ECE 602 – Section 3
Unconstrained minimization of smooth convex functions

- Unconstrained minimization
- First order optimality condition
- First and second order optimization methods
- Gradient descent and Newton-type algorithms
- Conjugate Gradient (CG) algorithm

Unconstrained minimization

- Consider solving

  $$\min_x f(x)$$

  where $f : \mathbf{R}^n \to \mathbf{R}$ is convex and $f \in \mathcal{C}^2(\mathbf{dom}\, f)$.

- Assuming there exists x^* such that $\inf_x f(x) = f(x^*) = p^*$, a *necessary and sufficient condition* for x^* to be optimal is

  $$\boxed{\nabla f(x^*) = 0}$$

  which is a set of n equations in n unknowns $x_1, \ldots, x_n$.

- Unless there is a closed form-solution (which is rare), we will look for a *minimizing sequence* $x^{(0)}, x^{(1)}, \ldots \in \mathbf{dom}\, f$, i.e., such that

  $$f(x^{(k)}) \to p^* \quad \text{as} \quad k \to \infty,$$

  with the iterations terminated when $f(x^{(k)}) - p^* < \epsilon$, for some $\epsilon > 0$.

Initial point and sublevel set

- Most algorithms require $x^{(0)} \in \mathbf{dom}\, f$ and the sublevel set S, defined by $\alpha_0 = f(x^{(0)})$ as $S = \{x \in \mathbf{dom}\, f \mid f(x) \le \alpha_0\}$, to be closed.
- The second condition on S is automatically satisfied when $f(x)$ (with $\mathbf{dom}\, f = \mathbf{R}^n$) is continuous and *coersive*, implying

$$f(x) \to \infty, \text{ as } x \to \mathbf{bd}\,\mathbf{dom}\, f.$$

- Some examples of twice-differentiable closed functions are

$$f(x) = \log\left(\sum_{i=1}^{m} \exp(a_i^T x + b_i)\right), \quad \mathbf{dom}\, f = \mathbf{R}^n$$

and

$$f(x) = -\sum_{i=1}^{m} \log(b_i - a_i^T x), \quad \mathbf{dom}\, f = \{x \mid Ax \prec b\}.$$

Strong convexity and implications

- The function f is *strongly convex* on S, if there is $\mu > 0$ such that

$$\nabla^2 f(x) \succeq \mu I, \quad \forall x \in S.$$

- In this case, we have

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{m}{2} \|y - x\|_2^2$$

which can be used to show that

$$f(x) - p^* \leq \frac{1}{2m} \|\nabla f(x)\|_2^2$$

thus giving us a useful stopping criterion.

- It can also be shown that

$$\|x - x^*\|_2 \leq \frac{2}{m} \|\nabla f(x)\|_2$$

which suggests that x^* is unique.

Descent methods

- Consider a minimizing sequence of the form

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)}, \quad t^{(k)} > 0$$

where $\Delta x^{(k)}$ is the *search direction* and $t^{(k)}$ the *step size*.

- In the case of *descent methods*, we have

$$f(x^{(k+1)}) < f(x^{(k)})$$

except when $f(x^{(k)})$ is optimal.

Search Direction

The *search direction* in any descent method must satisfy

$$\boxed{\nabla f(x^{(k)})^T \Delta x^{(k)} < 0}$$

- Such direction is always a *descent direction*.

General descent method

ALGORITHM: General descent method

```
given a starting point $x \in \textbf{dom}\, f$
repeat
    1. Determine a descent direction $\Delta x$
    2. Choose a step size $t > 0$ (line search)
    3. Update $x := x + t\Delta x$
until stopping criterion is satisfied.
```

- The *line search* procedure determines where along the line

  $$\{x + t\Delta x \mid t > 0\}$$

  the next iterate will be.

