Notes

1 Conjugate Gradient Method

1.1 Overview

The conjugate gradient method (cgm) is an algorithm used to solve a linear system of the form

$$Ax = b \tag{1}$$

Where A is a symmetric ($A^T = A$) positive definite ($x^T Ax > 0$) $n \times n$ matrix, $x,\ b$ vectors.
The algorithm is iterative, starting from a guess solution x_0 and taking a step towards the solution at each cycle.
The search directions are calculated from the residual term, defined as $r_i = b - Ax_i$.
It is possible to prove that by choosing the step direction to be A-orthogonal to all the previous ones, the solution converges the fastest (i.e. the error term $\|e_i\| = \|x_i - x\|$ is minimized).

1.2 Steepest descent

A simpler algorithm is the steepest descent.
The idea is to take a step in the direction of the residual so that the quadratic form is minimized.

$$x_{i+1} = x_i + \alpha_i r_i \tag{2}$$

$$\alpha_i \text{ such that } \frac{df(x_{i+1})}{d\alpha_i} = 0 \implies \alpha_i = \frac{r_i^T r_i}{r_i^T A r_i} \tag{3}$$

This method is inefficient as x_i often finds itself oscillating around the solution, since the search directions explore non-disjoint subspaces.

1.3 The algorithm

A better alternative is to set the search direction to be A-orthogonal to the error at the next iteration. If this is the case, it can be proven that the components of the error term are reduced to zero at each iteration, implying a convergence to the exact solution in n steps.

$$d_i^T A e_{i+1} = 0 \implies \frac{df(x_{i+1})}{d\alpha_i} = -r_{i+1}^T d_i = 0 \tag{4}$$

$$\alpha_i = \frac{r_i^T d_i}{d_i^T A d_i} \tag{5}$$

By definition, the residual is orthogonal to the previous search directions, we also have $r_i^T r_j = \delta_{ij}$. Since

$$r_{i+1} = -A(e_{i+1}) = -A(e_i + \alpha_i d_i) = r_i - \alpha_i A d_i \tag{6}$$

1.3.1 Procedure

The cgm algorithm can be summed up as follows:
Start with a guess solution x_0.
Let the first direction be the residual in x_0

$$d_0 = r_0 = b - Ax_0 \tag{7}$$

Now, at each iteration, we can compute

$$\begin{aligned} \alpha_i &= \frac{r_i^T r_i}{d_i^T A d_i} \\ x_{i+1} &= x_i + \alpha_i d_i \\ r_{i+1} &= r_i - \alpha_i A d_i \\ \beta_{i+1} &= \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i} \\ d_{i+1} &= r_{i+1} + \beta_{i+1} d_i \end{aligned}$$

1.4 Preconditioning

The rate of convergance of cgm depends on the conditioning of the matrix A, defined as $\kappa(A) = \frac{\max \lambda_i}{\min \lambda_i}$, where λ_i are the eigenvalues of the matrix.
The closer $\kappa(A)$ is to 1, the faster the convergence of the method.
Given a certain matrix M, symmetric, positive definite and easily invertible and such that $M^{-1}A$ has better conditioning than A, which is to say M well approximates A, we can hope to solve the problem

$$M^{-1}Ax = M^{-1}b \tag{8}$$

much faster than the original problem, where the two solutions will be the same.
The problem is that $M^{-1}A$ is not necessarily symmetric or positive definite.
The fact that $\exists E$ such that $M = EE^T$ and $E^{-1}AE^{-T}$ is symmetric and positive definite, we can solve the problem.

$$E^{-1}AE^{-T}x = E^{-1}b \tag{9}$$

By using some clever substitutions, we can go back to the original problem with the aid of the preconditioner, giving the following algorithm

$$\begin{aligned}
r_0 &= b - Ax_0 \\
d_0 &= M^{-1}r_0 \\
\alpha_i &= \frac{r_i^T M^{-1} r_i}{d_i^T A d_i} \\
x_{i+1} &= x_i + \alpha_i d_i \\
r_{i+1} &= r_i - \alpha_i A d_i \\
\beta_{i+1} &= \frac{r_{i+1}^T M^{-1} r_{i+1}}{r_i^T M^{-1} r_i} \\
d_{i+1} &= M^{-1} r_{i+1} + \beta_{i+1} d_i
\end{aligned}$$

2 Finding the smallest eigenvalue

Finding the smallest/biggest eigenvalue-eigenvector pair of a matrix amounts to evaluating the unconstrained minimum/maximum of the Reyleigh quotient

$$\lambda(x) = \frac{x^T A x}{x^T x} \tag{10}$$

Or, more generally

$$Ax = B\omega x \implies \lambda(x) = \frac{x^T A x}{x^T B x} \tag{11}$$

λ is not a quadratic form, hence the cgm needs to be modified to use it.

