CECAM tutorial BIGDFT

Stefan.Goedecker@unibas.ch

The variational principle

The ground state wave function Ψ_0 is the wave-function Ψ which minimizes

$$E = \int d\mathbf{x}_1 ... \int d\mathbf{x}_N \Psi^*(\mathbf{x}_1, ..., \mathbf{x}_N) \mathcal{H} \Psi(\mathbf{x}_1, ..., \mathbf{x}_N)$$

Proof: Since the eigenvectors of a Hermitian matrix form a complete set we can expand Ψ in terms of the eigenstates Ψ_i

$$\Psi(\mathbf{x}_1, ..., \mathbf{x}_N) = \sum_i c_i \Psi_i(\mathbf{x}_1, ..., \mathbf{x}_N)$$

The normalization condition $\langle \Psi_i | \Psi_i \rangle$ implies

$$\sum_i c_i^2 = 1$$

The expectation value for the energy is given by

$$E = \sum_i E_i c_i^2$$

Evidently the minimum is obtained if $c_0 = 1$ with all other coefficients being zero. Hence $E = E_0$ and $\Psi = \Psi_0$

The Kohn-Sham equations in the LDA approximation

For simplicity we will consider a closed shell system. The LDA total energy is given by

$$E = -\sum_{i=1}^{N/2} \int \phi_i^*(\mathbf{r}) \nabla^2 \phi_i(\mathbf{r}) d\mathbf{r} + \int V_{en}(\mathbf{r}) \rho(\mathbf{r}) d\mathbf{r} + \frac{1}{2} \int \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} d\mathbf{r}' d\mathbf{r} + E_{xc}^{LDA}[\rho(\mathbf{r})] \quad (1)$$

where the charge density $\rho(\mathbf{r})$ is the sum over the square of all the occupied Kohn-Sham orbitals

$$\rho(\mathbf{r}) = 2 \sum_{i=1}^{N/2} \phi_i^*(\mathbf{r}) \phi_i(\mathbf{r}) \quad (2)$$

The first term in Eq. 1 is the kinetic energy of N independent electrons, the second the interaction of the electrons with the nuclei and potentially other external potentials, the third the classical electron-electron repulsion and the last the above described exchange correlation energy. The Kohn-Sham equations are obtained by minimizing the total energy expression Eq. 1 under the constraint that the orbitals ϕ_i are orthonormal. The procedure is analogous to the HF case. Applying the rules for functional derivatives, we obtain for the unconstrained gradient $d_i(\mathbf{r}) = \frac{1}{2} \frac{\delta E}{\delta \phi_i^*(\mathbf{r})}$

$$d_i(\mathbf{r}) = -\frac{1}{2} \nabla^2 \phi_i(\mathbf{r}) + V_{en}(\mathbf{r}) \phi_i(\mathbf{r}) + \int \frac{\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} d\mathbf{r}' \phi_i(\mathbf{r}) + v_{xc}^{LDA}(\rho(\mathbf{r})) \phi_i(\mathbf{r}) \quad (3)$$

where the exchange correlation potential is defined as

$$v_{xc}^{LDA}(\rho(\mathbf{r})) = \frac{\delta E_{xc}^{LDA}(\rho(\mathbf{r}))}{\delta\rho(\mathbf{r})}$$

Introducing the Kohn-Sham Hamiltonian $\mathcal{H}^{KS}$

$$\mathcal{H}^{KS} = -\frac{1}{2}\nabla^2 + V(\mathbf{r}) \tag{4}$$

where the Kohn-Sham potential V is the sum of the external potential, the Hartree potential and the exchange correlation potential

$$V(\mathbf{r}) = V_{en}(\mathbf{r}) + \int \frac{\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} d\mathbf{r}' + v_{xc}^{LDA}(\rho(\mathbf{r}))$$

the unconstrained gradient can simply be written as

$$d_i(\mathbf{r}) = \mathcal{H}^{KS}\phi_i(\mathbf{r}) \tag{5}$$

The orthonormality constraints can be included by using Lagrange multipliers $\Lambda_{i,j}$ and the condition that the constrained gradient vanishes becomes

