THE UNIVERSITY of York

Function minimization and Quasi-Newton methods

Matt Probert
August-Wilhelm Scheer Visiting Prof TUM 2015
Condensed Matter Dynamics Group
Department of Physics,
University of York, U.K.
http://www-users.york.ac.uk/~mijp1

Overview

- What is function minimization?
- Quasi-Newton methods
  - Steepest descents
  - BFGS / L-BFGS
  - Conjugate gradients
  - TPSD
- Example
- Summary

Function Minimization

Function minimization

- Simple in 1D
  - Move downhill in some function $f(x)$ until get to the bottom
  - Minimum defined by $f'(x) = 0$
  - This is obviously easier if have derivative information about the function
    - If function is non-differentiable or too expensive to differentiate then have to proceed by search method, e.g. bisection

Newton's Method

- Start with Newton-Raphson method for finding *roots* of a function $x^* \; s.t. \; f\left(x^*\right) = 0$:

$$x_{n+1} = x_n - \frac{f\left(x_n\right)}{f'\left(x_n\right)}$$

- Can also be extended to locating minimum by finding roots of $f'\left(x\right)$ …

$$x_{n+1} = x_n + \Delta x = x_n - \frac{f'\left(x_n\right)}{f''\left(x_n\right)}$$

Newton in Higher Dimensions

- In 3D

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \left[\mathbf{H}(f)\right]^{-1} \nabla f\left(\mathbf{x}_n\right)$$

where $\nabla f\left(\mathbf{x}\right)$ is the gradient of $f\left(\mathbf{x}\right)$

$$\nabla f\left(\mathbf{x}\right) = \frac{\partial f}{\partial x}\mathbf{i} + \frac{\partial f}{\partial y}\mathbf{j} + \frac{\partial f}{\partial z}\mathbf{k}$$

and ***H*** is the *Hessian* of the function which is the matrix of second derivatives

$$\mathbf{H}_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

Problem with Newton's Method

- Newton-Raphson and Newton's method only work well if have analytical derivatives
- Generally it is very hard to calculate Hessian
  - And expensive to calculate matrix inverse
  - Approximating it by finite differences is both very expensive and not very accurate / stable
- Hence use quasi-Newton methods instead …

Avoiding Hessian altogether

- Before we go into quasi-Newton methods, what happens if we just use gradient?
- At any point we can do Taylor series expansion and so to 1st order:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla f\left(\mathbf{x}_n\right)$$