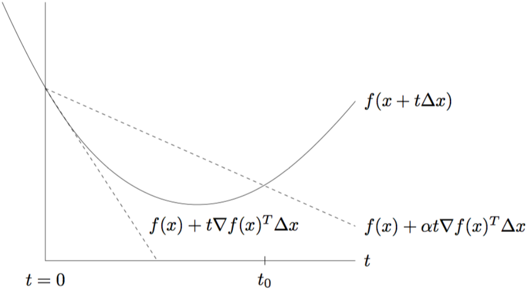
Line search types

- **Exact line search**: Choose t so that

$$t = \arg\min_{s} f(x + s\Delta x)$$

- **Backtracking line search**: Using some $\alpha \in (0, 0.5)$ and $\beta \in (0, 1)$, repeat $t := \beta t$ until

$$f(x + t\Delta x) \leq f(x) + \alpha t \Delta f(x)^T \Delta x$$



Gradient descent method: $\Delta x = -\nabla f(x)$

ALGORITHM: Gradient descent method

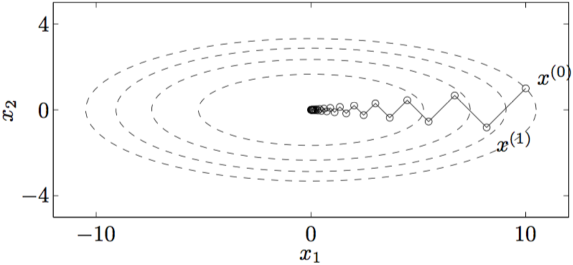
```
given a starting point $x \in \mathbf{dom}\, f$
repeat
    1. $\Delta x = -\nabla f(x)$
    2. Choose a step size $t > 0$ (line search)
    3. Update $x := x + t\Delta x$
until stopping criterion is satisfied.
```

- The stopping criterion is usually of the form $\|\nabla f(x)\| \leq \eta$ $(0 < \eta \ll 1)$.
- It can be shown that $f(x^{(k)}) - p^* \leq c^k (f(x^{(0)}) - p^*)$, where $c \in (0, 1)$ depends on m, $x^{(0)}$, and line search type.
- The method is very simple, but often very slow in practice (and, hence, rarely used).
- The method is known as the *Gradient Descent Method* (GDM).

Examples

- Consider minimizing $f(x) = 0.5(x_1^2 + \gamma x_2^2)$ (with $x^* = 0$ and $p^* = 0$).
- For $x^{(0)} = (\gamma, 1)$, gradient descent with exact line search can be shown to produce

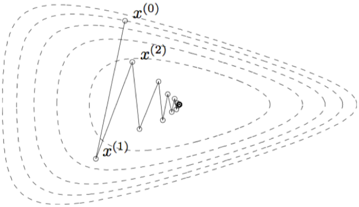
$$x_1^{(k)} = \gamma \left(\frac{\gamma - 1}{\gamma + 1}\right)^k, \quad x_2^{(k)} = \left(-\frac{\gamma - 1}{\gamma + 1}\right)^k$$



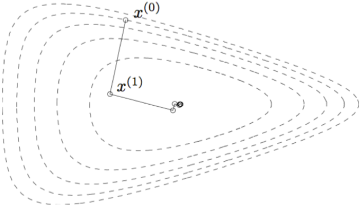
- The convergence is very slow for $\gamma \gg 1$ or $\gamma \ll 1$.

Examples (cont.)

- Consider minimizing $f(x_1, x_2) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$.
- We use gradient descent with a backtracking line search ($\alpha = 0.1$ and $\beta = 0.7$).



Backtracking line search



Exact line search

- Note that the sublevel sets of f are not too badly conditioned.

Some important observations/conclusions

- The gradient method often exhibits approximately *linear convergence*, i.e., $f(x^{(k)}) - p^*$ converges to zero approximately as a geometric series.

- The choice of backtracking parameters α, β has a noticeable but by no means dramatic effect on the convergence.

- An exact line search sometimes improves the convergence of the gradient method, but the effect is not large.

- The convergence rate depends greatly on the condition number of the Hessian, or the sublevel sets.

- When the condition number is 1000 or more, the gradient method is so slow that it is useless in practice.

- The main advantage of the gradient method is its simplicity.

Newton's method

- For $x \in \mathbf{dom}\, f$, the vector

$$\boxed{\Delta x_{\mathrm{nt}} = -\nabla^2 f(x)^{-1} \nabla f(x)}$$

  is called the *Newton direction*.

- The Newton direction is always a descent direction, since

$$\nabla f(x)^T \Delta x_{\mathrm{nt}} = -\nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x) < 0,$$

  unless $\nabla f(x) = 0$.

