Imperial College
London

Computational Physics: Minimisation

19 and 23 October 2018

Outline

- Minimisation: an introduction
- In one dimension
  - the Parabolic Method (briefly)
- Iterative Multi-dimensional methods
  - Univariate method
  - Gradient method
  - Newton's method
  - Global minimum search
- Monte Carlo Minimisation
  - Simulated annealing

Introduction to Function Minimisation

- Also “Maximisation” or “Optimisation”
  - refer to the same thing (modulo a minus sign or other minor difference in formulation)
- Finding local or global minima (maxima) of a function of one variable $f(x)$ and of many independent variables $f(x_1, x_2, \ldots, x_N)$ ($f(\vec{x})$, where $\vec{x}$ is an N–dimensional vector)
- Find both the position of a minimum $\vec{x}_\star$ and the value of the function there $f(\vec{x}_\star)$
- The function to be minimised is often called the “cost function” (and not just in economics) or “loss function”
- Mainly iterative methods
- Focus here on “unconstrained” minimisation (as opposed to constrained minimisation)
- Will also limit ourselves to real valued, scalar functions $f(\vec{x}) \in \mathbb{R}$ and real valued variables; $x \in \mathbb{R}$ and $\vec{x} \in \mathbb{R}^N$
- The principles used here apply to these cases

(by the way, ‘minimum’ is the singular and ‘minima’ the plural. You look for the global minimum out of many local minima)

Uses of Minimisation

- Fitting a model to data (parameter fitting)
  - observed data d_i^{obs} with errors σ_i
  - fit the model $d_i^{\text{model}} = a + \alpha X_i$
  - find the minimum χ^2 with respect to the parameters a and α:

$$\chi^2(a, \alpha) = \sum_{i=1}^{N^{\text{data}}} \frac{\left(d_i^{\text{obs}} - d_i^{\text{model}}\right)^2}{\sigma_i^2}$$

  - maximising the likelihood

$$L(a, \alpha) \propto e^{-\frac{1}{2}\chi^2(a,\alpha)},$$

  - this is a very simple example: the statistical analysis of data involves much more sophisticated methods (see Computational Physics in Action lectures)
- Roots of a function can be found through the minima of its square
- Matrix equations can be solved through minimisation (next week)

Minimisation Algorithms

- Analytically, minimisation is simple: find the roots of the derivative function
  - but this is often not trivial in practice: need methods that only use the function values, and sometimes approximations for the derivatives
- After the first several lectures on root-finding, function minimisation in 1D should be quite intuitive...

Simple Search One Dimension

- Start at one end of your interval and increment until the value of the function starts rising, and then go back a bit with smaller increments
  - may only find the first local minimum
  - or could miss the minimum
  - or could waste a lot of time before getting close to a minimum
  - depending on the choice of increment

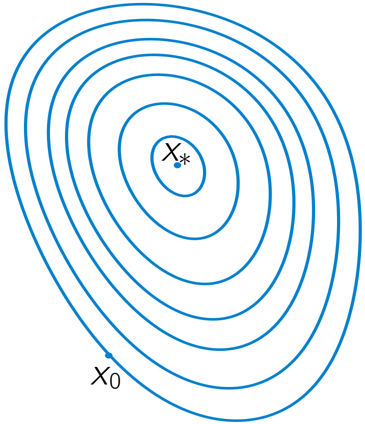
Parabolic Method in One Dimension

- Physical functions are often smooth–and approximate a parabola near a minimum
- In a region where the curvature of $f(x)$ is positive
- Select three points x_0, x_1, and x_2 where $f(x_0) = y_0,\ \ f(x_1) = y_1,\ \ \text{and}\ \ f(x_2) = y_2$
- Fit $P_2(x)$, the second-order Lagrange polynomial, through the points
- The minimum of the parabola is found at x_3 given by:

$$\boxed{x_3 = \frac{1}{2}\,\frac{(x_2^2 - x_1^2)y_0 + (x_0^2 - x_2^2)y_1 + (x_1^2 - x_0^2)y_2}{(x_2 - x_1)y_0 + (x_0 - x_2)y_1 + (x_1 - x_0)y_2}}$$

- Keep the three lowest points out of $f(x_0)$, $f(x_1)$, $f(x_2)$ $f(x_3)$ and repeat (relabelling the remaining points)
- x_3 will converge towards $x_\star$; stop when the change in x_3 is less than a desired value

Minimisation in Multiple Dimensions



- How can we go from x_0 to x_*?