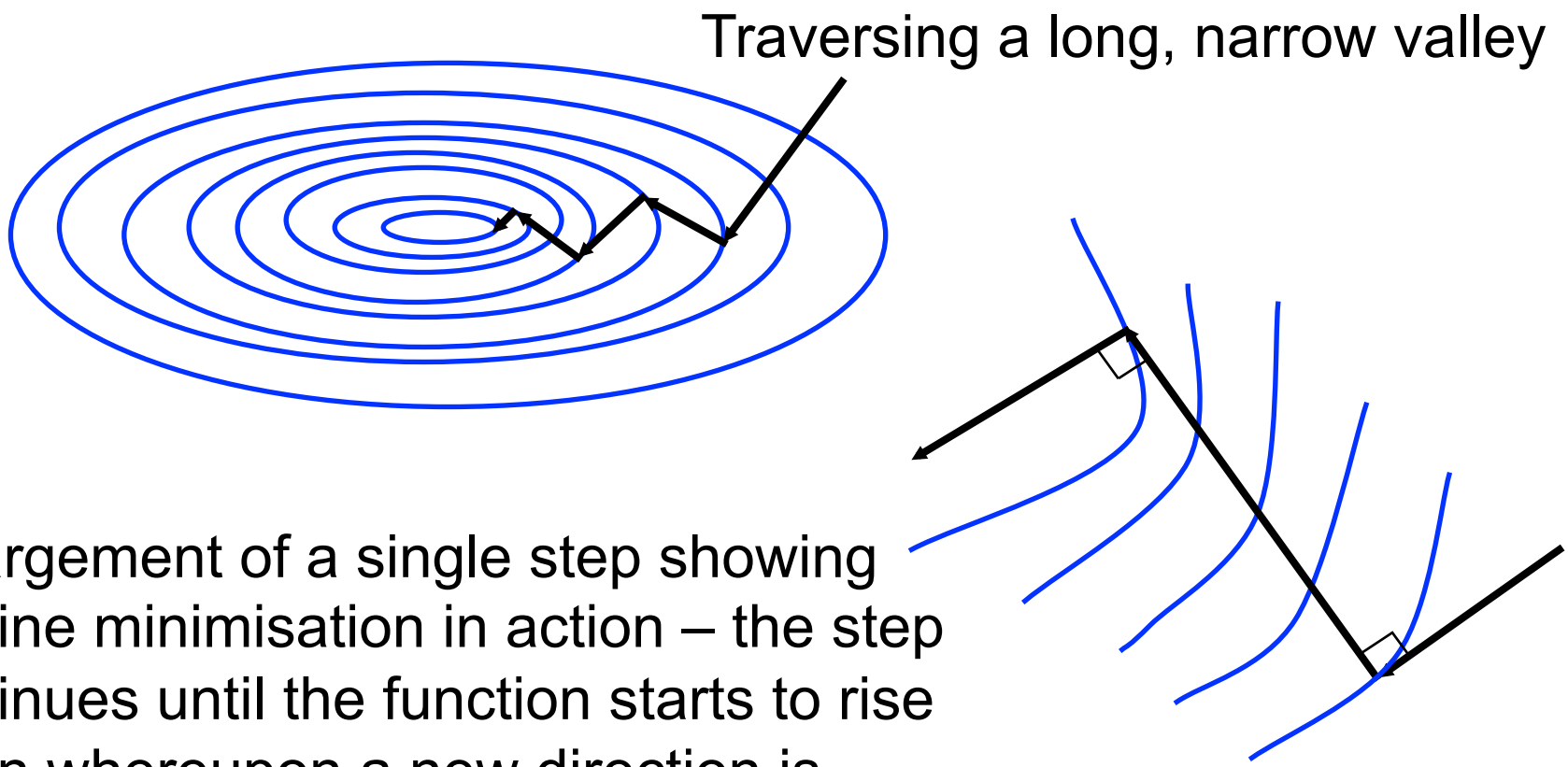
  where γ is the step-size

- Hence each iteration move downhill in direction of steepest slope → steepest descent method

Steepest Descent Method

- A very simple and robust method
- Need to combine with a *line minimization*
  - Find γ that minimizes f(x) in search direction
- Always goes downhill at each step
- No memory of previous directions
  - Hence can zig-zag depending on shape of f(x)
  - Can be very slow & inefficient
  - Hence quasi-Newton methods preferred …

Steepest Descents



Enlargement of a single step showing the line minimisation in action – the step continues until the function starts to rise again whereupon a new direction is selected which is orthogonal to the previous one

Quasi-Newton Methods

Quasi-Newton Method

- Start with Taylor series expansion:

$$f\left(\mathbf{x}_n + \Delta\mathbf{x}\right) \simeq f\left(\mathbf{x}_n\right) + \nabla f\left(\mathbf{x}_n\right)^T \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^T\mathbf{B}\Delta\mathbf{x}$$

- where ***B*** is an approximation to ***H***
- And change in gradient is

$$\nabla f\left(\mathbf{x}_n + \Delta\mathbf{x}\right) \simeq \nabla f\left(\mathbf{x}_n\right) + \mathbf{B}\Delta\mathbf{x}$$

- So best guess for how to update x_k is

$$\Delta\mathbf{x} = -\mathbf{B}^{-1}\nabla f\left(\mathbf{x}_n\right)$$

What about *B*?

- We start off with an initial guess for $\mathbf{B} = \mathbf{B}_0$
- Then do iterative update based upon

$$\nabla f\left(\mathbf{x}_n + \Delta\mathbf{x}\right) \simeq \nabla f\left(\mathbf{x}_n\right) + \mathbf{B}\Delta\mathbf{x}$$

also known as the *secant equation*

- So we use current $\mathbf{x}_n$,$f\left(\mathbf{x}_n\right)$etc to generate:

$$\Delta\mathbf{x_n} = -\alpha_n \mathbf{B}_n^{-1} \nabla f\left(\mathbf{x}_n\right)$$
$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta\mathbf{x}_n$$

- And then use $f\left(\mathbf{x}_{n+1}\right)$ etc to update $\mathbf{B}_n$ ….

