Optimization and gradient methods

Morten Hjorth-Jensen Email morten.hjorth-jensen@fys.uio.no[1]

Department of Physics and Center fo Computing in Science Education, University of Oslo, Oslo, Norway[1]

February 21, 2025

Plans for the week of February 17-21, 2025

- Reminder on Fokker-Planck equation and Langevin equations (partly from last week)
- Programming elements of correlation function
- Start optimization: Expressions for derivatives as functions of the variational parameters
- Newton's method, gradient descent, Steepest descent and Conjugate Gradient Descent

Teaching Material, videos and written material

- These lecture notes
- Video on the Conjugate Gradient methods
- Recommended background literature, Convex Optimization by Boyd and Vandenberghe. Their lecture slides are very useful (warning, these are some 300 pages).

Importance sampling, Fokker-Planck and Langevin equations

If we assume a discrete set of events, our initial probability distribution function can be given by

$$w_i(0) = \delta_{i,0},$$

and its time-development after a given time step $\Delta t = \epsilon$ is

$$w_i(t) = \sum_j W(j \to i) w_j(t = 0).$$

The continuous analog to $w_i(0)$ is

$$w(\mathbf{x}) \to \delta(\mathbf{x}),$$

where we now have generalized the one-dimensional position x to a generic-dimensional vector $\mathbf{x}$. The Kroenecker δ function is replaced by the δ distribution function $\delta(\mathbf{x})$ at $t = 0$.

Importance sampling, Fokker-Planck and Langevin equations

The transition from a state j to a state i is now replaced by a transition to a state with position y from a state with position x. The discrete sum of transition probabilities can then be replaced by an integral and we obtain the new distribution at a time $t + \Delta t$ as

$$w(\mathrm{y}, t + \Delta t) = \int W(\mathrm{y}, t + \Delta t | \mathrm{x}, t) w(\mathrm{x}, t) d\mathrm{x},$$

and after m time steps we have

$$w(\mathrm{y}, t + m\Delta t) = \int W(\mathrm{y}, t + m\Delta t | \mathrm{x}, t) w(\mathrm{x}, t) d\mathrm{x}.$$

When equilibrium is reached we have

$$w(\mathrm{y}) = \int W(\mathrm{y} | \mathrm{x}, t) w(\mathrm{x}) d\mathrm{x},$$

that is no time-dependence. Note our change of notation for W

Importance sampling, Fokker-Planck and Langevin equations

We can solve the equation for $w(\mathrm{y}, t)$ by making a Fourier transform to momentum space. The PDF $w(\mathrm{x}, t)$ is related to its Fourier transform $\tilde{w}(\mathrm{k}, t)$ through

$$w(\mathrm{x}, t) = \int_{-\infty}^{\infty} d\mathrm{k} \exp(i\mathrm{kx}) \tilde{w}(\mathrm{k}, t),$$

and using the definition of the δ-function

$$\delta(\mathrm{x}) = \frac{1}{2\pi} \int_{-\infty}^{\infty} d\mathrm{k} \exp(i\mathrm{kx}),$$

we see that

$$\tilde{w}(\mathrm{k}, 0) = 1/2\pi.$$

Importance sampling, Fokker-Planck and Langevin equations

We can then use the Fourier-transformed diffusion equation

$$\frac{\partial \tilde{w}(\mathrm{k},t)}{\partial t} = -D\mathrm{k}^2\tilde{w}(\mathrm{k},t),$$

with the obvious solution

$$\tilde{w}(\mathrm{k},t) = \tilde{w}(\mathrm{k},0)\exp\left[-(D\mathrm{k}^2 t)\right) = \frac{1}{2\pi}\exp\left[-(D\mathrm{k}^2 t)\right].$$

Importance sampling, Fokker-Planck and Langevin equations

With the Fourier transform we obtain

$$w(x,t)=\int_{-\infty}^{\infty} dk \exp\left[ikx\right]\frac{1}{2\pi}\exp\left[-(Dk^2t)\right]=\frac{1}{\sqrt{4\pi Dt}}\exp\left[-(x^2/4Dt)\right],$$

with the normalization condition

$$\int_{-\infty}^{\infty} w(x,t)dx=1.$$

Importance sampling, Fokker-Planck and Langevin equations

The solution represents the probability of finding our random walker at position x at time t if the initial distribution was placed at $x = 0$ at $t = 0$.

There is another interesting feature worth observing. The discrete transition probability W itself is given by a binomial distribution. The results from the central limit theorem state that transition probability in the limit $n \rightarrow \infty$ converges to the normal distribution. It is then possible to show that

$$W(il-jl, n\epsilon) \rightarrow W(y, t+\Delta t|x, t) = \frac{1}{\sqrt{4\pi D\Delta t}} \exp\left[-((y-x)^2/4D\Delta t)\right],$$

and that it satisfies the normalization condition and is itself a solution to the diffusion equation.

Importance sampling, Fokker-Planck and Langevin equations

Let us now assume that we have three PDFs for times $t_0 < t' < t$, that is $w(\mathrm{x}_0, t_0)$, $w(\mathrm{x}', t')$ and $w(\mathrm{x}, t)$. We have then

$$w(\mathrm{x}, t) = \int_{-\infty}^{\infty} W(\mathrm{x}.t|\mathrm{x}'.t')w(\mathrm{x}', t')d\mathrm{x}',$$

and

$$w(\mathrm{x}, t) = \int_{-\infty}^{\infty} W(\mathrm{x}.t|\mathrm{x}_0.t_0)w(\mathrm{x}_0, t_0)d\mathrm{x}_0,$$

and

$$w(\mathrm{x}', t') = \int_{-\infty}^{\infty} W(\mathrm{x}'.t'|\mathrm{x}_0, t_0)w(\mathrm{x}_0, t_0)d\mathrm{x}_0.$$

Importance sampling, Fokker-Planck and Langevin equations

We can combine these equations and arrive at the famous Einstein-Smoluchenski-Kolmogorov-Chapman (ESKC) relation

$$W(\mathrm{x}t|\mathrm{x}_0t_0) = \int_{-\infty}^{\infty} W(\mathrm{x},t|\mathrm{x}',t')W(\mathrm{x}',t'|\mathrm{x}_0,t_0)d\mathrm{x}'.$$

We can replace the spatial dependence with a dependence upon say the velocity (or momentum), that is we have

$$W(\mathrm{v},t|\mathrm{v}_0,t_0) = \int_{-\infty}^{\infty} W(\mathrm{v},t|\mathrm{v}',t')W(\mathrm{v}',t'|\mathrm{v}_0,t_0)d\mathrm{x}'.$$