- It can be shown that the Newton step is the steepest descent direction at x w.r.t. the quadratic norm defined by the Hessian $\nabla^2 f(x)$, i.e.,

$$\|u\|_{\nabla^2 f(x)} = \left(u^T \nabla^2 f(x) u\right)^{1/2}.$$

Newton's method (cont.)

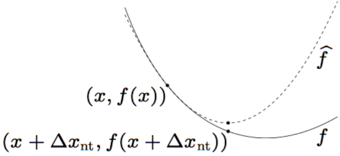
- Note that $x + \Delta x_{\mathrm{nt}}$ minimizes the 2nd-order approximation of $f(x)$ which is

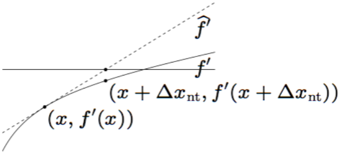
$$f(x+v) \approx \hat{f}(x+v) = f(x) + \nabla f(x)^T v + \frac{1}{2} v^T \nabla^2 f(x) v$$

(Differentiating w.r.t. v and equating to zero yields $v^* = \Delta x_{\mathrm{nt}}$.)

- Note also that $x + \Delta x_{\mathrm{nt}}$ solves the linearized optimality condition

$$\nabla f(x+v) \approx \nabla \hat{f}(x+v) = \nabla f(x) + \nabla^2 f(x) v = 0$$


$\widehat{f}$
$(x, f(x))$
$(x + \Delta x_{\mathrm{nt}}, f(x + \Delta x_{\mathrm{nt}}))$
f


$\widehat{f}'$
f'
$(x + \Delta x_{\mathrm{nt}}, f'(x + \Delta x_{\mathrm{nt}}))$
$(x, f'(x))$

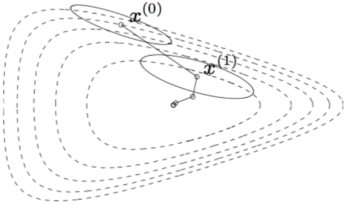
Newton's method (cont.)

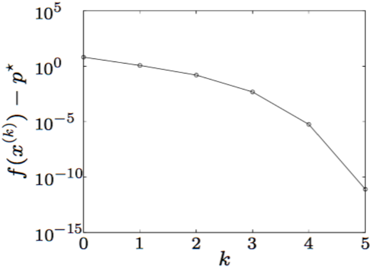
ALGORITHM: Newton method

```
given a starting point $x \in \textbf{dom}\, f$
repeat
    1. $\Delta x = -\nabla^2 f(x)^{-1} \nabla f(x)$
    2. Choose a step size $t > 0$ (line search)
    3. Update $x := x + t\Delta x$
until stopping criterion is satisfied.
```

- This algorithm is known as *the damped Newton method*, to distinguish it from *the pure Newton method*, which uses $t = 1$.
- Convergence of the method is rapid in general, and quadratic near x^*.
- Newton's method is affine invariant and scales well with problem size.
- Its performance is independent on the choice of algorithm parameters.
- The main disadvantage of Newton's method is the cost of forming and storing the Hessian, and the cost of computing the Newton step.

Example in R^2

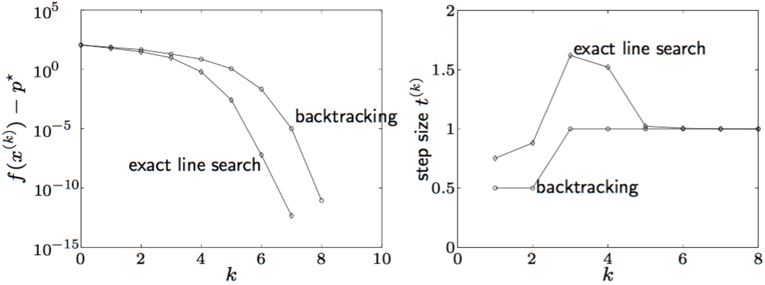




- Backtracking parameters: $\alpha = 0.1$, $\beta = 0.7$.
- Convergence in only 5 iterations.
- The local rate of convergence is quadratic.