BFGS scheme

- Broyden-Fletcher-Goldfarb-Shanno (BFGS)
- Instead of calculating $\mathbf{B}_n$ and then $\mathbf{B}_n^{-1}$ we make an iterative update of $\mathbf{B}_{n+1}^{-1} =$

$$\left(I - \frac{\Delta\mathbf{x}_n\mathbf{y}_n^T}{\mathbf{y}_n^T\Delta\mathbf{x}_n}\right)^T \mathbf{B}_n^{-1}\left(I - \frac{\mathbf{y}_n\Delta\mathbf{x}_n^T}{\mathbf{y}_n^T\Delta\mathbf{x}_n}\right) + \frac{\Delta\mathbf{x}_n\Delta\mathbf{x}_n^T}{\mathbf{y}_n^T\Delta\mathbf{x}_n}$$

- where

$$\mathbf{y}_n = \nabla f\left(\mathbf{x}_{n+1}\right) - \nabla f\left(\mathbf{x}_n\right)$$

BFGS advantages

- BFGS is an efficient scheme
  - For an *N*-dim quadratic function it should converge in max *N* iterations
    - As long as have exact line search etc
  - A general function may take more iterations due to non-quadratic and/or inexact line min
- The Hessian is built up iteratively
  - Can accelerate convergence with good initial guess – like a preconditioner
  - Can analyse ***H*** to get useful info about system

BFGS disadvantages

- Need a good initial guess at $\boldsymbol{B}_0$ (or $\boldsymbol{B}^{-1}_0$)
  - Solved in CASTEP for ionic minimization
- Requires ~$O(N^2)$ storage
  - If we used this electronic minimization then N~N_G for each band and **k**-point
    - Prohibitive!
  - OK for N_{ions} for geometry optimization
    - As long as N_{ions}<1000 as matrix is not distributed – memory limitation for large system – hence use L-BFGS instead

L-BFGS

- Low-memory version of BFGS
  - Each BFGS update adds 2 rank-1 vectors
  - So store list of updates instead of entire $\boldsymbol{B}^{-1}$
    - With m updates the storage is ~$O(mN)$
    - Typical m~30 so this is a BIG saving
  - This also helps with non-quadratic functions
    - Builds a short-term memory into $\boldsymbol{B}_n$ so does not remember initial steps where updates were poor due to non-quadratic shape
  - Added in CASTEP v6.0

What about electronic minimization?

Electronic Minimization

- So how can we minimize the electronic E?
  - Learned previously about iterative approach
    - Use energy derivative
    - Applying Hamiltonian in real/reciprocal space – diagonal representation for speed & RAM
    - Need to ensure orthogonalization
  - Cannot use BFGS as no Hessian
- Now for more details …

Recap

- Energy expression: $E = \sum_i \langle \Psi_i | \hat{H}_{ks} | \Psi_i \rangle$
- Gradient: $\frac{\delta E}{\delta \langle \Psi_i |} = \hat{H}_{ks} | \Psi_i \rangle$
- Orthonormalization constrained gradient:

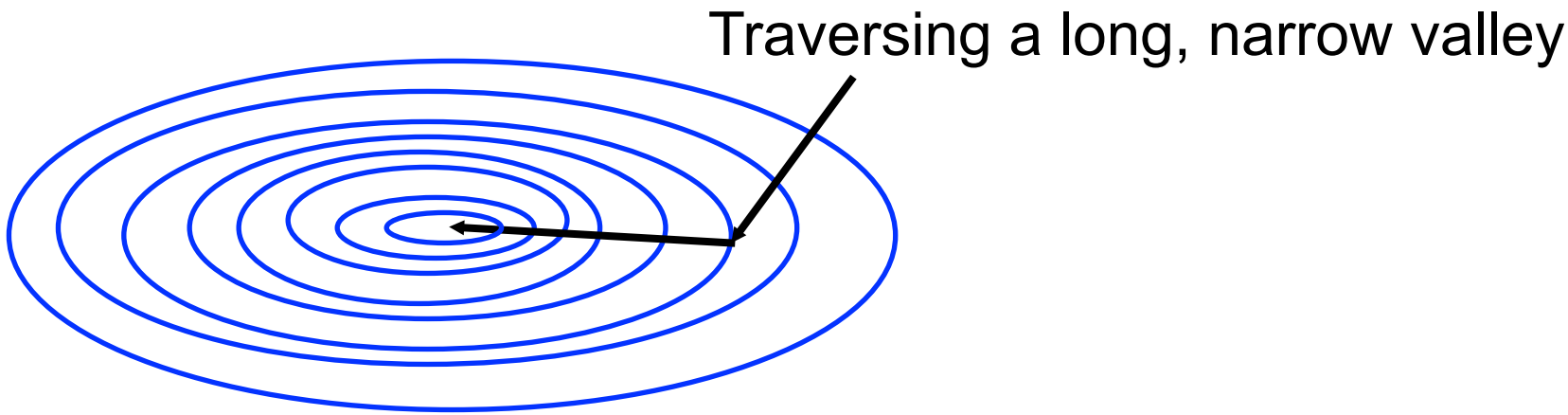
$$\hat{G} | \Psi_i \rangle = \hat{H} | \Psi_i \rangle - \sum_j \langle \Psi_j | \hat{H} | \Psi_i \rangle | \Psi_j \rangle$$

- So could use this for steepest descent search and do line minimization etc.

Conjugate Gradient Method

- Better approach than steepest descents (SD)
- The problem with SD is it has no memory
  - With exact line minimization each step is orthogonal to previous one
  - So new directions can repeat/undo old ones
- Conjugate Gradient (CG) constructs new search direction to be *conjugate* to present and all previous search directions!
  - Again is exact for quadratic form
  - And does not require an explicit Hessian

Conjugate Gradients



- The initial search direction is given by steepest descents.
- Subsequent search directions are constructed to be orthogonal to the previous *and* all prior search directions.
- No explicit Hessian required or generated.

CG in detail

- Conjugate?
  - Search direction $\boldsymbol{h}_i$ and gradient $\boldsymbol{g}_i$
  - Then conjugate => $\boldsymbol{g}_i . \boldsymbol{g}_j = 0$, $\boldsymbol{h}_i . \boldsymbol{B} . \boldsymbol{h}_j = 0$, $\boldsymbol{g}_i . \boldsymbol{h}_j = 0$
  - But we want to eliminate Hessian $\boldsymbol{B}$?
  - It can be shown that

$$\mathbf{h}_{i+1} = \mathbf{g}_{i+1} + \gamma_i \mathbf{h}_i$$

$$\gamma_i = \frac{(\mathbf{g}_{i+1} - \mathbf{g}_i) \cdot \mathbf{g}_{i+1}}{\mathbf{g}_i \cdot \mathbf{g}_i}$$

  - which is what CASTEP uses for electrons

CASTEP electronic minimization

- So use CG with constrained gradient to get min energy
  - But that was with given input initial $\rho^{in}(\mathbf{r})$
  - So not self-consistent as H=H[ρ]
  - Should we update H during the SCF cycle?
    - Cheaper to use fixed density during line min
  - Hence need to update $\rho^{out} = \sum_{b\mathbf{k}} |\psi_{b\mathbf{k}}|^2$

Density Mixing

- Correcting for line min error suggests

$$\rho^{new} = (1 - \alpha)\rho^{in} + \alpha\rho^{out}$$

- And iterate to self-consistency whence

$$\rho^{in} = \rho^{out}$$

- This is linear mixing – often sloshes
  - Better to use a dielectric response model
  - CASTEP has both Pulay & Broyden schemes
- Or could update ρ whenever ψ changes
  - Expensive but robust EDFT method

SD Revisited

Alternative Minimization

- Two-point steepest descents (TPSD)
  - A little known algorithm for minimization
  - No need for Hessian or a line minimization
  - Used as an alternative to BFGS – better for random structures and/or cell constraints
- Based upon normal SD but with two-point approximation to secant equation hence q-N:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla f(\mathbf{x}_n)$$