Importance sampling, Fokker-Planck and Langevin equations

We will now derive the Fokker-Planck equation. We start from the ESKC equation

$$W(x, t|x_0, t_0) = \int_{-\infty}^{\infty} W(x, t|x', t')W(x', t'|x_0, t_0)dx'.$$

Define $s = t' - t_0$, $\tau = t - t'$ and $t - t_0 = s + \tau$. We have then

$$W(x, s + \tau|x_0) = \int_{-\infty}^{\infty} W(x, \tau|x')W(x', s|x_0)dx'.$$

Importance sampling, Fokker-Planck and Langevin equations

Assume now that τ is very small so that we can make an expansion in terms of a small step xi, with $\mathrm{x}' = \mathrm{x} - \xi$, that is

$$W(\mathrm{x}, s|\mathrm{x}_0) + \frac{\partial W}{\partial s}\tau + O(\tau^2) = \int_{-\infty}^{\infty} W(\mathrm{x}, \tau|\mathrm{x} - \xi) W(\mathrm{x} - \xi, s|\mathrm{x}_0) d\mathrm{x}'.$$

We assume that $W(\mathrm{x}, \tau|\mathrm{x} - \xi)$ takes non-negligible values only when ξ is small. This is just another way of stating the Master equation!!

Importance sampling, Fokker-Planck and Langevin equations

We say thus that x changes only by a small amount in the time interval τ. This means that we can make a Taylor expansion in terms of ξ, that is we expand

$$W(\mathrm{x},\tau|\mathrm{x}-\xi)W(\mathrm{x}-\xi,s|\mathrm{x}_0) = \sum_{n=0}^{\infty}\frac{(-\xi)^n}{n!}\frac{\partial^n}{\partial x^n}\left[W(\mathrm{x}+\xi,\tau|\mathrm{x})W(\mathrm{x},s|\mathrm{x}_0)\right].$$

Importance sampling, Fokker-Planck and Langevin equations

We can then rewrite the ESKC equation as

$$\frac{\partial W}{\partial s}\tau = -W(\mathrm{x}, s|\mathrm{x}_0) + \sum_{n=0}^{\infty} \frac{(-\xi)^n}{n!} \frac{\partial^n}{\partial x^n} \left[W(\mathrm{x}, s|\mathrm{x}_0) \int_{-\infty}^{\infty} \xi^n W(\mathrm{x}+ \ \xi, \tau|\mathrm{x}) d$$

We have neglected higher powers of τ and have used that for $n = 0$ we get simply $W(\mathrm{x}, s|\mathrm{x}_0)$ due to normalization.

Importance sampling, Fokker-Planck and Langevin equations

We say thus that x changes only by a small amount in the time interval τ. This means that we can make a Taylor expansion in terms of ξ, that is we expand

$$W(\mathrm{x},\tau|\mathrm{x}-\xi)W(\mathrm{x}-\xi,s|\mathrm{x}_0) = \sum_{n=0}^{\infty}\frac{(-\xi)^n}{n!}\frac{\partial^n}{\partial x^n}\left[W(\mathrm{x}+\xi,\tau|\mathrm{x})W(\mathrm{x},s|\mathrm{x}_0)\right].$$

Importance sampling, Fokker-Planck and Langevin equations

We can then rewrite the ESKC equation as

$$\frac{\partial W(\mathrm{x}, s|\mathrm{x}_0)}{\partial s}\tau = -W(\mathrm{x}, s|\mathrm{x}_0) + \sum_{n=0}^{\infty}\frac{(-\xi)^n}{n!}\frac{\partial^n}{\partial x^n}\left[W(\mathrm{x}, s|\mathrm{x}_0)\int_{-\infty}^{\infty}\xi^n W(\mathrm{x} +\right.$$

We have neglected higher powers of τ and have used that for $n=0$ we get simply $W(\mathrm{x}, s|\mathrm{x}_0)$ due to normalization.

Importance sampling, Fokker-Planck and Langevin equations

We simplify the above by introducing the moments

$$M_n = \frac{1}{\tau}\int_{-\infty}^{\infty} \xi^n W(\mathrm{x}+\xi,\tau|\mathrm{x})d\xi = \frac{\langle[\Delta x(\tau)]^n\rangle}{\tau},$$

resulting in

$$\frac{\partial W(\mathrm{x},s|\mathrm{x}_0)}{\partial s} = \sum_{n=1}^{\infty}\frac{(-\xi)^n}{n!}\frac{\partial^n}{\partial x^n}\left[W(\mathrm{x},s|\mathrm{x}_0)M_n\right].$$

Importance sampling, Fokker-Planck and Langevin equations

When $\tau \to 0$ we assume that $\langle [\Delta x(\tau)]^n \rangle \to 0$ more rapidly than τ itself if $n > 2$. When τ is much larger than the standard correlation time of system then M_n for $n > 2$ can normally be neglected. This means that fluctuations become negligible at large time scales.
If we neglect such terms we can rewrite the ESKC equation as

$$\frac{\partial W(\mathrm{x}, s|\mathrm{x}_0)}{\partial s} = -\frac{\partial M_1 W(\mathrm{x}, s|\mathrm{x}_0)}{\partial x} + \frac{1}{2}\frac{\partial^2 M_2 W(\mathrm{x}, s|\mathrm{x}_0)}{\partial x^2}.$$

Importance sampling, Fokker-Planck and Langevin equations

In a more compact form we have

$$\frac{\partial W}{\partial s} = -\frac{\partial M_1 W}{\partial x} + \frac{1}{2}\frac{\partial^2 M_2 W}{\partial x^2},$$

which is the Fokker-Planck equation! It is trivial to replace position with velocity (momentum).

Langevin equation

Consider a particle suspended in a liquid. On its path through the liquid it will continuously collide with the liquid molecules. Because on average the particle will collide more often on the front side than on the back side, it will experience a systematic force proportional with its velocity, and directed opposite to its velocity. Besides this systematic force the particle will experience a stochastic force $\mathsf{F}(t)$. The equations of motion are

- $\frac{d\mathsf{r}}{dt} = \mathsf{v}$ and
- $\frac{d\mathsf{v}}{dt} = -\xi\mathsf{v} + \mathsf{F}$.