$$\mathcal{H}^{KS}\phi_i(\mathbf{r}) - \sum_{j=1}^{N/2} \Lambda_{i,j}\phi_j(\mathbf{r}) = 0 \tag{6}$$

where

$$\Lambda_{i,j} = \int \phi_j^*(\mathbf{r}) d_i(\mathbf{r}) d\mathbf{r} = \int \phi_i(\mathbf{r}) d_j^*(\mathbf{r}) d\mathbf{r} = \int \phi_j^*(\mathbf{r}) \mathcal{H}^{KS} \phi_i(\mathbf{r}) d\mathbf{r} \tag{7}$$

The total energy (Eq. 1) as well as any other physical observable is invariant under unitary transformations among the occupied orbitals, i.e we can form a new set of equivalent orbitals Ψ_i^{new}

$$\Psi_i^{new} = \sum_j U_{i,j} \Psi_j$$

where U is a unitary matrix. In particular we may therefore choose canonical orbitals which diagonalize the matrix Λ in Eq. 7 and the condition that the constrained gradient vanishes (Eq. 6) results in the eigenvalue problem

$$\mathcal{H}^{KS} \phi_i(\mathbf{r}) = \varepsilon_i \phi_i(\mathbf{r}) \tag{8}$$

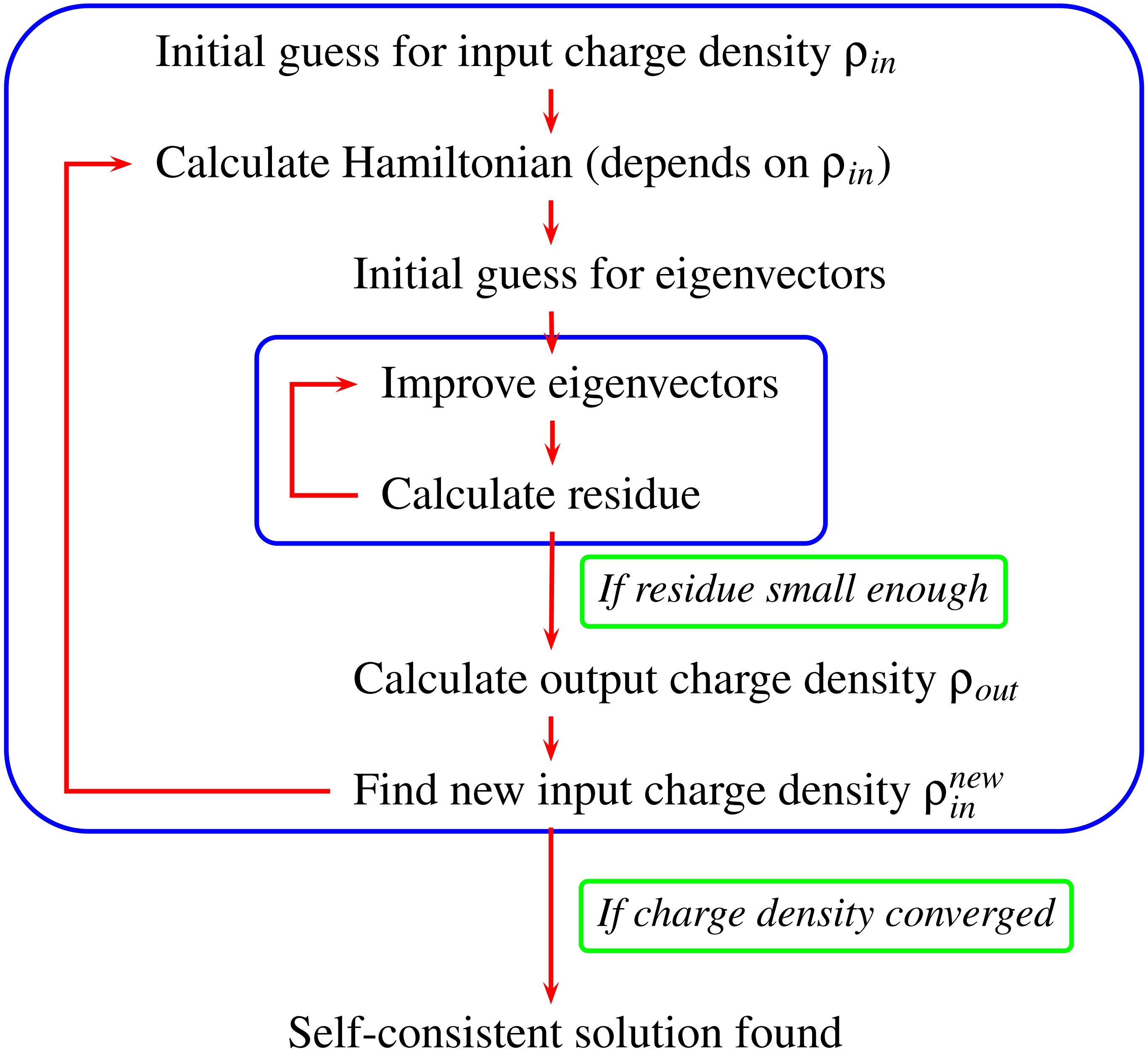
The canonical orbitals satisfying the above equation are called the Kohn-Sham orbitals and the eigenvalues are called the Kohn-Sham energies. Only the occupied orbitals enter into the energy expression of Eq. 1. The virtual levels have no physical meaning except that the virtual ε_i's give like in HF approximative electron affinities. The occupied ε_i's give also like in HF approximate ionization energies.