2.1 Useful multivariable relations

Given $f(x) = x^T A x$ and taking the derivative of f in the direction of v

$$f(x + hv) = (x + hv)^T A(x + hv) = f(x) + hv^T A x + h x^T A v + o(h) \tag{12}$$

$$\frac{\mathrm{d}f}{\mathrm{d}v} = \lim_{h \to 0} \frac{f(x + hv) - f(x)}{h} = v^T A x + x^T A v = v^T A x + v^T A^T x \tag{13}$$

We can now evaluate the gradient of f in x

$$\nabla_x f(x) = \frac{\mathrm{d}f}{\mathrm{d}v} = (A + A^T)x \tag{14}$$

We can now take the gradient of the Rayleigh quotient

$$\nabla \lambda(x) = \frac{(A + A^T)x x^T B x - (B + B^T) x x^T A x}{(x^T B x)^2} \tag{15}$$

Using the fact that A and B are symemtric

$$\nabla \lambda(x) = 2\frac{A x x^T B x - B x x^T A x}{(x^T B x)^2} = 2\frac{Ax - \lambda(x) B x}{x^T B x} \tag{16}$$

2.2 Non linear conjugate gradient

Using a non quadratic form as function to be minimized, the things that will change will be

- The step size α_i will be different, we may now have multiple zeros regarding the orthogonality of the gradient and search direction.
- The factor β to compute conjugated directions no longer has equivalent forms.
- The residual needs to be computed each time as $-\nabla f(x_i)$

Let's take a look at each problem and find a workaround.

2.2.1 Step size

We ought to find the step size for which λ is minimized at each iteration. Being non linear (and non quadratic), an approximation must be done.
We can Taylor expand the function around x_i, in the direction αd_i, and find the minimum of the polynomial.
Regarding the Rayleigh quotient, it amounts to finding the positive roots of the following polynomial:

$$\begin{aligned} a\alpha_i^2 + b\alpha_i + c = 0 \\ a = (d_i^T A d_i)(x_i^T B d_i) - (x_i^T A d_i)(d_i^T B d_i) \\ b = (d_i^T A d_i)(x_i^T B x_i) - (x_i^T A x_i)(d_i^T B d_i) \\ c = (x_i^T A d_i)(x_i^T B x_i) - (x_i^T A x_i)(x_i^T B d_i) \end{aligned}$$

Being the search direction always descending, we can simply select the positive root.

2.2.2 Factor β

The choice for β is neither trivial nor unique, different formulations lead to distinct convergence properties and applicabilities.
Two possible choices are

$$\beta_{i+1}^{\text{FR}} = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i} \quad \text{or} \quad \beta_{i+1}^{\text{PR}} = \max\left\{\frac{r_{i+1}^T (r_{i+1} - r_i)}{r_i^T r_i}, 0\right\} \tag{17}$$

The max operation will restart the method if β is negative in the Polak Ribière, guaranteeing convergence.

2.2.3 Algorithm

The algorithm for minimizing the Rayleigh quotient can now be formulated as follows.

Choose an initial guess x_0.
Set the first search direction as the residual in x_0: $d_0 = r_0 = -g(x_0)$.
At each iteration, we can compute

$$\begin{aligned}
\alpha_i \text{ such that } f(x + \alpha_i d_i) \text{ minimized} \\
x_{i+1} = x_i + \alpha_i d_i \\
r_{i+1} = -g(x_{i+1}) \\
\beta_{i+1} \text{ from one of the possible choices} \\
d_{i+1} = r_{i+1} + \beta_{i+1} d_i
\end{aligned}$$

Since $\lambda(x)$ is not a quadratic form, the algorithm won't converge in n steps, so that we will need to check for convergence at each iteration.