Langevin equation

From hydrodynamics we know that the friction constant ξ is given by

$$\xi = 6\pi\eta a/m$$

where η is the viscosity of the solvent and a is the radius of the particle .
Solving the second equation in the previous slide we get

$$\mathrm{v}(t) = \mathrm{v}_0 e^{-\xi t} + \int_0^t d\tau e^{-\xi(t-\tau)}\mathrm{F}(\tau).$$

Langevin equation

If we want to get some useful information out of this, we have to average over all possible realizations of $\mathbf{F}(t)$, with the initial velocity as a condition. A useful quantity for example is

$$\langle \mathbf{v}(t)\cdot\mathbf{v}(t)\rangle_{\mathbf{v}_0} = v_0^{-\xi 2t} + 2\int_0^t d\tau e^{-\xi(2t-\tau)}\mathbf{v}_0\cdot\langle\mathbf{F}(\tau)\rangle_{\mathbf{v}_0}$$

$$+\int_0^t d\tau'\int_0^t d\tau e^{-\xi(2t-\tau-\tau')}\langle\mathbf{F}(\tau)\cdot\mathbf{F}(\tau')\rangle_{\mathbf{v}_0}.$$

Importance sampling, Fokker-Planck and Langevin equations

Langevin equation

In order to continue we have to make some assumptions about the conditional averages of the stochastic forces. In view of the chaotic character of the stochastic forces the following assumptions seem to be appropriate

$$\langle \mathrm{F}(t) \rangle = 0,$$

and

$$\langle \mathrm{F}(t) \cdot \mathrm{F}(t') \rangle_{\mathrm{v}_0} = C_{\mathrm{v}_0} \delta(t - t').$$

We omit the subscript v_0, when the quantity of interest turns out to be independent of v_0. Using the last three equations we get

$$\langle \mathrm{v}(t) \cdot \mathrm{v}(t) \rangle_{\mathrm{v}_0} = v_0^2 e^{-2\xi t} + \frac{C_{\mathrm{v}_0}}{2\xi}(1 - e^{-2\xi t}).$$

For large t this should be equal to 3kT/m, from which it follows that

$$\langle \mathrm{F}(t) \cdot \mathrm{F}(t') \rangle = 6 \frac{kT}{m} \xi \delta(t - t').$$

Importance sampling, Fokker-Planck and Langevin equations

Langevin equation

Integrating

$$\mathrm{v}(t) = \mathrm{v}_0 e^{-\xi t} + \int_0^t d\tau e^{-\xi(t-\tau)} \mathrm{F}(\tau),$$

we get

$$\mathrm{r}(t) = \mathrm{r}_0 + \mathrm{v}_0 \frac{1}{\xi}(1 - e^{-\xi t}) + \int_0^t d\tau \int_0^\tau \tau' e^{-\xi(\tau-\tau')} \mathrm{F}(\tau'),$$

from which we calculate the mean square displacement

$$\langle (\mathrm{r}(t) - \mathrm{r}_0)^2 \rangle_{\mathrm{v}_0} = \frac{v_0^2}{\xi}(1 - e^{-\xi t})^2 + \frac{3kT}{m\xi^2}(2\xi t - 3 + 4e^{-\xi t} - e^{-2\xi t}).$$

Importance sampling, Fokker-Planck and Langevin equations

Langevin equation

For very large t this becomes

$$\langle (\mathrm{r}(t) - \mathrm{r}_0)^2 \rangle = \frac{6kT}{m\xi} t$$

from which we get the Einstein relation

$$D = \frac{kT}{m\xi}$$

where we have used $\langle (\mathrm{r}(t) - \mathrm{r}_0)^2 \rangle = 6Dt$.

Importance sampling, programming elements

The general derivative formula of the Jastrow factor (or the ansatz for the correlated part of the wave function) is (the subscript C stands for Correlation)

$$\frac{1}{\Psi_C}\frac{\partial \Psi_C}{\partial x_k} = \sum_{i=1}^{k-1}\frac{\partial g_{ik}}{\partial x_k} + \sum_{i=k+1}^{N}\frac{\partial g_{ki}}{\partial x_k}$$

However, with our written in way which can be reused later as

$$\Psi_C = \prod_{i<j} g(r_{ij}) = \exp\left\{\sum_{i<j} f(r_{ij})\right\},$$

the gradient needed for the quantum force and local energy is easy to compute. The function $f(r_{ij})$ will depends on the system under study. In the equations below we will keep this general form.

Importance sampling, program elements

In the Metropolis/Hasting algorithm, the *acceptance ratio* determines the probability for a particle to be accepted at a new position. The ratio of the trial wave functions evaluated at the new and current positions is given by (OB for the onebody part)

$$R \equiv \frac{\Psi_T^{new}}{\Psi_T^{old}} = \frac{\Psi_{OB}^{new}}{\Psi_{OB}^{old}}\frac{\Psi_C^{new}}{\Psi_C^{old}}$$

Here Ψ_{OB} is our onebody part (Slater determinant or product of boson single-particle states) while Ψ_C is our correlation function, or Jastrow factor. We need to optimize the $\nabla\Psi_T/\Psi_T$ ratio and the second derivative as well, that is the $\nabla^2\Psi_T/\Psi_T$ ratio. The first is needed when we compute the so-called quantum force in importance sampling. The second is needed when we compute the kinetic energy term of the local energy.

$$\frac{\nabla\Psi}{\Psi} = \frac{\nabla(\Psi_{OB}\,\Psi_C)}{\Psi_{OB}\,\Psi_C} = \frac{\Psi_C\nabla\Psi_{OB} + \Psi_{OB}\nabla\Psi_C}{\Psi_{OB}\Psi_C} = \frac{\nabla\Psi_{OB}}{\Psi_{OB}} + \frac{\nabla\Psi_C}{\Psi_C}$$

Importance sampling

The expectation value of the kinetic energy expressed in scaled units for particle i is

$$\langle \hat{K}_i \rangle = -\frac{1}{2}\frac{\langle\Psi|\nabla_i^2|\Psi\rangle}{\langle\Psi|\Psi\rangle},$$

$$\hat{K}_i = -\frac{1}{2}\frac{\nabla_i^2\Psi}{\Psi}.$$

Importance sampling

The second derivative which enters the definition of the local energy is

$$\frac{\nabla^2\Psi}{\Psi} = \frac{\nabla^2\Psi_{OB}}{\Psi_{OB}} + \frac{\nabla^2\Psi_C}{\Psi_C} + 2\frac{\nabla\Psi_{OB}}{\Psi_{OB}} \cdot \frac{\nabla\Psi_C}{\Psi_C}$$

We discuss here how to calculate these quantities in an optimal way.

Importance sampling

We have defined the correlated function as

$$\Psi_C = \prod_{i<j} g(r_{ij}) = \prod_{i<j}^{N} g(r_{ij}) = \prod_{i=1}^{N} \prod_{j=i+1}^{N} g(r_{ij}),$$

with $r_{ij} = |\mathrm{r}_i - \mathrm{r}_j| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$ in three dimensions or $r_{ij} = |\mathrm{r}_i - \mathrm{r}_j| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ if we work with two-dimensional systems.
In our particular case we have

$$\Psi_C = \prod_{i<j} g(r_{ij}) = \exp\left\{\sum_{i<j} f(r_{ij})\right\}.$$

Importance sampling

The total number of different relative distances r_{ij} is $N(N-1)/2$. In a matrix storage format, the relative distances form a strictly upper triangular matrix

$$
\mathrm{r} \equiv \begin{pmatrix} 0 & r_{1,2} & r_{1,3} & \cdots & r_{1,N} \\ \vdots & 0 & r_{2,3} & \cdots & r_{2,N} \\ \vdots & \vdots & 0 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & r_{N-1,N} \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}.
$$

This applies to $\mathrm{g} = \mathrm{g}(r_{ij})$ as well.