Basic approaches to the numerical solution of the Kohn-Sham equations

There are two possibilities to solve the Kohn-Sham equations

- Direct numerical minimization of the total energy:
  One calculates the gradient (Eq. 6) and ”goes down” along this gradient until it vanishes

- Self-consistent eigenvalue problem:
  One solves the eigenvalue problem of Eq. 8. The equation is however more difficult than a simple eigenvalue problem. Note that the Hartree and exchange correlation potentials depend on the charge density. The correct charge density which is given in terms of the solution by Eq. 2 is of course not yet available at the start of the calculation. This problem can be circumvented by self-consistency iterations. One repeats the solution of the eigenvalue problem until the output charge density calculated form the eigenorbitals is equal to the input charge density that was used for the calculation of the potential. In the resulting flowchart shown below it is also assumed that the eigenvalue problem is solved by an iterative eigenvalue problem solver. The eigenvalue iterations are indicated by the small blue inner box, the self-consistency iterations by the large blue outer box.



Initial guess for input charge density ρ_{in}
Calculate Hamiltonian (depends on ρ_{in})
Initial guess for eigenvectors
Improve eigenvectors
Calculate residue
If residue small enough
Calculate output charge density ρ_{out}
Find new input charge density ρ_{in}^{new}
If charge density converged
Self-consistent solution found

Discretization of continuous differential equations

Up to now we have used in this course the language of traditional calculus, i.e. continuous functions and differential and integral operators acting on these functions. This language is not suited for numerical work, since one can not represent a continuous function on a computer. All one can represent are vectors of numbers. For numerical processing we consequently have to discretize our equations. This is done by expanding the wave-function in terms of a finite number M of orthonormal basis functions $U_k(r)$

$$\psi_i(\mathbf{r}) = \sum_{k=1}^{M} u_i(k) U_k(\mathbf{r}) \quad ; \quad \psi_i^*(\mathbf{r}) = \sum_{k=1}^{M} u_i^*(k) U_k^*(\mathbf{r}) \tag{9}$$

This gives us an energy expression that depends on N finite vectors $u_i(k)$ of length M. Taking then the partial derivatives with respect to the degrees of freedom $u_i(k)$ of the numerical wave-function under the orthogonality constraints gives a set of discrete Euler-Lagrange equations, whose solution will provide us with the numerical wave-function that minimizes the energy within this basis set, i.e. with the numerical solution. The partial derivatives containing the orthogonality constraints can be calculated using the chain rule

$$\frac{\partial E}{\partial u_i^*(k)} = \int \frac{\delta E}{\delta \psi_i^*(\mathbf{r})} \frac{\partial \psi^*(\mathbf{r})}{\partial u_i^*(k)} d\mathbf{r} = \int \frac{\delta E}{\delta \psi_i^*(\mathbf{r})} U_k^*(\mathbf{r}) d\mathbf{r} \tag{10}$$