Example in $\mathbf{R}^{100}$

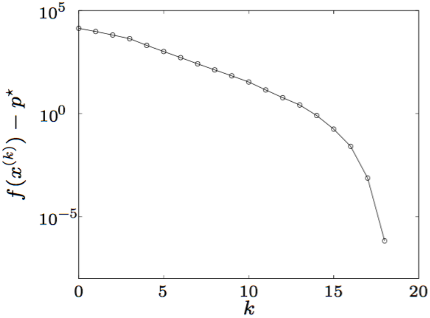
Consider minimizing $f(x) = c^T x - \sum_{i=1}^{m} \log(b_i - a_i^T x)$ with $m = 500$ and $n = 100$.



- Backtracking parameters: $\alpha = 0.01$, $\beta = 0.5$.
- Backtracking line search is almost as efficient as exact (which is much more "expensive").
- Note the two phases of the algorithm (viz., damped and pure).

Example in $\mathbf{R}^{10000}$

Consider minimizing $f(x) = -\sum_{i=1}^{10000} \log(1 - x_i^2) - \sum_{i=1}^{100000} \log(b_i - a_i^T x)$.



- Backtracking parameters: $\alpha = 0.01$, $\beta = 0.5$.
- Performance is similar as for small examples.

Gauss-Newton method

- Let $e_i : \mathbf{R}^n \to \mathbf{R}$ be an "error function", where $i = 1, \ldots, m$.
- E.g., $e_i(x) = y_i - M(t_i \mid x)$, where $M(t_i \mid x)$ models the behaviour of a system with parameters x at time t_i, while y_i is an observation.
- Consider the *least squares* (LS) problem of estimating x via minimizing the combined quadratic cost given by

$$f(x) = \frac{1}{2} \sum_{i=1}^{m} e_i^2(x).$$

- In general, this problem may not be convex (only local solutions can be found).

Gauss-Newton method (cont.)

- The gradient and Hessian of $f(x)$ are given by

$$\nabla f(x) = \sum_{i=1}^{m} e_i(x) \nabla e_i(x)$$

and

$$\nabla^2 f(x) = \sum_{i=1}^{m} \nabla e_i(x) \nabla e_i(x)^T + e_i(x) \nabla^2 e_i(x)^T$$

- The *Gauss-Newton* step is defined as

$$\boxed{\Delta x_{\mathrm{gn}} = -\left(\sum_{i=1}^{m} \nabla e_i(x) \nabla e_i(x)^T \right)^{-1} \left(\sum_{i=1}^{m} e_i(x) \nabla e_i(x) \right)}$$

- This search direction can be considered an approximate Newton direction.

- The method does not require the computation of 2nd-order derivatives.

Gauss-Newton method (cont.)

- Alternatively, one can consider $e : \mathbf{R}^n \to \mathbf{R}^m$ defined as

  $$e(x) := [e_1(x), e_2(x), \ldots, e_m(x)]^T$$

  in which case

  $$f(x) = \frac{1}{2}\|e(x)\|_2^2 = \frac{1}{2}e(x)^T e(x)$$

- Let $J(x)$ be the Jacobian of e, so that $e(x+v) \approx e(x) + J(x)v$. Then,

  $$\Delta x_{\mathrm{gn}} = \arg\min_v \frac{1}{2}\|e(x) + J(x)v\|_2^2 = -\left(J(x)^T J(x)\right)^{-1} J(x)^T e(x)$$

- In other words, Δx_{gn} minimizes the linear approximation of $f(x)$.

- Stable implementation requires: $m \geq n$ and $\operatorname{rank} J(x) = n$ for all x.

Levenberg - Marquardt algorithm

- *Levenberg's* contribution was to use "damping", i.e.

$$\boxed{\Delta x_{\mathrm{lm}} = -\left(J(x)^T J(x) + \lambda I\right)^{-1} J(x)^T e(x)}$$

with a *variable* regularization $\lambda > 0$.

- Note: $\Delta x_{\mathrm{lm}} \to \Delta x_{\mathrm{gn}}$, when $\lambda \to 0$, while $\Delta x_{\mathrm{lm}} \to \Delta x_{\mathrm{gd}}$, when $\lambda \to \infty$ (where Δx_{gd} stands for a gradient descent direction).

- *Marquardt* added an additional "trick", i.e.

$$\Delta x_{\mathrm{lm}} = -\left(J(x)^T J(x) + \lambda\, \mathrm{diag}(J(x)^T J(x))\right)^{-1} J(x)^T e(x)$$

which avoids slow convergence in the direction of small gradient.