In our algorithm we will move one particle at the time, say the *kth*-particle. This sampling will be seen to be particularly efficient when we are going to compute a Slater determinant.

Importance sampling

We have that the ratio between Jastrow factors R_C is given by

$$R_C = \frac{\Psi_C^{\mathrm{new}}}{\Psi_C^{\mathrm{cur}}} = \prod_{i=1}^{k-1} \frac{g_{ik}^{\mathrm{new}}}{g_{ik}^{\mathrm{cur}}} \prod_{i=k+1}^{N} \frac{g_{ki}^{\mathrm{new}}}{g_{ki}^{\mathrm{cur}}}.$$

For the Pade-Jastrow form

$$R_C = \frac{\Psi_C^{\mathrm{new}}}{\Psi_C^{\mathrm{cur}}} = \frac{\exp U_{new}}{\exp U_{cur}} = \exp \Delta U,$$

where

$$\Delta U = \sum_{i=1}^{k-1} \left(f_{ik}^{\mathrm{new}} - f_{ik}^{\mathrm{cur}} \right) + \sum_{i=k+1}^{N} \left(f_{ki}^{\mathrm{new}} - f_{ki}^{\mathrm{cur}} \right)$$

Importance sampling

One needs to develop a special algorithm that runs only through the elements of the upper triangular matrix g and have k as an index. The expression to be derived in the following is of interest when computing the quantum force and the kinetic energy. It has the form

$$\frac{\nabla_i \Psi_C}{\Psi_C} = \frac{1}{\Psi_C}\frac{\partial \Psi_C}{\partial x_i},$$

for all dimensions and with i running over all particles.

Importance sampling

For the first derivative only $N-1$ terms survive the ratio because the g-terms that are not differentiated cancel with their corresponding ones in the denominator. Then,

$$\frac{1}{\Psi_C}\frac{\partial \Psi_C}{\partial x_k} = \sum_{i=1}^{k-1}\frac{1}{g_{ik}}\frac{\partial g_{ik}}{\partial x_k} + \sum_{i=k+1}^{N}\frac{1}{g_{ki}}\frac{\partial g_{ki}}{\partial x_k}.$$

An equivalent equation is obtained for the exponential form after replacing g_{ij} by $\exp(f_{ij})$, yielding:

$$\frac{1}{\Psi_C}\frac{\partial \Psi_C}{\partial x_k} = \sum_{i=1}^{k-1}\frac{\partial g_{ik}}{\partial x_k} + \sum_{i=k+1}^{N}\frac{\partial g_{ki}}{\partial x_k},$$

with both expressions scaling as $\mathcal{O}(N)$.

Importance sampling

Using the identity

$$\frac{\partial}{\partial x_i} g_{ij} = -\frac{\partial}{\partial x_j} g_{ij},$$

we get expressions where all the derivatives acting on the particle are represented by the *second* index of g:

$$\frac{1}{\Psi_C}\frac{\partial \Psi_C}{\partial x_k} = \sum_{i=1}^{k-1}\frac{1}{g_{ik}}\frac{\partial g_{ik}}{\partial x_k} - \sum_{i=k+1}^{N}\frac{1}{g_{ki}}\frac{\partial g_{ki}}{\partial x_i},$$

and for the exponential case:

$$\frac{1}{\Psi_C}\frac{\partial \Psi_C}{\partial x_k} = \sum_{i=1}^{k-1}\frac{\partial g_{ik}}{\partial x_k} - \sum_{i=k+1}^{N}\frac{\partial g_{ki}}{\partial x_i}.$$

Importance sampling

For correlation forms depending only on the scalar distances r_{ij} we can use the chain rule. Noting that

$$\frac{\partial g_{ij}}{\partial x_j} = \frac{\partial g_{ij}}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial x_j} = \frac{x_j - x_i}{r_{ij}} \frac{\partial g_{ij}}{\partial r_{ij}},$$

we arrive at

$$\frac{1}{\Psi_C} \frac{\partial \Psi_C}{\partial x_k} = \sum_{i=1}^{k-1} \frac{1}{g_{ik}} \frac{\mathrm{r_{ik}}}{r_{ik}} \frac{\partial g_{ik}}{\partial r_{ik}} - \sum_{i=k+1}^{N} \frac{1}{g_{ki}} \frac{\mathrm{r_{ki}}}{r_{ki}} \frac{\partial g_{ki}}{\partial r_{ki}}.$$

Importance sampling

Note that for the Pade-Jastrow form we can set $g_{ij} \equiv g(r_{ij}) = e^{f(r_{ij})} = e^{f_{ij}}$ and

$$\frac{\partial g_{ij}}{\partial r_{ij}} = g_{ij}\frac{\partial f_{ij}}{\partial r_{ij}}.$$

Therefore,

$$\frac{1}{\Psi_C}\frac{\partial \Psi_C}{\partial x_k} = \sum_{i=1}^{k-1}\frac{\mathrm{r_{ik}}}{r_{ik}}\frac{\partial f_{ik}}{\partial r_{ik}} - \sum_{i=k+1}^{N}\frac{\mathrm{r_{ki}}}{r_{ki}}\frac{\partial f_{ki}}{\partial r_{ki}},$$

where

$$\mathrm{r}_{ij} = |\mathrm{r}_j - \mathrm{r}_i| = (x_j - x_i)\mathrm{e}_1 + (y_j - y_i)\mathrm{e}_2 + (z_j - z_i)\mathrm{e}_3$$

is the relative distance.