Performing the algebraic operations gives the discrete gradient g

$$g_i(k) = \sum_l H^{KS}(k,l)u_i(l) - \sum_j \Lambda_{i,j}u_j(k) \tag{11}$$

The above equation is the discretized analogue of equation 6. The Kohn Sham matrix is given by

$$H^{KS}(k,l) = \int d\mathbf{r} U_k^*(\mathbf{r}) \mathcal{H}^{KS} U_k^*(\mathbf{r})$$

and

$$\Lambda_{i,j} = \sum_{l,k} u_i(k) H^{KS}(k,l) u_j(l) \tag{12}$$

Eq. 11 is the central equation for numerical work. In the discrete case the solution vectors u_i are again not uniquely defined. Any other set of vectors that is related to the original set of vectors by an unitary transformation is an equally valid solution. There exists therefore a set of solution vectors that diagonalizes Λ and satisfies

$$d_j(k) - \varepsilon_j u_j(k) = 0 \tag{13}$$

Eq. 13 is the basic equation if a selfconsistent diagonalization approach is adopted whereas Eq. 11 is the basic gradient expression in a numerical minimization procedure. Once one has found the solution vectors of Eq. 11 one can transform the set of solution vectors u_i into the canonical set satisfying Eq. 13 if the canonical orbitals are desired.

Minimizing a continuous 1-dim function

Minimizing a smooth function is considerably easier than minimizing a non-smooth or even discontinuous function. If the first derivative exists, it points in the direction of the strongest increase of the function. The opposite direction consequently gives the strongest decrease of the function. The so-called steepest descent iteration

$$x_{l+1} = x_l - \alpha f'(x_l) \tag{14}$$

will therefore converge to the minimum $f(x_M)$ of the function f if the step size α is sufficiently small. If α is too large the iteration will diverge.

Next, we will discuss the case where the second derivative exists as well. Using in a combined way the information on the first and second derivative gives the most efficient minimization algorithms. The information about even higher derivatives is typically not used since this would be too complicated. Consequently we can assume in our discussion of minimization algorithms that we have to minimize a quadratic function. Then we can do a Taylor expansion of the function f and its derivative around an arbitrary point $\tilde{x}$

$$\begin{aligned} f(x) &= f(\tilde{x}) + (x-\tilde{x})\, f'(\tilde{x}) + \frac{1}{2}(x-\tilde{x})^2\, f'' \\ f'(x) &= f'(\tilde{x}) + (x-\tilde{x})\, f'' \end{aligned} \tag{15}$$

The stationary point $x = x_M$ where the derivative vanishes can easily be obtained by solving Eq. 15.

$$x_M = \tilde{x} - f'(\tilde{x})/f'' \tag{16}$$

We assume it is a minimum (i.e. $f'' > 0$) and not a maximum. Eq. 16 gives rise to the Newton iteration

$$x_{l+1} = x_l - f'(x_l)/f'' \tag{17}$$

The iteration of Eq. 17 will obviously converge in a single step for a quadratic function, but several iterations are needed for a general function. In the case of a quadratic function we did not have to worry where to evaluate the second derivative since it was a constant. This is of course not any more true for a general function. As a matter of fact we see that for the one-dimensional case we are discussing, Eq. 14 and Eq. 17 are identical if we put $\alpha = 1/f''$. Therefore one best adopts for the one-dimensional case the point of view that we just do steepest descent iterations where α is of the order of $1/f''$, but small enough to ensure convergence. In this case we do not have to answer the question where to evaluate f''.

