Lecture 4: Unconstrained optimization algorithms

Method of choice

Consider the unconstrained optimization problem to

$$\underset{\boldsymbol{x} \in \mathbb{R}^n}{\text{minimize}}\ f(\boldsymbol{x}), \tag{1}$$

where $f \in C^0$ on $\mathbb{R}^n$ (f is continuous). Mostly, we assume that $f \in C^1$ holds (f is continuously differentiable), sometimes even C^2

- Size of the problem (n)?
- Are $\nabla f(\boldsymbol{x})$ and/or $\nabla^2 f(\boldsymbol{x})$ available; to what cost?
- What it is the goal? (Global/local minimum, stationary point?)
- What are the convexity properties of f?
- Do we have a good estimate of the location of a stationary point $\boldsymbol{x}^*$? (Can we use locally-only convergent methods?)

Example: curve fitting by least-squares

- Suppose we have m data points (t_i, b_i) believed to be related as

$$x_1 + x_2 \exp(x_3 t_i) + x_4 \exp(x_5 t_i) = b_i, \qquad i = 1, \ldots, m,$$

with unknown parameters $x_1, \ldots, x_5$. (Here, $\exp(x) = \mathrm{e}^x$.) The best description minimizes the total "residual error," given by the norm of the residual

$$f_i(\boldsymbol{x}) := b_i - [x_1 + x_2 \exp(x_3 t_i) + x_4 \exp(x_5 t_i)], \qquad i = 1, \ldots, m$$

- Resulting optimization problem:

$$\min_{\boldsymbol{x} \in \mathbb{R}^5} f(\boldsymbol{x}) := \sum_{i=1}^{m} |f_i(\boldsymbol{x})|^2 = \sum_{i=1}^{m} (b_i - [x_1 + x_2 \exp(x_3 t_i) + x_4 \exp(x_5 t_i)])^2$$

- Very often solved problem type within numerical analysis and mathematical statistics

Typical algorithm

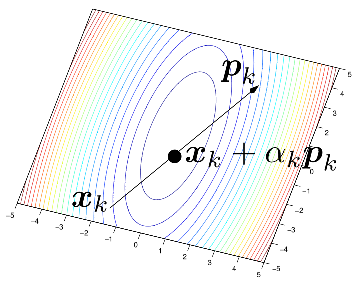
Step 0. *Starting point*: $\boldsymbol{x}_0 \in \mathbb{R}^n$. Set $k := 0$

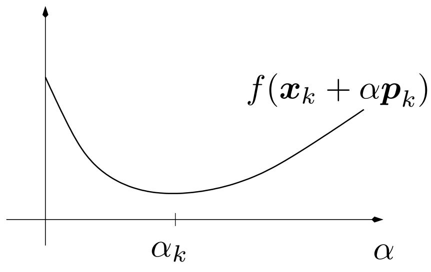
Step 1. *Search direction*: $\boldsymbol{p}_k \in \mathbb{R}^n$

Step 2. *Step length*: $\alpha_k > 0$ such that $f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) < f(\boldsymbol{x}_k)$ holds

Step 3. Let $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$

Step 4. *Termination criterion*: If fulfilled, then stop! Otherwise, let $k := k + 1$ and go to step 1





Notes

- The figure was plotted using several thousands of function evaluations
- Never possible in reality! (And total waste of time)
- An "orienteering map" never exists
- Most algorithms are inherently *local*, only based on info at the current point $\boldsymbol{x}_k$, that is, $f(\boldsymbol{x}_k)$, $\nabla f(\boldsymbol{x}_k)$, and $\nabla^2 f(\boldsymbol{x}_k)$
- Possibly also on previous points passed
- An algorithm is a "near-sighted mountain climber" when trying to reach the summit (for a max problem!)
- The mountain climber is in a deep fog and can only check her barometer for the height and feel the steepness of the slope under her feet

Step 1: Search directions

- If $\nabla f(\boldsymbol{x}_k) \neq \boldsymbol{0}^n$, then $\boldsymbol{p} = -\nabla f(\boldsymbol{x}_k)$ is a descent direction for f at $\boldsymbol{x}_k$ (Part of necessary condition proof!)
- This *steepest descent direction* solves the problem to

$$\underset{\boldsymbol{p} \in \mathbb{R}^n : \|\boldsymbol{p}\| = 1}{\text{minimize}} \ \nabla f(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{p}$$