Importance sampling

The second derivative of the Jastrow factor divided by the Jastrow factor (the way it enters the kinetic energy) is

$$\left[\frac{\nabla^2 \Psi_C}{\Psi_C}\right]_x = 2\sum_{k=1}^{N}\sum_{i=1}^{k-1}\frac{\partial^2 g_{ik}}{\partial x_k^2} + \sum_{k=1}^{N}\left(\sum_{i=1}^{k-1}\frac{\partial g_{ik}}{\partial x_k} - \sum_{i=k+1}^{N}\frac{\partial g_{ki}}{\partial x_i}\right)^2$$

Importance sampling

But we have a simple form for the function, namely

$$\Psi_C = \prod_{i<j} \exp f(r_{ij}),$$

and it is easy to see that for particle k we have

$$\frac{\nabla_k^2 \Psi_C}{\Psi_C} = \sum_{ij \neq k} \frac{(\mathbf{r}_k - \mathbf{r}_i)(\mathbf{r}_k - \mathbf{r}_j)}{r_{ki} r_{kj}} f'(r_{ki}) f'(r_{kj}) + \sum_{j \neq k} \left(f''(r_{kj}) + \frac{2}{r_{kj}} f'(r_{kj}) \right)$$

Top-down optimization view

- We will start with a top-down view, with a simple harmonic oscillator problem in one dimension as case.
- Thereafter we continue with implementing the simplest possible steepest descent approach to our two-electron problem with an electrostatic (Coulomb) interaction. Our code includes also importance sampling. The simple Python code here illustrates the basic elements which need to be included in our own code.
- Then we move on to the mathematical description of various gradient methods.

Motivation

Our aim with this part of the project is to be able to

- find an optimal value for the variational parameters using only some few Monte Carlo cycles
- use these optimal values for the variational parameters to perform a large-scale Monte Carlo calculation

To achieve this will look at methods like the simplest possible *gradient descent*, *Steepest descent*, the *conjugate gradient method* and stochastic gradient descent. These methods allow us to find the minima of a multivariable function like our energy (function of several variational parameters). Alternatively, you can always use Newton's method. In particular, since we will normally have one variational parameter, Newton's method can be easily used in finding the minimum of the local energy.

Simple example and demonstration

Let us illustrate what is needed in our calculations using a simple example, the harmonic oscillator in one dimension. For the harmonic oscillator in one-dimension we have a trial wave function and probability

$$\psi_T(x;\alpha) = \exp -(\frac{1}{2}\alpha^2 x^2),$$

which results in a local energy

$$\frac{1}{2}\left(\alpha^2 + x^2(1-\alpha^4)\right).$$

We can compare our numerically calculated energies with the exact energy as function of α

$$\overline{E}[\alpha] = \frac{1}{4}\left(\alpha^2 + \frac{1}{\alpha^2}\right).$$

Simple example and demonstration

The derivative of the energy with respect to α gives

$$\frac{d\langle E_L[\alpha]\rangle}{d\alpha} = \frac{1}{2}\alpha - \frac{1}{2\alpha^3}$$

and a second derivative which is always positive (meaning that we find a minimum)

$$\frac{d^2\langle E_L[\alpha]\rangle}{d\alpha^2} = \frac{1}{2} + \frac{3}{2\alpha^4}$$

The condition

$$\frac{d\langle E_L[\alpha]\rangle}{d\alpha} = 0,$$

gives the optimal $\alpha = 1$, as expected.

a) Derive the local energy for the harmonic oscillator in one dimension and find its expectation value.
b) Show also that the optimal value of optimal $\alpha = 1$
c) Repeat the above steps in two dimensions for N bosons or electrons. What is the optimal value of α?

Variance in the simple model

We can also minimize the variance. In our simple model the variance is

$$\sigma^2[\alpha] = \frac{1}{4}\left(1 + (1-\alpha^4)^2 \frac{3}{4\alpha^4}\right) - \overline{E}^2.$$

which yields a second derivative which is always positive.

Computing the derivatives

In general we end up computing the expectation value of the energy in terms of some parameters $\alpha_0, \alpha_1, \ldots, \alpha_n$ and we search for a minimum in this multi-variable parameter space. This leads to an energy minimization problem *where we need the derivative of the energy as a function of the variational parameters*.

In the above example this was easy and we were able to find the expression for the derivative by simple derivations. However, in our actual calculations the energy is represented by a multi-dimensional integral with several variational parameters. How can we can then obtain the derivatives of the energy with respect to the variational parameters without having to resort to expensive numerical derivations?

Expressions for finding the derivatives of the local energy

To find the derivatives of the local energy expectation value as function of the variational parameters, we can use the chain rule and the hermiticity of the Hamiltonian.
Let us define

$$\bar{E}_\alpha = \frac{d\langle E_L[\alpha]\rangle}{d\alpha}.$$

as the derivative of the energy with respect to the variational parameter α (we limit ourselves to one parameter only). In the above example this was easy and we obtain a simple expression for the derivative. We define also the derivative of the trial function (skipping the subindex T) as

$$\bar{\psi}_\alpha = \frac{d\psi[\alpha]\rangle}{d\alpha}.$$

Derivatives of the local energy

The elements of the gradient of the local energy are then (using the chain rule and the hermiticity of the Hamiltonian)

$$\bar{E}_\alpha = 2\left(\langle\frac{\bar{\psi}_\alpha}{\psi[\alpha]}E_L[\alpha]\rangle - \langle\frac{\bar{\psi}_\alpha}{\psi[\alpha]}\rangle\langle E_L[\alpha]\rangle\right).$$

From a computational point of view it means that you need to compute the expectation values of

$$\langle\frac{\bar{\psi}_\alpha}{\psi[\alpha]}E_L[\alpha]\rangle,$$

and

$$\langle\frac{\bar{\psi}_\alpha}{\psi[\alpha]}\rangle\langle E_L[\alpha]\rangle$$

a) Show that

$$\bar{E}_\alpha = 2\left(\langle\frac{\bar{\psi}_\alpha}{\psi[\alpha]}E_L[\alpha]\rangle - \langle\frac{\bar{\psi}_\alpha}{\psi[\alpha]}\rangle\langle E_L[\alpha]\rangle\right).$$

b) Find the corresponding expression for the variance.

Python program for 2-electrons in 2 dimensions

```
# 2-electron VMC code for 2dim quantum dot with importance sampling
# Using gaussian rng for new positions and Metropolis- Hastings
# Added energy minimization with gradient descent using fixed step siz
# To do: replace with optimization codes from scipy and/or use stochas
from math import exp, sqrt
from random import random, seed, normalvariate
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import sys



# Trial wave function for the 2-electron quantum dot in two dims
def WaveFunction(r,alpha,beta):
    r1 = r[0,0]**2 + r[0,1]**2
    r2 = r[1,0]**2 + r[1,1]**2
    r12 = sqrt((r[0,0]-r[1,0])**2 + (r[0,1]-r[1,1])**2)
    deno = r12/(1+beta*r12)
    return exp(-0.5*alpha*(r1+r2)+deno)

# Local energy  for the 2-electron quantum dot in two dims, using anal
def LocalEnergy(r,alpha,beta):

    r1 = (r[0,0]**2 + r[0,1]**2)
    r2 = (r[1,0]**2 + r[1,1]**2)
```