And then imagine

Minimisation in Multiple Dimensions

Univariate Method

- Method
  - Search along the
  - Then search along “ “
  - Iterate until convergence
- Not efficient
  - especially if the contours are at an angle to the coordinate axes

Minimisation in Multiple Dimensions

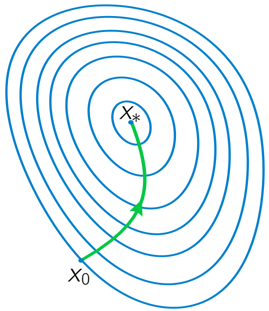
Gradient Method

- Also known as following the path of "steepest descent"
- Method
  - Find the gradient vector
  - Take a small step in the opposite direction:

Minimisation in Multiple Dimensions

Gradient Method (continued)

- The gradient $\vec{\nabla} f$ can be found analytically, or using a finite difference approximation (to be studied in detail later in the course); e.g. via a Forward Difference Scheme applied in each variable x_i for $i = 1, \ldots, N$:



$$\frac{\partial f}{\partial x_i} \approx \frac{f(x_1, x_2, \ldots, x_i + \Delta, \ldots, x_N) - f(x_1, x_2, \ldots, x_i, \ldots, x_N)}{\Delta}$$

Minimisation in Multiple Dimensions

Newton's Method

- Use the local curvature of the function to improve efficiency
- Starting at a location $\vec{x}_0$, the minimum is displaced $\vec{\delta}$ away, at position $\vec{x}_0 + \vec{\delta}$
- How can we estimate $\vec{\delta}$?
- The Taylor expansion for the function about $\vec{x}$ (later, $\vec{x}$ will be set to $\vec{x}_0$) is:

$$f(\vec{x}+\vec{\delta}) = f(\vec{x}) + [\vec{\nabla} f(\vec{x})]^T \cdot \vec{\delta} + \frac{1}{2}\vec{\delta}^{\,T} \cdot \mathbf{H}(\vec{x}) \cdot \vec{\delta} + \mathcal{O}\left(|\vec{\delta}|^3\right),$$

- where **H** is the **Hessian** or **Curvature** matrix (an $N \times N$ matrix)

$$H_{ij}(\vec{x}) = \frac{\partial^2 f(\vec{x})}{\partial x_i \partial x_j}$$

  - See Eq. 2.4 and the paragraph which follows it in the printed Lecture Notes

Two-Dimensional Functions

And using the Hessian:

- Any function that is the sum of the terms x^2, y^2, $x \times y$, x, y with coefficients and a constant can be written:

$$f(x,y) = \frac{1}{2}\begin{pmatrix}x & y\end{pmatrix} \qquad \begin{pmatrix}x\\y\end{pmatrix} + \begin{pmatrix}a & b\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} + C$$

- Which is to say:

$$f(x,y) = \frac{1}{2}\left(\qquad\qquad\qquad\qquad \right) + ax + by + C$$

Minimisation in Multiple Dimensions

An Example

Example (adapted from Gerald and Wheatley p. 424)

Consider the 2D parabolic function

$$f(x,y) = x^2 + 2y^2 + xy + 3x$$

where we know the true minimum is at $\vec{x}_\star = (-12/7, 3/7)$ from equating the derivatives to zero.

- At any given point (x, y), the gradient and Hessian are:

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} \quad \\ \quad \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix} = \begin{pmatrix} \quad \\ \quad \end{pmatrix}$$

- It is easy to find the inverse of the Hessian matrix:

$$\mathbf{H}^{-1} = \begin{pmatrix} \quad \\ \quad \end{pmatrix}$$

Minimisation in Multiple Dimensions

An Example

Example (adapted from Gerald and Wheatley p. 424)
Consider the 2D parabolic function

$$f(x, y) = x^2 + 2y^2 + xy + 3x$$

where we know the true minimum is at $\vec{x}_\star = (-12/7, 3/7)$ from equating the derivatives to zero.