Minimizing continuous many-dimensional functions

The basic concepts of the 1-dimensional case can be carried over into the many dimensional case. The steepest descent iteration becomes

$$\vec{x}_{l+1} = \vec{x}_l - \alpha\, \vec{g}(\vec{x}_l) \tag{18}$$

where $\vec{g}(\vec{x}) = \nabla f(\vec{x})$ is the gradient of the function f. As in the 1-dim case this will converge to a minimum if α is sufficiently small.
For a function where the second derivatives exist we can again do a Taylor expansion

$$\begin{aligned} f(\vec{x}) &= f(\tilde{\vec{x}}) + (\vec{x}-\tilde{\vec{x}})^T\, \vec{g}(\tilde{\vec{x}}) + \frac{1}{2}\,(\vec{x}-\tilde{\vec{x}})^T A\,(\vec{x}-\tilde{\vec{x}}) && (19) \\ \vec{g}(\vec{x}) &= \vec{g}(\tilde{\vec{x}}) + A(\vec{x}-\tilde{\vec{x}}) && (20) \end{aligned}$$

where A is the Hessian matrix

$$A(i,j) = \frac{\partial}{\partial x(i)} \frac{\partial}{\partial x(j)} f(\vec{x})$$

For a quadratic form the Hessian matrix would not depend on the evaluation point, for a general function it of course does and the problem where to evaluate it will be postponed.
Solving Eq. 20 for $\vec{x}$ leads to the Newton iteration

$$\vec{x}_{l+1} = \vec{x}_l - A^{-1}\, \vec{g}(\vec{x}_l) = \vec{x}_l - \vec{p}_l \tag{21}$$

Note that we have to solve in each iteration of the Newton method a linear system of equations for the preconditioned gradient vector p

$$A\vec{p} = \vec{g} \tag{22}$$

There are several basic problems with the Newton iteration:

- As mentioned before, it is not clear where to evaluate it for a non-quadratic form
- Realistic functions are not quadratic forms and so the theory is anyway only an approximation.
- The calculation of the exact Hessian matrix is numerically too expensive for complicated high-dimensional functions
- The matrix inversion of Eq. 22 is too expensive for high-dimensional functions.

Let us therefore define a slightly more general iteration that we will call preconditioned steepest descent iteration

$$\vec{x}_{l+1} = \vec{x}_l - P\,\vec{g}(\vec{x}_l) \tag{23}$$

where P is a still unspecified preconditioning matrix. Evidently we get the steepest descent iteration of Eq. 18 if we put $P = \alpha I$ and we get the Newton iteration of Eq. 21 if we put $P = A^{-1}$

Convergence analysis of the steepest descent iteration

For the convergence analysis we will again assume that we are already sufficiently close to the minimum, so that the function is a quadratic form. Because by definition the gradient vanishes at x_M the Taylor expansion of Eq. 19 becomes

$$f(\vec{x}) - f(\vec{x}_M) = \frac{1}{2}\,(\vec{x}-\vec{x}_M)^T\,A\,(\vec{x}-\vec{x}_M)$$