Using Broyden's algorithm in scipy

The following function uses the above described BFGS algorithm. Here we have defined a function which calculates the energy and a function which computes the first derivative.

```
# 2-electron VMC code for 2dim quantum dot with importance sampling
# Using gaussian rng for new positions and Metropolis- Hastings
# Added energy minimization using the BFGS algorithm, see p. 136 of ht
from math import exp, sqrt
from random import random, seed, normalvariate
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
from scipy.optimize import minimize
import sys


# Trial wave function for the 2-electron quantum dot in two dims
def WaveFunction(r,alpha,beta):
    r1 = r[0,0]**2 + r[0,1]**2
    r2 = r[1,0]**2 + r[1,1]**2
    r12 = sqrt((r[0,0]-r[1,0])**2 + (r[0,1]-r[1,1])**2)
    deno = r12/(1+beta*r12)
    return exp(-0.5*alpha*(r1+r2)+deno)

# Local energy  for the 2-electron quantum dot in two dims, using anal
def LocalEnergy(r,alpha,beta):
```

Brief reminder on Newton-Raphson's method

Let us quickly remind ourselves how we derive the above method. Perhaps the most celebrated of all one-dimensional root-finding routines is Newton's method, also called the Newton-Raphson method. This method requires the evaluation of both the function f and its derivative f' at arbitrary points. If you can only calculate the derivative numerically and/or your function is not of the smooth type, we normally discourage the use of this method.

The equations

The Newton-Raphson formula consists geometrically of extending the tangent line at a current point until it crosses zero, then setting the next guess to the abscissa of that zero-crossing. The mathematics behind this method is rather simple. Employing a Taylor expansion for x sufficiently close to the solution s, we have

$$f(s) = 0 = f(x) + (s-x)f'(x) + \frac{(s-x)^2}{2}f''(x) + \dots.$$

For small enough values of the function and for well-behaved functions, the terms beyond linear are unimportant, hence we obtain

$$f(x) + (s-x)f'(x) \approx 0,$$

yielding

$$s \approx x - \frac{f(x)}{f'(x)}.$$

Having in mind an iterative procedure, it is natural to start iterating with

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

Simple geometric interpretation

The above is Newton-Raphson's method. It has a simple geometric interpretation, namely x_{n+1} is the point where the tangent from $(x_n, f(x_n))$ crosses the x-axis. Close to the solution, Newton-Raphson converges fast to the desired result. However, if we are far from a root, where the higher-order terms in the series are important, the Newton-Raphson formula can give grossly inaccurate results. For instance, the initial guess for the root might be so far from the true root as to let the search interval include a local maximum or minimum of the function. If an iteration places a trial guess near such a local extremum, so that the first derivative nearly vanishes, then Newton-Raphson may fail totally

Extending to more than one variable

Newton's method can be generalized to systems of several non-linear equations and variables. Consider the case with two equations

$$\begin{aligned} f_1(x_1,x_2) &= 0 \\ f_2(x_1,x_2) &= 0, \end{aligned}$$

which we Taylor expand to obtain

$$\begin{array}{ll} 0 = f_1(x_1+h_1,x_2+h_2) = & f_1(x_1,x_2) + h_1\partial f_1/\partial x_1 + h_2\partial f_1/\partial x_2 + \dots \\ 0 = f_2(x_1+h_1,x_2+h_2) = & f_2(x_1,x_2) + h_1\partial f_2/\partial x_1 + h_2\partial f_2/\partial x_2 + \dots \end{array}.$$

Defining the Jacobian matrix $\hat{J}$ we have

$$\hat{J} = \left(\begin{array}{cc} \partial f_1/\partial x_1 & \partial f_1/\partial x_2 \\ \partial f_2/\partial x_1 & \partial f_2/\partial x_2 \end{array} \right),$$

we can rephrase Newton's method as

$$\left(\begin{array}{c} x_1^{n+1} \\ x_2^{n+1} \end{array} \right) = \left(\begin{array}{c} x_1^{n} \\ x_2^{n} \end{array} \right) + \left(\begin{array}{c} h_1^{n} \\ h_2^{n} \end{array} \right),$$

where we have defined

Steepest descent

The basic idea of gradient descent is that a function $F(\mathrm{x})$, $\mathrm{x} \equiv (x_1, \cdots, x_n)$, decreases fastest if one goes from x in the direction of the negative gradient $-\nabla F(\mathrm{x})$.
It can be shown that if

$$\mathrm{x}_{k+1} = \mathrm{x}_k - \gamma_k \nabla F(\mathrm{x}_k),$$

with $\gamma_k > 0$.
For γ_k small enough, then $F(\mathrm{x}_{k+1}) \leq F(\mathrm{x}_k)$. This means that for a sufficiently small γ_k we are always moving towards smaller function values, i.e a minimum.

More on Steepest descent

The previous observation is the basis of the method of steepest descent, which is also referred to as just gradient descent (GD). One starts with an initial guess x_0 for a minimum of F and computes new approximations according to

$$\mathrm{x}_{k+1} = \mathrm{x}_k - \gamma_k \nabla F(\mathrm{x}_k), \quad k \geq 0.$$

The parameter γ_k is often referred to as the step length or the learning rate within the context of Machine Learning.

The ideal

Ideally the sequence $\{\mathrm{x}_k\}_{k=0}$ converges to a global minimum of the function F. In general we do not know if we are in a global or local minimum. In the special case when F is a convex function, all local minima are also global minima, so in this case gradient descent can converge to the global solution. The advantage of this scheme is that it is conceptually simple and straightforward to implement. However the method in this form has some severe limitations:
In machine learing we are often faced with non-convex high dimensional cost functions with many local minima. Since GD is deterministic we will get stuck in a local minimum, if the method converges, unless we have a very good intial guess. This also implies that the scheme is sensitive to the chosen initial condition.
Note that the gradient is a function of $\mathrm{x} = (x_1, \cdots, x_n)$ which makes it expensive to compute numerically.

The sensitiveness of the gradient descent

The gradient descent method is sensitive to the choice of learning rate γ_k. This is due to the fact that we are only guaranteed that $F(\mathrm{x}_{k+1}) \leq F(\mathrm{x}_k)$ for sufficiently small γ_k. The problem is to determine an optimal learning rate. If the learning rate is chosen too small the method will take a long time to converge and if it is too large we can experience erratic behavior.
Many of these shortcomings can be alleviated by introducing randomness. One such method is that of Stochastic Gradient Descent (SGD), see below.

Convex function

Convex function: Let $X \subset \mathbb{R}^n$ be a convex set. Assume that the function $f : X \to \mathbb{R}$ is continuous, then f is said to be convex if

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

for all $x_1, x_2 \in X$ and for all $t \in [0,1]$. If $\leq$ is replaced with a strict inequaltiy in the definition, we demand $x_1 \neq x_2$ and $t \in (0,1)$ then f is said to be strictly convex. For a single variable function, convexity means that if you draw a straight line connecting $f(x_1)$ and $f(x_2)$, the value of the function on the interval $[x_1, x_2]$ is always below the line as illustrated below.