- Nowadays, the *Levenberg-Marquardt algorithm* (aka *damped LS*) is a standardly used to solve non-linear LS problems.

Gram-Schmidt orthogonalization

- Given n linearly independent vectors $\{v_i\}_{i=1}^n$ in $\mathbf{R}^n$, we initiate the following procedure:

$$\text{Step 1:} \quad \tilde{e}_1 = v_1, \;\; e_1 = \frac{\tilde{e}_1}{\|\tilde{e}_1\|}.$$

$$\text{Step 2:} \quad \tilde{e}_2 = v_2 - (v_2^T e_1) e_1, \;\; e_2 = \frac{\tilde{e}_2}{\|\tilde{e}_2\|}.$$

$$\ldots$$

$$\text{Step } k\text{:} \quad \tilde{e}_k = v_k - \Big((v_k^T e_1) e_1 + \ldots + (v_k^T e_{k-1}) e_{k-1} \Big), \;\; e_k = \frac{\tilde{e}_k}{\|\tilde{e}_k\|}.$$

- More concisely, with $e_1 = v_1 / \|v_1\|$ and $k = 2, 3, \ldots, n$, we have

$$\boxed{e_k = \frac{v_k - \sum_{i=1}^{k-1} (v_k^T e_i) e_i}{\left\| v_k - \sum_{i=1}^{k-1} (v_k^T e_i) e_i \right\|}}$$

- The above procedure is known as *Gram-Schmidt orthogonalization* (or, simply, *Gram-Schmidt procedure*) and it guarantees that the resulting vectors $\{e_i\}_{i=1}^n$ are *orthonormal*.

Gram-Schmidt orthogonalization (cont.)

- Let $P \in \mathbf{S}^n_{++}$ be a positive definite matrix, which can be used to define an *inner product* on $\mathbf{R}^n$ as

  $$\langle x, y \rangle_P = x^T P\, y = (P^{1/2} x)^T (P^{1/2} y) \quad x, y \in \mathbf{R}^n,$$

  with the associated *weighted Euclidean norm* given by $\|x\|_P = x^T P\, x$.

- What if we apply the Gram-Schmidt procedure to linearly independent $\{v_i\}_{i=1}^n$, but now proceeding according to

  $$\boxed{e_k = \frac{v_k - \sum_{i=1}^{k-1} \langle v_k, e_i \rangle_P\, e_i}{\left\| v_k - \sum_{i=1}^{k-1} \langle v_k, e_i \rangle_P\, e_i \right\|_P}}$$

  for $k = 2, 3, \ldots, n$ and $e_1 = v_1 / \|v_1\|_P$.

- By construction, the resulting vectors $\{e_i\}_{i=1}^n$ are now *P-orthogonal* (implying $\langle e_i, e_j \rangle_P = \delta_{i,j}$), and they are conventionally referred to as P-*conjugate* (or, simply, *conjugate*) directions.

- The linear spaces spanned by conjugate vectors are known as *Krylov spaces* (with the e_i above being an example of a Krylov basis).

Unconstrained quadratic minimization

- Now, let us consider the problem of minimizing

$$f(x) = (1/2)x^T Px + p^T x + q,$$

  for some $p \in \mathbf{R}^n$ and $q \in \mathbf{R}$.

- We express the optimal solution x^* *in a parametric form* as

$$x^* = \sum_{i=1}^{n} \alpha_i e_i = E\alpha,$$

  where $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_n]^T$ and the columns of $E \in \mathbf{R}^{n\times n}$ are formed by the P-conjugate directions $\{e_i\}_{i=1}^n$ (implying that $E^T PE = I$).

- Then, substituting the parametrization into the original cost function yields

$$f(\alpha) = f(x)\Big|_{x=E\alpha} = (1/2)\alpha^T \underbrace{E^T PE}_{I} \alpha + p^T E\alpha + q =$$

$$= \frac{\|\alpha\|_2^2}{2} + \beta^T \alpha + q, \quad \text{with } \beta := E^T p.$$

Unconstrained quadratic minimization (cont.)

- Thus, in the Krylov space, the quadratic cost

$$f(\alpha) = \sum_{i=1}^{n} \underbrace{\left(\frac{\alpha_i^2}{2} + \beta_i \alpha_i + \frac{q}{n} \right)}_{f_i(\alpha_i)} = \sum_{i=1}^{n} f_i(\alpha_i)$$

  is *separable*, which means that *it can be minimized w.r.t. each α_i independently of the others*.