- Hence starting at $\vec{x} = \vec{x}_0 = (x_0, y_0)$, the equation for $\vec{\delta}$ yields:

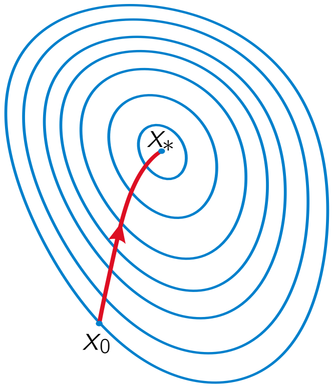
$$\begin{aligned} \vec{\delta} &= -\mathbf{H}^{-1}\nabla f = -\left(\qquad\qquad \right) \cdot \left(\qquad\qquad \right) \\ &= -\frac{1}{7}\left(\begin{pmatrix} 8x_0 + 4y_0 + 12 - x_0 - 4y_0 \\ -2x_0 - y_0 - 3 + 2x_0 + 8y_0 \end{pmatrix}\right) = \begin{pmatrix} -x_0 - \frac{12}{7} \\ -y_0 + \frac{3}{7} \end{pmatrix} = \vec{x}_\star - \vec{x}_0 \end{aligned}$$

- so that $\vec{\delta}$ takes us directly to the minimum (i.e. $\vec{x}_0 + \vec{\delta} = \vec{x}_\star$) in this case of a parabolic function.

Minimisation in Multiple Dimensions

Newton's Method (Continued)

- For a more realistic non-parabolic function, one needs to iterate Newton's Method to approach the minimum



Minimisation in Multiple Dimensions

The Quasi-Newton Method

Newton's method works very well, but

- needs the first and second derivatives
- and the inverse of the Hessian matrix ($\mathcal{O}\left(N^3\right)$ operation)

The Quasi-Newton Method instead directly approximates the inverse Hessian matrix, using the local gradient only.

- The basic iteration step is

$$\vec{x}_{n+1} = \vec{x}_n - \alpha\, \mathbf{G}_n \cdot \vec{\nabla} f(\vec{x}_n)$$

  where $\mathbf{G}_n$ is an **approximation** of $\mathbf{H}^{-1}(\vec{x}_n)$, and $\alpha \ll 1$.
- For the first iteration $\mathbf{G}_0$ is set to the identity matrix $\mathbf{I}$, which makes this iteration the same as a gradient search
- To find $\mathbf{G}_{n+1}$ we use the changes in $\vec{x}$ and $\vec{\nabla} f$:

Minimisation in Multiple Dimensions

The Quasi-Newton Method

- From the previous slide:

$$\vec{\delta}_n = \vec{x}_{n+1} - \vec{x}_n, \;\; \vec{\gamma}_n = \vec{\nabla} f(\vec{x}_{n+1}) - \vec{\nabla} f(\vec{x}_n)$$

- Then comparing the above expression $\vec{\gamma}_n$ with the **gradient of the Taylor expansion of $f(\vec{x})$,** to linear order:

$$\vec{\nabla} f(\vec{x}_{n+1}) \approx \vec{\nabla} f(\vec{x}_n) + \vec{\nabla}\left(\vec{\nabla} f(\vec{x}_n) \cdot \vec{\delta}_n\right)$$

- We obtain

$$\mathbf{H}_n^{-1} \cdot \vec{\gamma}_n = \vec{\delta}_n$$

(see earlier relationship between **H** and $\vec{\Delta} f$)

- The trick is to update **G** to satisfy

$$\mathbf{G}_n \cdot \vec{\gamma}_n = \vec{\delta}_n$$

Minimisation in Multiple Dimensions

The Quasi-Newton Method

- The trick is to update **G** to satisfy

$$\mathbf{G}_n \cdot \vec{\gamma}_n = \vec{\delta}_n$$

- Various methods exist on how to do this, with one common example being the Davidon–Fletcher–Power algorithm

  where $(\vec{u} \otimes \vec{v})_{ij} \equiv u_i v_j$ is the outer product of $\vec{u}$ and $\vec{v}$
  - we will not prove this—the existence of such algorithms is key, not individual specifics
- Only involves (at worst) matrix multiplications at each iteration and the resulting (approximated) Hessian is positive definite by construction

Previously: Lecture 8

- Minimisation: an introduction
- In one dimension
  - the Parabolic Method (briefly)
- Iterative Multi-dimensional methods
  - Univariate method
  - Gradient method
  - Newton's method
  - Global minimum search

Search for the Global Minimum

- None of these methods are foolproof against finding a local minimum instead of the global minimum
- Combining different strategies, different starting points etc., should help
- Once again, there is no single recipe that will always work for any function all the time
- Even a simple grid search (literally to calculate the values of the function on a pre-defined grid) can be a useful tool