Conditions on convex functions

In the following we state first and second-order conditions which ensures convexity of a function f. We write D_f to denote the domain of f, i.e the subset of R^n where f is defined. For more details and proofs we refer to: S. Boyd and L. Vandenberghe. Convex Optimization. Cambridge University Press.

First order condition

Suppose f is differentiable (i.e $\nabla f(x)$ is well defined for all x in the domain of f). Then f is convex if and only if D_f is a convex set and

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

holds for all $x, y \in D_f$. This condition means that for a convex function the first order Taylor expansion (right hand side above) at any point a global under estimator of the function. To convince yourself you can make a drawing of $f(x) = x^2 + 1$ and draw the tangent line to $f(x)$ and note that it is always below the graph.

Second order condition

Assume that f is twice differentiable, i.e the Hessian matrix exists

More on convex functions

The next result is of great importance to us and the reason why we are going on about convex functions. In machine learning we frequently have to minimize a loss/cost function in order to find the best parameters for the model we are considering.
Ideally we want the global minimum (for high-dimensional models it is hard to know if we have local or global minimum). However, if the cost/loss function is convex the following result provides invaluable information:

Any minimum is global for convex functions

Consider the problem of finding $x \in \mathbb{R}^n$ such that $f(x)$ is minimal, where f is convex and differentiable. Then, any point x^* that satisfies $\nabla f(x^*) = 0$ is a global minimum.

This result means that if we know that the cost/loss function is convex and we are able to find a minimum, we are guaranteed that it is a global minimum.

Some simple problems

1. Show that $f(x) = x^2$ is convex for $x \in \mathbb{R}$ using the definition of convexity. Hint: If you re-write the definition, f is convex if the following holds for all $x, y \in D_f$ and any $\lambda \in [0, 1]$
   $\lambda f(x) + (1-\lambda)f(y) - f(\lambda x + (1-\lambda)y) \geq 0$.
2. Using the second order condition show that the following functions are convex on the specified domain.
   - $f(x) = e^x$ is convex for $x \in \mathbb{R}$.
   - $g(x) = -\ln(x)$ is convex for $x \in (0, \infty)$.
3. Let $f(x) = x^2$ and $g(x) = e^x$. Show that $f(g(x))$ and $g(f(x))$ is convex for $x \in \mathbb{R}$. Also show that if $f(x)$ is any convex function than $h(x) = e^{f(x)}$ is convex.
4. A norm is any function that satisfy the following properties
   - $f(\alpha x) = |\alpha| f(x)$ for all $\alpha \in \mathbb{R}$.
   - $f(x + y) \leq f(x) + f(y)$
   - $f(x) \leq 0$ for all $x \in \mathbb{R}^n$ with equality if and only if $x = 0$

Using the definition of convexity, try to show that a function satisfying the properties above is convex (the third condition is not needed to show this).

Standard steepest descent

Before we proceed, we would like to discuss the approach called the **standard Steepest descent**, which again leads to us having to be able to compute a matrix. It belongs to the class of Conjugate Gradient methods (CG).

The success of the CG method for finding solutions of non-linear problems is based on the theory of conjugate gradients for linear systems of equations. It belongs to the class of iterative methods for solving problems from linear algebra of the type

$$\hat{A}\hat{x} = \hat{b}.$$

In the iterative process we end up with a problem like

$$\hat{r} = \hat{b} - \hat{A}\hat{x},$$

where $\hat{r}$ is the so-called residual or error in the iterative process. When we have found the exact solution, $\hat{r} = 0$.

Gradient method

The residual is zero when we reach the minimum of the quadratic equation

$$P(\hat{x}) = \frac{1}{2}\hat{x}^T\hat{A}\hat{x} - \hat{x}^T\hat{b},$$

with the constraint that the matrix $\hat{A}$ is positive definite and symmetric. This defines also the Hessian and we want it to be positive definite.

Steepest descent method

We denote the initial guess for $\hat{x}$ as $\hat{x}_0$. We can assume without loss of generality that

$$\hat{x}_0 = 0,$$

or consider the system

$$\hat{A}\hat{z} = \hat{b} - \hat{A}\hat{x}_0,$$

instead.

Steepest descent method

One can show that the solution $\hat{x}$ is also the unique minimizer of the quadratic form

$$f(\hat{x}) = \frac{1}{2}\hat{x}^T\hat{A}\hat{x} - \hat{x}^T\hat{x}, \quad \hat{x} \in \mathrm{R}^n.$$

This suggests taking the first basis vector $\hat{r}_1$ (see below for definition) to be the gradient of f at $\hat{x} = \hat{x}_0$, which equals

$$\hat{A}\hat{x}_0 - \hat{b},$$

and $\hat{x}_0 = 0$ it is equal $-\hat{b}$.

Final expressions

We can compute the residual iteratively as

$$\hat{r}_{k+1} = \hat{b} - \hat{A}\hat{x}_{k+1},$$

which equals

$$\hat{b} - \hat{A}(\hat{x}_k + \alpha_k \hat{r}_k),$$

or

$$(\hat{b} - \hat{A}\hat{x}_k) - \alpha_k \hat{A}\hat{r}_k,$$

which gives

$$\alpha_k = \frac{\hat{r}_k^T \hat{r}_k}{\hat{r}_k^T \hat{A}\hat{r}_k}$$

leading to the iterative scheme

$$\hat{x}_{k+1} = \hat{x}_k - \alpha_k \hat{r}_k,$$

Conjugate gradient method

In the CG method we define so-called conjugate directions and two vectors $\hat{s}$ and $\hat{t}$ are said to be conjugate if

$$\hat{s}^T\hat{A}\hat{t} = 0.$$

The philosophy of the CG method is to perform searches in various conjugate directions of our vectors $\hat{x}_i$ obeying the above criterion, namely

$$\hat{x}_i^T\hat{A}\hat{x}_j = 0.$$

Two vectors are conjugate if they are orthogonal with respect to this inner product. Being conjugate is a symmetric relation: if $\hat{s}$ is conjugate to $\hat{t}$, then $\hat{t}$ is conjugate to $\hat{s}$.

Conjugate gradient method

An example is given by the eigenvectors of the matrix

$$\hat{v}_i^T \hat{A} \hat{v}_j = \lambda \hat{v}_i^T \hat{v}_j,$$

which is zero unless $i = j$.