- Suppose $\boldsymbol{Q} \in \mathbb{R}^{n \times n}$ is a symmetric, positive definite matrix. Then $\boldsymbol{p} = -\boldsymbol{Q} \nabla f(\boldsymbol{x}_k)$ is a descent direction for f at $\boldsymbol{x}_k$, because

$$\nabla f(\boldsymbol{x}_k)^{\mathrm{T}} \boldsymbol{p} = -\nabla f(\boldsymbol{x}_k)^{\mathrm{T}} \boldsymbol{Q} \nabla f(\boldsymbol{x}_k) < 0,$$

  due to the positive definiteness of $\boldsymbol{Q}$
- Special case: $\boldsymbol{Q} = \boldsymbol{I}^n$ yield steepest descent
- Special case: $\boldsymbol{Q}^{-1} = \nabla^2 f(\boldsymbol{x}_k)$, if the Hessian is positive definite. This is *Newton's method*

Additional requirements

$$|\nabla f(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{p}_k| \geq s_1\|\nabla f(\boldsymbol{x}_k)\|^2, \qquad \text{and} \quad \|\boldsymbol{p}_k\| \leq s_2\|\nabla f(\boldsymbol{x}_k)\|,$$

or

$$-\frac{\nabla f(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{p}_k}{\|\nabla f(\boldsymbol{x}_k)\|\cdot\|\boldsymbol{p}_k\|} \geq s_1, \qquad \text{and} \quad \|\boldsymbol{p}_k\| \geq s_2\|\nabla f(\boldsymbol{x}_k)\|$$

- Purpose: prevent the descent directions to deteriorate in quality, and prevent premature convergence
- $\nabla f(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{p}_k$ is the directional derivative of f at $\boldsymbol{x}_k$ in the direction of $\boldsymbol{p}_k$. Make sure it stays away from zero!
- Also, make sure that $\boldsymbol{p}_k$ stays bounded and that it tends to zero if and only if $\nabla f(\boldsymbol{x}_k)$ does
- These conditions hold for the above examples

Newton's method

- Steepest descent is most often not a very good algorithm. Why?
- It fails to take into account more than information about ∇f
- Let

$$f(\boldsymbol{x}+\boldsymbol{p}) - f(\boldsymbol{x}) \approx \varphi_{\boldsymbol{x}}(\boldsymbol{p}) = \nabla f(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^{\mathrm{T}}\nabla^2 f(\boldsymbol{x})\boldsymbol{p}$$

Minimize by setting gradient of $\varphi_{\boldsymbol{x}}(\boldsymbol{p})$ to zero:

$$\nabla_{\boldsymbol{p}}\varphi_{\boldsymbol{x}}(\boldsymbol{p}) = \nabla f(\boldsymbol{x}) + \nabla^2 f(\boldsymbol{x})\boldsymbol{p} = \boldsymbol{0}^n$$

- $n = 1$: $f'(x) + f''(x)p = 0 \implies p = -f'(x)/f''(x)$
- Provides descent if $f''(x) > 0$: $f'(x)p = -[f'(x)]^2/f''(x) < 0$
- Corresponding story in $\mathbb{R}^n$: $\boldsymbol{p} = -[\nabla^2 f(\boldsymbol{x})]^{-1}\nabla f(\boldsymbol{x})$, yields descent at non-stationary points if $\nabla^2 f(\boldsymbol{x})$ is positive definite!

Why do we not always choose Newton directions?

- **Lack of positive definiteness.** $\nabla^2 f(\boldsymbol{x})$ is not positive definite (PD). Solution: add diagonal matrix so that the result is PD: $\nabla^2 f(\boldsymbol{x}) + \gamma \boldsymbol{I}^n$ for $\gamma > 0$ large enough
- Note: If value of γ is very large $\Longrightarrow \approx$ steepest descent
- Name: *Levenberg–Marquardt*
- **Lack of enough differentiability.** If $f \notin C^2$, what do we do?
- $n = 1$: the *secant method*:

$$f''(x_k) \approx \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}$$

- $n > 1$: *quasi-Newton*: choose approximate matrix $\boldsymbol{B}_k$ so that

$$\boldsymbol{B}_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) = \nabla f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_{k-1}),$$

and more choices (the above does not specify the entire matrix!)