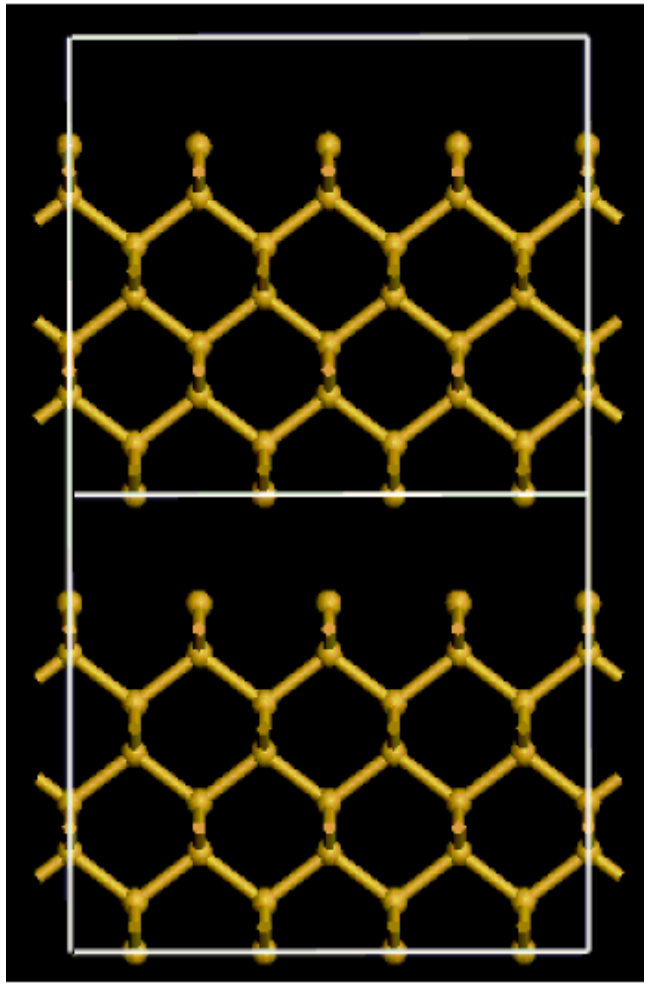
$$\gamma_n = \frac{(\mathbf{x}_n - \mathbf{x}_{n-1})^T (\nabla f(\mathbf{x}_n) - \nabla f(\mathbf{x}_{n-1}))}{(\nabla f(\mathbf{x}_n) - \nabla f(\mathbf{x}_{n-1}))^T (\nabla f(\mathbf{x}_n) - \nabla f(\mathbf{x}_{n-1}))}$$

TPSD in action

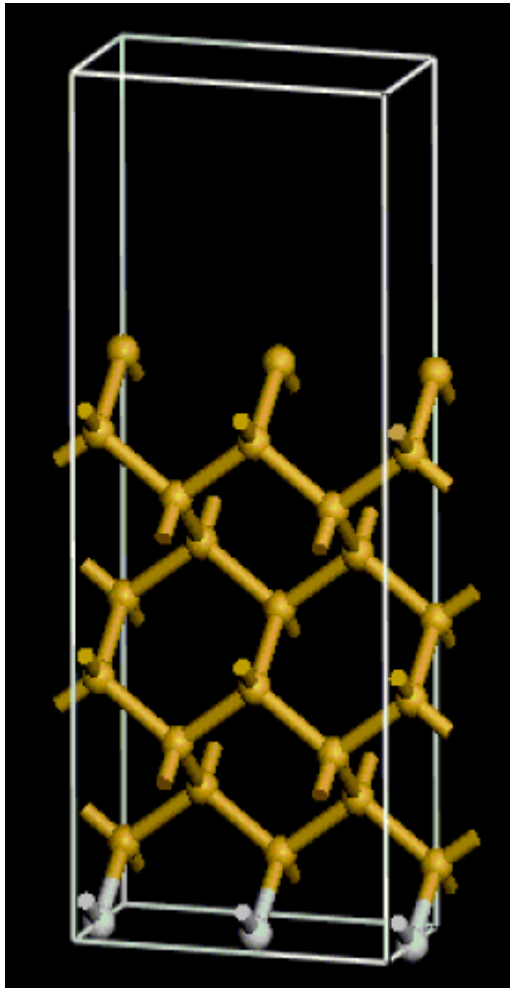
- TPSD has been shown to be significantly faster than SD
  - But ought to be a LOT slower than CG or BFGS
  - BUT for CASTEP geometry optimization we can borrow some preconditioning tricks from the BFGS routines which make it a LOT better
- And does not require line minimization
  - which is what stops BFGS from working efficiently with cell constraints (long story)
  - But not *guaranteed* to go downhill

