have distinct advantages over classical single-point optimisation techniques for particular classes of problems:
  - 1. Best area of configuration space is not known
  - 2. Many peaks/discontinuous Objective Function
  - 3. Best solution not required - 'good enough' needed
- Hybrid solutions are popular, combining several methods.
- No particular algorithm is best in the general case.

Wolpert and Macready (1995)

- The 'No free lunch theorem'
- Important general theorem of search algorithms:
  - **'All algorithms that search for an extremum of a cost function perform exactly the same, when averaged over all possible cost functions.'**
- In other words, if algorithm A outperforms algorithm B for some cost functions, then there must exist as many functions where B outperforms A.
- The corollary to this is that *the algorithm must be matched to the particular objective function* to perform well.

Weights and Constraints: Practical Issues

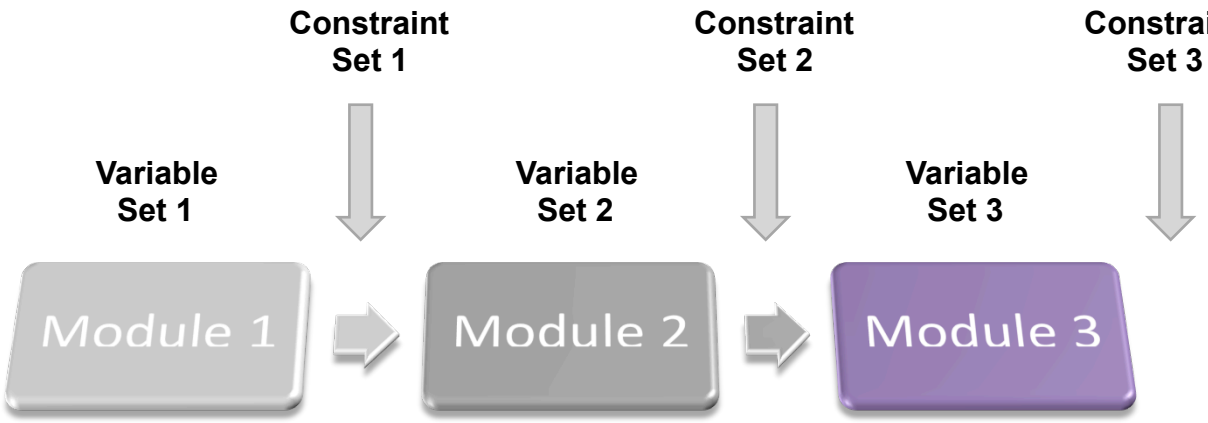


- **Variables** give you a region of configuration space to work in
  - e.g. limits on quad strengths
- **Constraints** are your target values
  - e.g. beta functions, tunes, chromaticity
- The **Objective Function F** is the combination of **Constraints** and **Weights**
- **Tolerance** is when to stop – when the change in **F** is less than the Tolerance

Over-Constrained and Under-Constrained Problems

- An **Over-Constrained** problem is one where **Constraints > Variables**
- Typical symptoms:
  - Objective function target cannot be achieved
  - Two or more variables go to their limits (but watch out for your variable range)
- An **Under-Constrained** problem is one where **Constraints < Variables**
- Typical symptoms:
  - Objective function target is achieved easily, but some features of the system take on wild values (crazy beta functions are very common)
  - A single variable (e.g. a quad strength) seems to oscillate wildly without any particular benefit, especially between runs – a sign that it is not coupled to the constraints
- Note: sometimes it can be difficult to spot whether a system is over- or under-constrained, as some constraints are implicitly coupled:
  - Example - tunes vs. beta functions, which are dependent on each other

Tips for Setting Constraints and Variables – Aperiodic Optical System



- Stages:

1. Constraint Set 1 with Variable Set 1
2. Constraint Set 2 with Variable Set 1 & 2
3. Constraint Set 3 with Variable Set 1,2,3

- But you should be flexible. This is an art not a science! ;)

Tips for Constraints and Weights

- Constraints can contribute to the objective function in a number of ways – this will depend on the code you use (or write).
- A typical routine will have targets with the following pseudo-code:

- `betax=20,weight=1;`
- `betay=10,weight=1;`
- `etax=0,weight=5;`

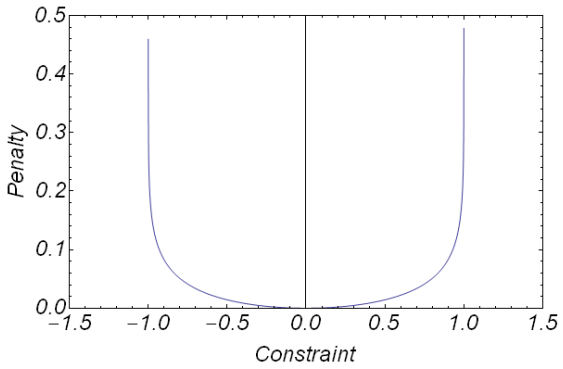
- Typical formulation with weights: $F = \sum_i w_i (x_i - c_i)^2$

- But you also see routines with the following code:

$$F = \sum_i \frac{-w_i}{(x_i - c_i)^2}$$ **Inverse barrier**

$$F_i = -w_i \epsilon log[(x_i - c_l)(c_u - x_i)]$$ **Logarithmic barrier**

- `betax<20,weight=1;`
- `betay<10,weight=1;`
- `etax=0,weight=5;`



Logarithmic barrier plot

Fashionable Topics

- There are a number of other techniques in optimisation that you may encounter or use. For example:

- **Pareto-front/Multi-objective optimisation:**
- This looks at the trade-offs of one variable with respect to another on the overall optimisation of a system.
- Example application: what is the trade-off between bunch length and emittance obtainable for different bunch charges.

- Also other optimisation methods, such as **particle swarm optimisation** which is quite fashionable at the moment.

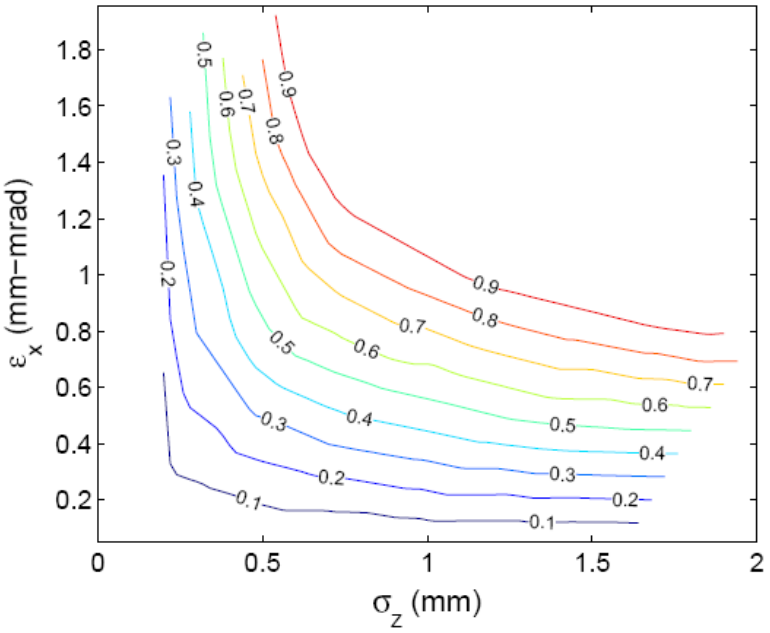


Figure 3: Transverse normalized rms emittance vs. bunch length for various charges in the injector (nC).

(Bazarov, PAC 2005)