- **Computational burden.** It may be too much to ask for to solve a linear system many times when $n > 1000$ or so; it is enough to do *some* work on the linear system and still get a descent property. (See book for an example)
- Specific choices of matrices $\boldsymbol{B}_k$ lead to *quasi-Newton* methods

Step 2: Line search

- Approximately solve the one-dimensional problem to

$$\operatorname*{minimize}_{\alpha \geq 0} \; \varphi(\alpha) := f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k)$$

  Its optimality conditions are that

$$\varphi'(\alpha^*) \geq 0, \qquad \alpha^* \cdot \varphi'(\alpha^*) = 0, \qquad \alpha^* \geq 0,$$

  that is,

$$\nabla f(\boldsymbol{x}_k + \alpha^* \boldsymbol{p}_k)^{\mathrm{T}} \boldsymbol{p}_k \geq 0, \quad \alpha^* \cdot \nabla f(\boldsymbol{x}_k + \alpha^* \boldsymbol{p}_k)^{\mathrm{T}} \boldsymbol{p}_k = 0, \quad \alpha^* \geq 0,$$

  holds
- If $\alpha^* > 0$, then $\varphi'(\alpha^*) = 0$ holds, hence $\nabla f(\boldsymbol{x}_k + \alpha^* \boldsymbol{p}_k)^{\mathrm{T}} \boldsymbol{p}_k = 0$
- The search direction $\boldsymbol{p}_k$ is orthogonal to the gradient of f at the point $\boldsymbol{x}_k + \alpha^* \boldsymbol{p}_k$

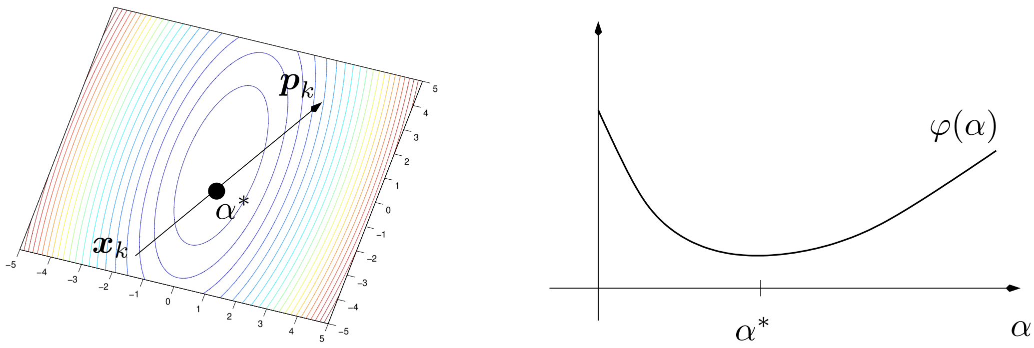


Figure 1: A line search in a descent direction

Approximate line search

- No point solving the one-dimensional problem exactly! Why? The optimum to the entire problem lies elsewhere!
- `Interpolation`: Use $f(\boldsymbol{x}_k), \nabla f(\boldsymbol{x}_k), \nabla f(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{p}_k$ to model a quadratic function approximating f along $\boldsymbol{p}_k$. Minimize it by using the analytic formula for quadratics
- `Newton's method`: Repeat the improvements gained from a quadratic approximation: $\alpha := \alpha - \varphi'(\alpha)/\varphi''(\alpha)$
- `Golden section`: Derivative-free method that shrinks an interval where $\varphi'(\alpha) = 0$ lies

Armijo rule

- Idea: quickly generate a step α which provides "sufficient" decrease in f. Note: $f(\boldsymbol{x}_k + \alpha\boldsymbol{p}_k) \approx f(\boldsymbol{x}_k) + \alpha \cdot \nabla f(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{p}_k$, valid for small values of $\alpha > 0$
- Requirement: we get a decrease in f which is at least a *fraction* of that predicted in the right-hand side above. Let $\mu \in (0,1)$ be this fraction. Acceptable step lengths are $\alpha > 0$ satisfying

$$\varphi(\alpha) - \varphi(0) \leq \mu\alpha\varphi'(0), \tag{2a}$$

that is,

$$f(\boldsymbol{x}_k + \alpha\boldsymbol{p}_k) - f(\boldsymbol{x}_k) \leq \mu\alpha\nabla f(\boldsymbol{x}_k)\boldsymbol{p}_k \tag{2b}$$