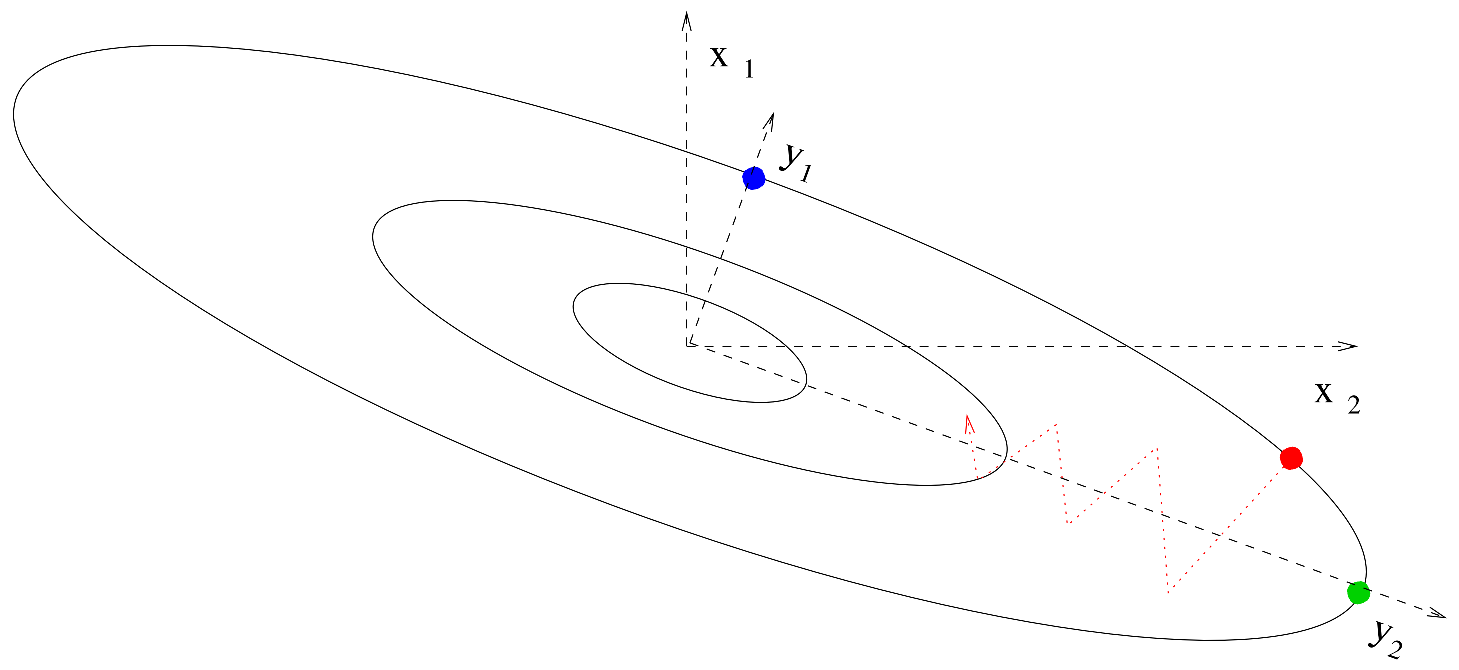
By shifting the origin (such that $\vec{x}_M = 0$) and the function (such that $f(\vec{x}_M) = 0$) we can without any restriction consider the simpler case

$$f(\vec{x}) = \frac{1}{2}\,\vec{x}^T\,A\,\vec{x} \tag{24}$$

Since the Hessian A is a positive definite symmetric matrix, we can go into an coordinate system $\vec{y}$, that is obtained by applying a unitary transformation U on the original coordinate system $\vec{x}$, where A becomes a positive real diagonal matrix $D = U^T AU$.

$$f(\vec{y}) = \frac{1}{2}\sum_k D(k,k)\,y(k)^2 \qquad ; \qquad g(k) = D(k,k)\,y(k) \tag{25}$$

Things are illustrated in the figure below. The ellipsoids represent the equipotential lines of the function f. The axis of the y coordinate system coincide with the principal axis of the ellipsoids.



Let us now assume that at a certain stage of a steepest descent iteration the current point $\vec{x}_l$ coincides with the blue dot. Since in this case the gradient points exactly in the direction of the minimum, we can find the minimum of this one-dimensional subproblem with a single steepest descent step if we chose $\alpha = 1/D(1,1)$. If we are at the green dot the same arguments apply except that now $\alpha = 1/D(2,2)$. In general our current iterations points are not located on any principle axis. The gradient of an arbitrary point such as the red dot has components of both principal axis. In order to guarantee convergence we have to be conservative and to choose $\alpha = 1/max[D(1,1), D(2,2)]$. Since the components of the gradient that correspond to principal axis with small eigenvalues will be damped too strongly, a steepest descent iteration in more than two dimensions is approaching the minimum very slowly by a large number of zigzag moves.