Conjugate gradient method

Assume now that we have a symmetric positive-definite matrix $\hat{A}$ of size $n \times n$. At each iteration $i + 1$ we obtain the conjugate direction of a vector

$$\hat{x}_{i+1} = \hat{x}_i + \alpha_i \hat{p}_i.$$

We assume that $\hat{p}_i$ is a sequence of n mutually conjugate directions. Then the $\hat{p}_i$ form a basis of R^n and we can expand the solution $\hat{A}\hat{x} = \hat{b}$ in this basis, namely

$$\hat{x} = \sum_{i=1}^{n} \alpha_i \hat{p}_i.$$

Conjugate gradient method

The coefficients are given by

$$\mathsf{A}\mathsf{x} = \sum_{i=1}^{n} \alpha_i \mathsf{A}\mathsf{p}_i = \mathsf{b}.$$

Multiplying with $\hat{p}_k^T$ from the left gives

$$\hat{p}_k^T \hat{A}\hat{x} = \sum_{i=1}^{n} \alpha_i \hat{p}_k^T \hat{A}\hat{p}_i = \hat{p}_k^T \hat{b},$$

and we can define the coefficients α_k as

$$\alpha_k = \frac{\hat{p}_k^T \hat{b}}{\hat{p}_k^T \hat{A}\hat{p}_k}$$

Conjugate gradient method and iterations

If we choose the conjugate vectors $\hat{p}_k$ carefully, then we may not need all of them to obtain a good approximation to the solution $\hat{x}$. We want to regard the conjugate gradient method as an iterative method. This will us to solve systems where n is so large that the direct method would take too much time.
We denote the initial guess for $\hat{x}$ as $\hat{x}_0$. We can assume without loss of generality that

$$\hat{x}_0 = 0,$$

or consider the system

$$\hat{A}\hat{z} = \hat{b} - \hat{A}\hat{x}_0,$$

instead.

Conjugate gradient method

One can show that the solution $\hat{x}$ is also the unique minimizer of the quadratic form

$$f(\hat{x}) = \frac{1}{2}\hat{x}^T \hat{A}\hat{x} - \hat{x}^T \hat{x}, \quad \hat{x} \in \mathrm{R}^n.$$

This suggests taking the first basis vector $\hat{p}_1$ to be the gradient of f at $\hat{x} = \hat{x}_0$, which equals

$$\hat{A}\hat{x}_0 - \hat{b},$$

and $\hat{x}_0 = 0$ it is equal $-\hat{b}$. The other vectors in the basis will be conjugate to the gradient, hence the name conjugate gradient method.

Conjugate gradient method

Let $\hat{r}_k$ be the residual at the k-th step:

$$\hat{r}_k = \hat{b} - \hat{A}\hat{x}_k.$$

Note that $\hat{r}_k$ is the negative gradient of f at $\hat{x} = \hat{x}_k$, so the gradient descent method would be to move in the direction $\hat{r}_k$. Here, we insist that the directions $\hat{p}_k$ are conjugate to each other, so we take the direction closest to the gradient $\hat{r}_k$ under the conjugacy constraint. This gives the following expression

$$\hat{p}_{k+1} = \hat{r}_k - \frac{\hat{p}_k^T \hat{A}\hat{r}_k}{\hat{p}_k^T \hat{A}\hat{p}_k}\hat{p}_k.$$

Conjugate gradient method

We can also compute the residual iteratively as

$$\hat{r}_{k+1} = \hat{b} - \hat{A}\hat{x}_{k+1},$$

which equals

$$\hat{b} - \hat{A}(\hat{x}_k + \alpha_k \hat{p}_k),$$

or

$$(\hat{b} - \hat{A}\hat{x}_k) - \alpha_k \hat{A}\hat{p}_k,$$

which gives

$$\hat{r}_{k+1} = \hat{r}_k - \hat{A}\hat{p}_k,$$

Broyden–Fletcher–Goldfarb–Shanno algorithm

The optimization problem is to minimize $f(\mathrm{x})$ where x is a vector in R^n, and f is a differentiable scalar function. There are no constraints on the values that x can take.
The algorithm begins at an initial estimate for the optimal value x_0 and proceeds iteratively to get a better estimate at each stage.
The search direction p_k at stage k is given by the solution of the analogue of the Newton equation

$$B_k \mathrm{p}_k = -\nabla f(\mathrm{x}_k),$$

where B_k is an approximation to the Hessian matrix, which is updated iteratively at each stage, and $\nabla f(\mathrm{x}_k)$ is the gradient of the function evaluated at x_k. A line search in the direction p_k is then used to find the next point x_{k+1} by minimising

$$f(\mathrm{x}_k + \alpha \mathrm{p}_k),$$

over the scalar $\alpha > 0$.

Codes from numerical recipes

You can use codes we have adapted from the text Numerical Recipes in C++, see chapter 10.7. Here we present a program, which you also can find at the webpage of the course we use the functions **dfpmin** and **lnsrch**. This is a variant of the Broyden et al algorithm discussed in the previous slide.

- The program uses the harmonic oscillator in one dimensions as example.
- The program does not use armadillo to handle vectors and matrices, but employs rather my own vector-matrix class. These auxiliary functions, and the main program *model.cpp* can all be found under the program link here.

Below we show only excerpts from the main program. For the full program, see the above link.

Finding the minimum of the harmonic oscillator model in one dimension

```
//   Main function begins here
int main()
{
     int n, iter;
     double gtol, fret;
     double alpha;
     n = 1;
//   reserve space in memory for vectors containing the variational
//   parameters
     Vector g(n), p(n);
     cout << "Read in guess for alpha" << endl;
     cin >> alpha;
     gtol = 1.0e-5;
//   now call dfmin and compute the minimum
     p(0) = alpha;
     dfpmin(p, n, gtol, &iter, &fret, Efunction, dEfunction);
     cout << "Value of energy minimum = " << fret << endl;
     cout << "Number of iterations = " << iter << endl;
     cout << "Value of alpha at minimum = " << p(0) << endl;
      return 0;
}  // end of main program
```

Functions to observe

The functions **Efunction** and **dEfunction** compute the expectation value of the energy and its derivative. They use the the quasi-Newton method of Broyden, Fletcher, Goldfarb, and Shanno (BFGS) It uses the first derivatives only. The BFGS algorithm has proven good performance even for non-smooth optimizations. These functions need to be changed when you want to your own derivatives.

```
//  this function defines the expectation value of the local energy
double Efunction(Vector  &x)
{
  double value = x(0)*x(0)*0.5+1.0/(8*x(0)*x(0));
  return value;
} // end of function to evaluate

//  this function defines the derivative of the energy
void dEfunction(Vector &x, Vector &g)
{
  g(0) = x(0)-1.0/(4*x(0)*x(0)*x(0));
} // end of function to evaluate
```

You need to change these functions in order to compute the local energy for your system. I used 1000 cycles per call to get a new