- Can add condition making α also large enough (*Wolfe*)

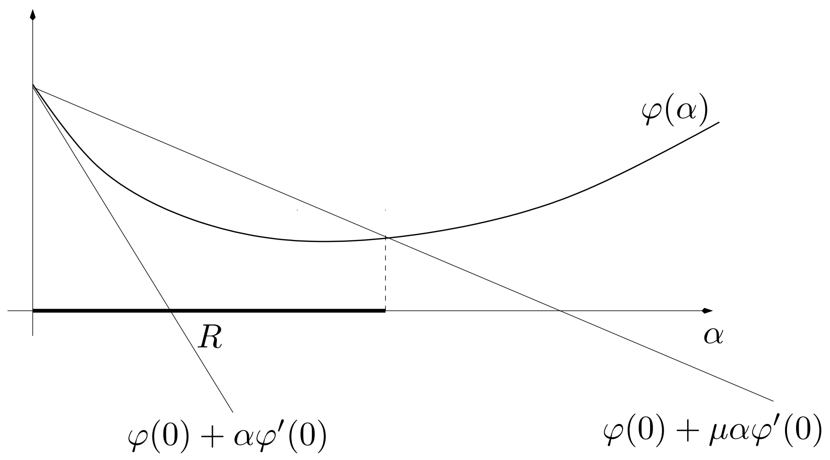


Figure 2: The interval (R) accepted by the Armijo step length rule

Typical convergence result

- *Suppose that $f \in C^1$, and that for the starting point $\boldsymbol{x}_0$ it holds that the level set* $\operatorname{lev}_f(f(\boldsymbol{x}_0)) = \{\, \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \le f(\boldsymbol{x}_0) \,\}$ *is bounded. Consider the iterative algorithm defined on Page 1, with the following choices for each k:*
  - $\boldsymbol{p}_k$ *satisfies the second sufficient descent condition on Page 7;*
  - $\|\boldsymbol{p}_k\| \le M$, *where M is some positive constant; and*
  - *the Armijo step length rule is used*

  Then, the sequence $\{\boldsymbol{x}_k\}$ is bounded, the sequence $\{f(\boldsymbol{x}_k)\}$ is descending, lower bounded and therefore converges, and every limit point of $\{\boldsymbol{x}_k\}$ is stationary $\square$
- For convex f much stronger convergence properties:

  Optimum exists $\iff$ $\{\boldsymbol{x}_k\}$ *converges to an optimal solution*

  (Theorem proved for gradient projection method later)

Step 4: Termination criteria

- Lesson number one: Cannot terminate based on the exact optimality conditions, because $\nabla f(\boldsymbol{x}) = \mathbf{0}^n$ never happens!
- The recommendation is the combination of the following:
  1. $\|\nabla f(\boldsymbol{x}_k)\| \le \varepsilon_1(1 + |f(\boldsymbol{x}_k)|)$, $\varepsilon_1 > 0$ small;
  2. $f(\boldsymbol{x}_{k-1}) - f(\boldsymbol{x}_k) \le \varepsilon_2(1 + |f(\boldsymbol{x}_k)|)$, $\varepsilon_2 > 0$ small; and
  3. $\|\boldsymbol{x}_{k-1} - \boldsymbol{x}_k\| \le \varepsilon_3(1 + \|\boldsymbol{x}_k\|)$, $\varepsilon_3 > 0$ small
- Why? Need to cover cases of very steep and very flat functions
- May need to use ∞-*norm*: $\|\boldsymbol{x}\|_\infty := \max_{1 \le j \le n} |x_j|$, for large n

- Problem with the scaling of the problem: If

$$\boldsymbol{x}_{k-1} = (1.44453, 0.00093, 0.0000079)^{\mathrm{T}},$$
$$\boldsymbol{x}_k = (1.44441, 0.00012, 0.0000011)^{\mathrm{T}};$$

$$\|\boldsymbol{x}_{k-1} - \boldsymbol{x}_k\|_\infty = \|(0.00012, 0.00081, 0.0000068)^{\mathrm{T}}\|_\infty = 0.00081$$

- Small absolute error but large relative error!
- Better to apply the algorithm from a scaled problem where elements of $\boldsymbol{x}$ have similar magnitude
- Newton methods define good such scalings

Why is the C^1 property important?