Example: Si(100) surface reconstruction

Initial structure



Si(100) supercell with vacuum gap



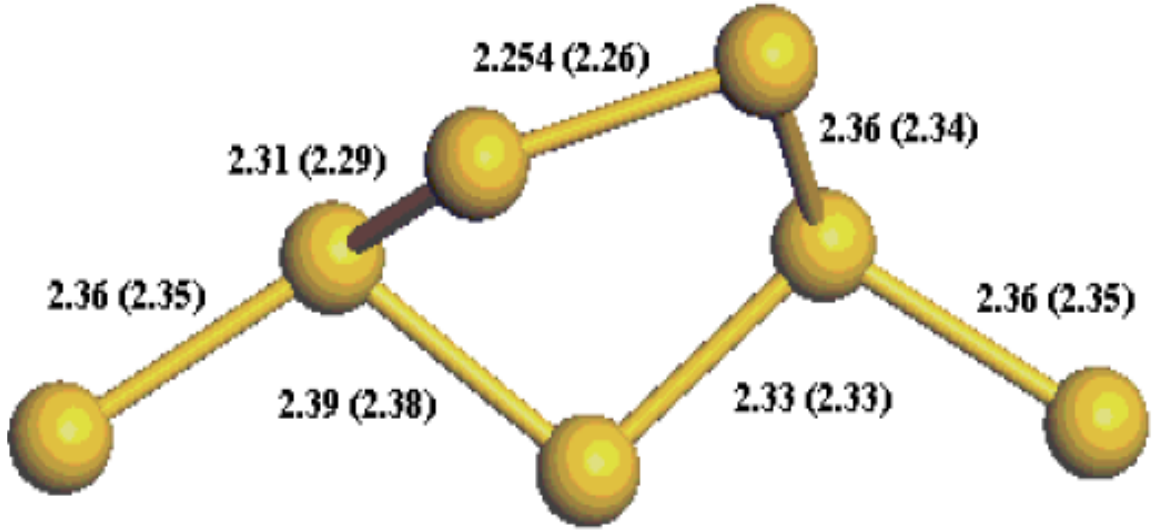
… with added hydrogen passivation

9 layers of Silicon and 7 Å vacuum