The generalization to more than 2 dimensions is obvious. The α of a steepest descent iteration has to be taken to be the reciprocal of the largest eigenvalue of the Hessian. Let us now examine the convergence rate for the multi-dimensional case in a more mathematical way. Since the steepest descent iteration is invariant under unitary transformations of the coordinate system we can without restriction consider a diagonal Hessian. The steepest descent iteration then becomes

$$x_{l+1}(k) = x_l(k) - \alpha d(k) x_l(k)$$

where $d(k)$ is the vector containing the diagonal elements of the diagonal matrix A. Hence

$$x_{l+1}(k) = x_1(k)(1 - \alpha d(k))^l$$

where $\vec{x}_1$ is the starting vector for the iteration. Convergence can only be obtained if $|1 - \alpha d(k)| < 1$. Hence α can be at most twice of the reciprocal of the largest eigenvalue. So let us put

$$\alpha = t / d_{max} \tag{26}$$

where t is in between 0 and 2. For $t = 1$, the component k that will converge most slowly is the one associated to the smallest eigenvalue. Requiring this component to be equal to a certain precision p gives

$$(1 - t \frac{d_{min}}{d_{max}})^l = p$$

The number of iterations l necessary to obtain this precision p is then given by

$$l = \ln(p)/\ln(1-t\frac{d_{min}}{d_{max}}) \tag{27}$$

If $\frac{d_{min}}{d_{max}}$ is small, this is asymptotically equal to

$$l = -\ln(p)\frac{d_{max}}{t\,d_{min}} \propto \kappa \tag{28}$$

The ratio between the largest and the smallest eigenvalue of the Hessian matrix is called the condition number $\kappa = \frac{d_{max}}{d_{min}}$. We have thus the result that the number of iterations is proportional to the condition number κ in the steepest descent method. This is a big problem. As we will see the conditioning number is typically growing rapidly with respect to the size of the physical system represented by the matrix. Hence the number of iterations is growing substantially as well.

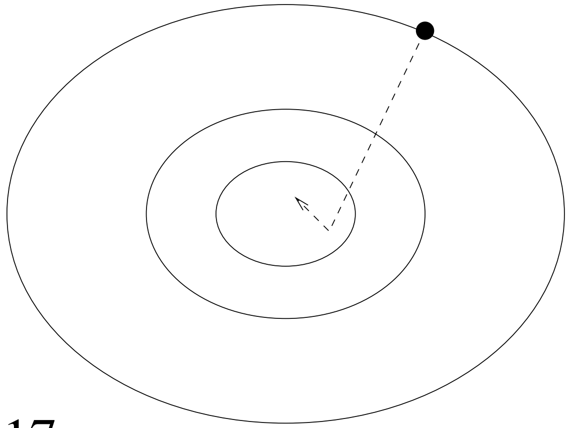
Convergence analysis of the preconditioned steepest descent iteration

The convergence analysis of the preconditioned steepest descent iteration of Eq. 23 is analogous to the one for the simple steepest descent iteration. The only difference is that we perform the analysis in a coordinate system that diagonalizes *PA* instead of *A*. The number of iterations is consequently given by the same formula

$$l = -\ln(p)\frac{d_{max}}{t\, d_{min}}$$

the only difference being that d_{max} and d_{min} are now the largest and smallest eigenvalues of *PA*. If the conditioning number of *PA* is smaller than of *A*, the number of iterations of the preconditioned steepest descent method will be reduced compared to the simple steepest descent method. A good preconditioning matrix is a compromise between 2 requirements. On the one hand it should give a small condition number, on the other hand it should be easy to calculate and to apply to the gradient. A frequent choice for *P* is a diagonal or sparse matrix.

Topology of preconditioned problem:



Steepest descent with line minimization

The prescription for a steepest descent iteration with line minimization for a function f is formally identical to an ordinary steepest descent minimization

$$\vec{x}_{l+1} = \vec{x}_l - \alpha \vec{g}$$

The difference is that α is not fixed, but optimized such that

$$\frac{\partial}{\partial \alpha} f(\vec{x} + \alpha \vec{g}) = 0$$

The line minimization ensures that the function will decrease at each iteration point. This does however not imply that one comes as close as possible to the minimum. As a matter of fact it turns out that with an optimal value of t (Eq.26) the convergence is as fast as with line minimization. In addition one iteration is much cheaper without the line minimization. The conclusion is that one should avoid line minimizations unless one can not at all estimate the largest eigenvalue of the Hessian matrix. If this estimation is possible steepest descent with some feedback is a recommendable strategy.