- In particular, we have the *optimal* values of α_i given by

$$\alpha_i = -\beta_i = -p^T e_i,$$

  which leads to a sequence of very simple updates.

- The use of conjugate directions as a means for rendering the quadratic optimization problem *separable* forms the basis of the *Conjugate Gradient Method* (CGM) (aka *Conjugate Directions Method* (CDM)).

Conjugate Gradient Method

- To formulate the CGM in the form of an iterative procedure, we first assume that x^* belongs to the affine space defined by

$$x^* = x^{(0)} + \sum_{i=1}^{n} \alpha_i \tilde{e}_i = x^{(0)} + \tilde{E}\alpha,$$

  where $x^{(0)} \in \mathbf{R}^n$ is an initialization. Note that, in this case, $\alpha_i = -\tilde{\beta}_i$, where $\tilde{\beta} = E^T(Px^{(0)} + p)$.

- The CGM takes advantage of the following result.

Expanding Manifold Property

Let $\mathcal{G}_k$ be an affine space defined as $\mathcal{G}_k = \{x \in \mathbf{R}^n \mid x = x^{(0)} + \sum_{i=1}^{k} \gamma_i \tilde{e}_i\}$. Then, the k-th iteration of the CGM produces

$$x^{(k)} = \arg\min_{x \in \mathcal{G}_k} f(x).$$

Moreover, the gradient of $f(x)$ at $x = x_0 + \sum_{i=1}^{k} \alpha_i \tilde{e}_i$ w.r.t. the vector of *partial* coefficients $\alpha^{(k)} = [\alpha_1, \alpha_2, \ldots, \alpha_k]^T$ is *orthogonal* to all $\tilde{e}_1, \tilde{e}_2, \ldots, \tilde{e}_k$.

Conjugate Gradient Method (cont.)

ALGORITHM: CGM for quadratic costs

```
given a starting point $x^{(0)}$ and $d^{(0)} = -(Px^{(0)} + p)$
repeat
    1. Update $x^{(k+1)} := x^{(k)} + \gamma^{(k)} d^{(k)}$ (via exact line search or BT)
    1. Compute $g^{(k+1)} = Px^{(k+1)} + p$ and
        $\beta^{(k)} = \frac{(g^{(k+1)} - g^{(k)})^T g^{(k+1)}}{(g^{(k+1)} - g^{(k)})^T d^{(k)}}$
        $d^{(k+1)} = -g^{(k+1)} + \sum_{i=0}^{k} \frac{\langle g^{(k+1)}, d^{(i)} \rangle_P}{\|d^{(i)}\|_P} d^{(i)} = -g^{(k+1)} + \beta^{(k)} d^{(k)}$
until stopping criterion is satisfied or $k = n$.
```

- The above method of computing $\beta^{(k)}$ is known as the *Polak-Ribiere method*. Using some further simplifications, one can also compute $\beta^{(k)}$ as

  $$\beta^{(k)} = \frac{\|g^{(k+1)}\|}{g^{(k)}},$$

  which is known as the *Fletcher-Reeves* method.

Convergence rate of CGM

- Generally, CGM generates much better descent directions compared to GDM.

- For the quadratic minimization problem $\min_x \{0.5\, x^T P x + p^T x\}$, one can show that

$$\|x^{(k+1)} - x^*\|_2 \leq \underbrace{\frac{\sqrt{\lambda_{\max}} - \sqrt{\lambda_{\min}}}{\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}}}}_{\approx 1 - 2\sqrt{\lambda_{\min}/\lambda_{\max}}} \|x^{(k)} - x^*\|_2,$$

  where $\lambda_{\max}$ and $\lambda_{\min}$ denote the maximum and the minimum eigenvalue of $P \succ 0$, respectively.

- Thus, the convergence rate of CGM is defined by $1 - 2/\sqrt{C}$, while that of GDM is only $1 - 2/C$ (where $C = \lambda_{\max}/\lambda_{\min}$, as before).

- For example, with $C = 10^6$, one iteration of CGM reduces the error by 0.998, while one iteration of GDM reduces the error by only 0.999998. Thus, one would need 1000 iterations of GDM to achieve the result of one (!) iteration of CGM.