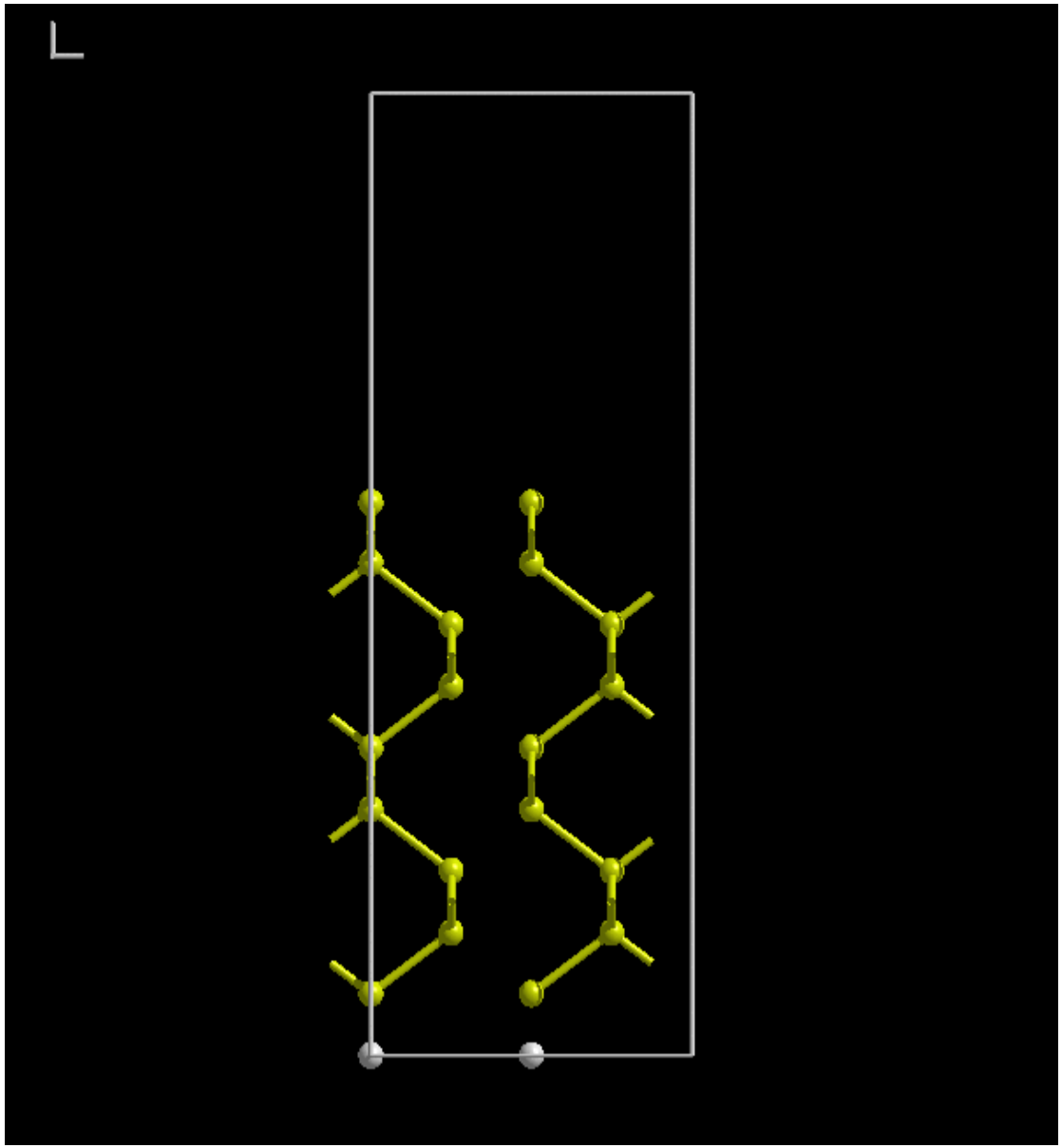
Convergence Tests

- Converge cut-off energy → 370 eV
- Converge k-point sampling → 9 k-points
- Converge number of bulk layers → 9 layers
- Converge vacuum gap → 9 Å
- only then see asymmetric dimerisation:

()=best lit. value

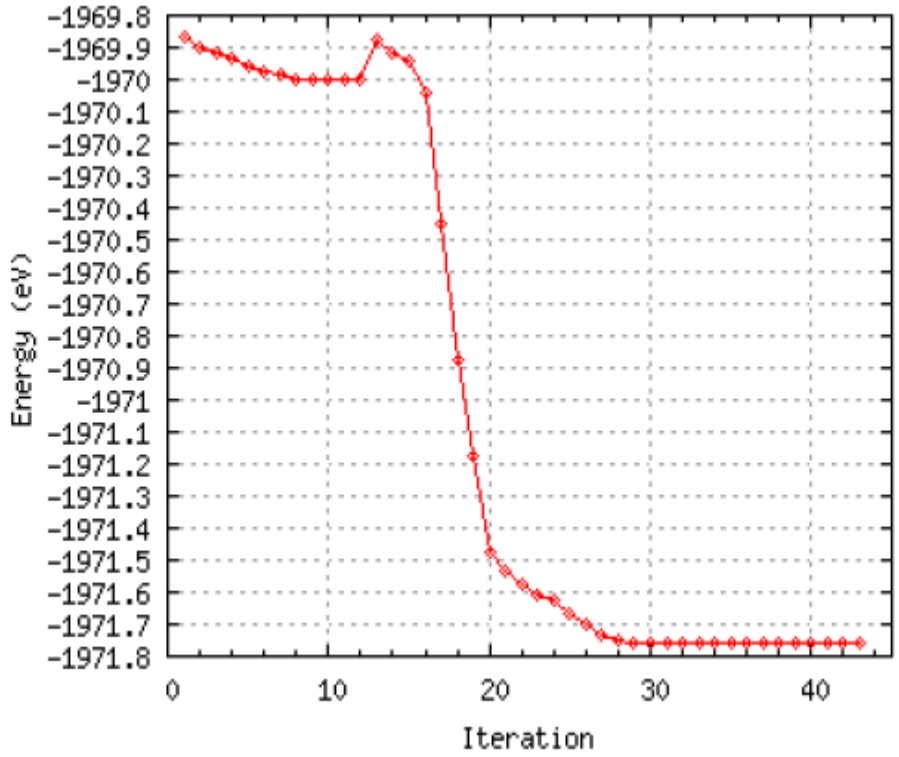


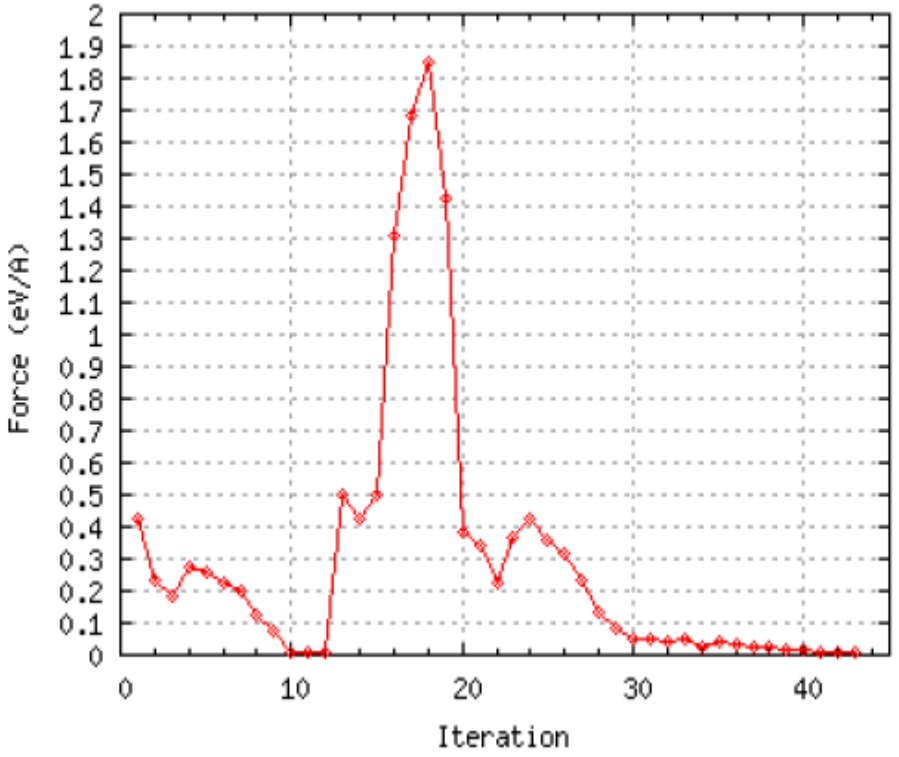
Si(100) – the movie



THE UNIVERSITY of York

Si(100) – BFGS progress





- Initial slow decrease in energy due to surface layer compression.
- Then small barrier to dimer formation overcome around iteration 14.
- Then rapid energy drop due to dimerisation.
- Final barrier to asymmetric dimerisation overcome around iteration 24.

Summary

Summary

- Newton method
  - Simple but too expensive in CASTEP as requires analytical Hessian
- Quasi-Newton methods
  - Either approximate/iterative Hessian (BFGS) or no Hessian at all (CG)
  - CASTEP uses BFGS for ions & CG electrons
  - But there are alternatives e.g. TPSD
- NB All of these ONLY do *local minimization* – *global minimization* is another lecture …

References

- MC Payne et al., Rev. Mod. Phys **64**, 1045 (1992)

- WH Press et al, "*Numerical Recipes: The Art of Scientific Computing*", Cambridge University Press (1989 – 2007)

- J Barzilai & JM Borwein, IMA J Num. Anal. **8** 141 (1988)