Steepest descent with energy feedback

A simple and powerful modification of the simple steepest descent method is the steepest descent with energy feedback. Assuming that the functional value represents the energy, we decrease the step size α if the energy rises in an iteration, otherwise we increase it. Since we know that in the case of an energy increase the parameter t (Eq.26) is roughly twice as large as would be optimal for the elimination of the stiff components, associated to large eigenvalues of the Hessian, we decrease α by a factor of $1/2$. If the energy goes down, as it should, we slightly increase α (e.g. by a factor of 1.05) to speed up the convergence.

The DIIS (Direct Inversion in Iterative Subspace) minimization method

Be $\vec{c}_0$ the exact solution of a quadratic minimization problem and $\vec{c}_i$ (i=1, ..,m) a set of m approximate solution vectors. Their error vectors are defined by

$$\vec{e}_m = \vec{c}_m - \vec{c}_0$$

We form a new vector $\tilde{\vec{c}}_m$

$$\tilde{\vec{c}}_m = \sum_{i=1}^{m} d_i \vec{c}_i$$

If $\tilde{\vec{c}}_m$ was the exact solution, it would fulfill

$$\begin{aligned}
\sum_{i=1}^{m} d_i \vec{c}_i &= \vec{c}_0 \\
\sum_{i=1}^{m} d_i (\vec{c}_0 + \vec{e}_i) &= \vec{c}_0 \\
\sum_{i=1}^{m} d_i \vec{c}_0 + \sum_{i=1}^{m} d_i \vec{e}_i &= \vec{c}_0
\end{aligned}$$

Satisfied if

$$\sum_{i=1}^{m} d_i = 1 \qquad ; \qquad \sum_{i=1}^{m} d_i \vec{e}_i = 0$$

Last condition can only be fulfilled approximately, leading to the minimization problem

$$\min \left[\langle \sum_{i=1}^{m} d_i e_i | \sum_{i=1}^{m} d_i e_i \rangle \right]$$

under the constraint $\sum_{i=1}^{m} d_i = 1$. This leads to the system of equations

$$\begin{pmatrix} \langle e_1|e_1\rangle & \langle e_1|e_2\rangle & ... & \langle e_1|e_m\rangle & 1 \\ \langle e_2|e_1\rangle & \langle e_2|e_2\rangle & ... & \langle e_2|e_m\rangle & 1 \\ ... & ... & ... & ... & . \\ \langle e_m|e_1\rangle & \langle e_m|e_2\rangle & ... & \langle e_m|e_m\rangle & 1 \\ 1 & 1 & ... & 1 & 0 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ . \\ d_m \\ d_{m+1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

In practice the error vectors are approximated by $e_i = P g_i$
The new vector is then given by

$$\vec{g}_{m+1} = \nabla f(\tilde{\vec{c}}_m)$$

$$\vec{c}_{m+1} = \tilde{\vec{c}}_m + \alpha P \vec{g}_{m+1}$$

Preconditioning Schroedinger's equation

Preconditioning is essential for the solution of the Kohn Sham equations in any systematic basis set. Without preconditioning the condition number would be huge. The large eigenvalues of the Hessian of the Kohn Sham minimization problem are due to plane wave like eigenfunctions with high kinetic energy. A good preconditioner for the Kohn Sham minimization is therefore the inverse of the shifted kinetic energy operator

$$\frac{1}{2}\nabla^2 + C$$

The constant C has to be added to remove the singularity in the inverse. In practice we do not calculate the inverse but we obtain the preconditioned gradient **p** from the gradient **g** by solving the linear system of equations

$$\left(\frac{1}{2}\nabla^2 + C\right)\mathbf{p} = \mathbf{g}$$

This linear system is generally solved by an iterative method such as the conjugate gradient method. The main numerical cost in iterative methods is the repeated application of the matrix onto a vector. In a wavelet basis the kinetic energy operator is a sparse matrix and the matrix vector products can rapidly be calculated.