Monte Carlo Minimisation

Monte Carlo methods offer an entirely different approach to minimisation, with complementary properties that are often very useful—in particular when:

- the degrees of freedom in the system are so many that a direct search for an optimal configuration is too time consuming
- many local minima exist in which direct search methods can get stuck instead of finding the global minimum

Here we discuss the combination of the Metropolis method and *Simulated Annealing*

- “The thermodynamic approach to the structure analysis of crystals”, Khachaturyan, A.; Semenovsovskaya, S.; Vainshtein, B., Acta Crystallographica Section A vol. 37, issue 5, pp. 742–754 (Sep 1981)
- “Optimization by simulated annealing”, S. Kirkpatrick , C. D. Gelatt , M. P. Vecchi (1983), Science, New Series, Vol. 220, No. 4598. (May 13, 1983), pp. 671–680
- Will demonstrate these at the end of the course “Computational Physics in Action”

“Annealing”

Oxford English Dictionary *anneal, v.* 4a:
To toughen anything, made brittle from the action of fire, by exposure to continuous and slowly diminished heat, or by other equivalent process.

Monte Carlo Minimisation

Simulated Annealing

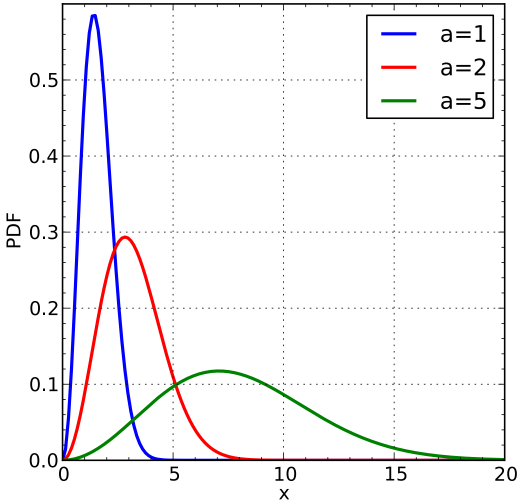
Motivated by thermodynamics:

- Random fluctuations can temporarily move a system to a less-optimal (higher-energy) configuration
- So use the Boltzmann PDF:

$$P(E)\,dE \sim$$

  where the “energy” E encodes a cost function and the “temperature” T determines the probability of changes in the “energy”
  - note that that these are not the actual E and T in a physical system, but are used to emphasise the analogy to real annealing
- This is a use of the Metropolis Algorithm
  - and a form of Markov Chain Monte Carlo (MCMC) if you would like to do some optional reading around this topic
- The additional idea in simulated annealing is that the “temperature” is gradually lowered

Maxwell–Boltzmann Distribution



- a is the scale factor: proportional to $\sqrt{T}$

Monte Carlo Minimisation

Simulated Annealing

The algorithm:

- At each step in the iteration, change the system in a random way
    - the specifics of this are system-dependent
- The value of the function to minimise is the "energy": $E = f(\vec{x})$
- Calculate the "energy" of the system before (E_1) and after (E_2) the change to obtain the change in the system's "energy": $\Delta E = E_2 - E_1$
- Accept the step with a probability p_{acc} given by:

- Iterate as with the Metropolis Algorithm,

Conclusion

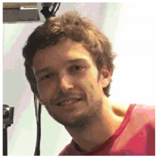
We have looked at

- Function Minimisation/Maximisation or Optimisation
- Iterative methods
  - in one dimension
  - in multiple dimensions
    - Univariate Method
    - Gradient Method
    - Newton's Method
    - The Quasi-Newton Method
  - Monte Carlo methods
    - Simulated Annealing

Alex

Bingshen

Carl


Hugh

Tom L

Phill

Ravi

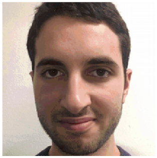
Justin

Computational Physics Demonstrators 2018

Blackett Laboratory

Sam

Šarunas


Timur

Tom H

Vito

Not-a-Bonus: The Assignment

- Later today, the Assignment will appear on Blackboard
- The deadline is **12 noon on Monday 12th November 2018**
- Code submission as a single zip file, including a `README.txt` file
- Write-up as a PDF
- Submission details are provided alongside the questions
- If you are not using Python, you **must** let us know by email
- Practical sessions start at 9am tomorrow in the Computing Suite; for both Problem Sheets and Assignment work
- Please have a go at the PSes beforehand!