- Suppose f is only in C^0, not C^1. Example:

$$f(\boldsymbol{x}) := \underset{i \in \{1,\dots,m\}}{\text{maximum}} \{\boldsymbol{c}_i^{\mathrm{T}} \boldsymbol{x} + b_i\}, \qquad \boldsymbol{x} \in \mathbb{R}^n$$

- This is a piece-wise linear and convex function (see next page)
- It is differentiable almost everywhere, but *not* at the optimal solution!
- Ignoring non-differentiability may lead to the convergence to a non-optimal point
- Convex functions always has *subgradients*, corresponding to all the possible slopes of the function
- More on these when looking at Lagrangian duality!

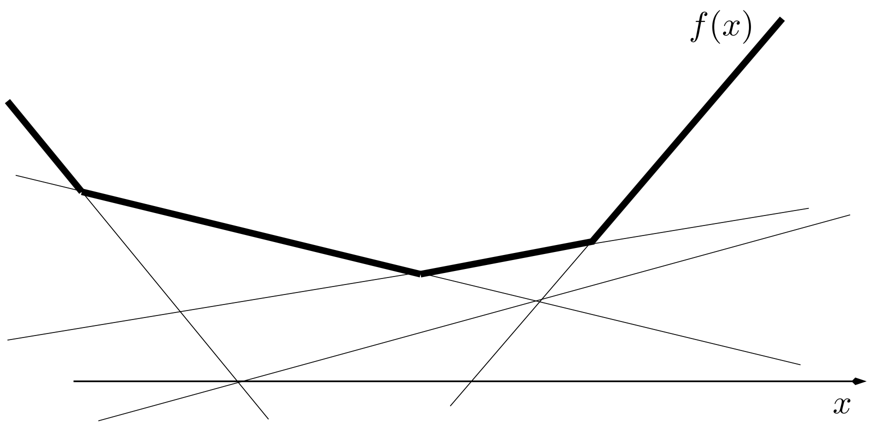


Figure 3: A piece-wise linear convex function

Trust region methods

- Trust region methods use quadratic models (as Newton)
- Avoids line searches by bounding the length of the search direction, at the same time influencing its direction
- Let $\psi_k(\boldsymbol{p}) := f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^{\mathrm{T}}\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{p}$
- The model ψ_k is *trusted* in a neighbourhood of $\boldsymbol{x}_k : \|\boldsymbol{p}\| \leq \Delta_k$
- Very useful when $\nabla^2 f(\boldsymbol{x}_k)$ is not positive semi-definite
- Easy to minimize $\psi_k(\boldsymbol{p})$ subject to $\|\boldsymbol{p}\| \leq \Delta_k$
- Idea: when $\nabla^2 f(\boldsymbol{x}_k)$ is badly conditioned, Δ_k should be small (more of a steepest descent method); if well conditioned, Δ_k should be large to allow for unit steps (Newton! fast convergence)
- If Δ_k is small enough, $f(\boldsymbol{x}_k + \boldsymbol{p}_k) < f(\boldsymbol{x}_k)$ holds
- Even if $\nabla f(\boldsymbol{x}_k) = \mathbf{0}^n$ holds, $f(\boldsymbol{x}_k + \boldsymbol{p}_k) < f(\boldsymbol{x}_k)$ still holds, if

$\nabla^2 f(\boldsymbol{x}_k)$ is not positive definite

- Progress from stationary points if saddle points or local maxima
- Update of trust region size based on a measure of similarity between the model ψ_k and f: Let

$$\rho_k = \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{p}_k)}{f(\boldsymbol{x}_k) - \psi_k(\boldsymbol{p}_k)} = \frac{\text{actual reduction}}{\text{predicted reduction}}$$

If $\rho_k \leq \mu$ let $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ (unsuccessful step), else

$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{p}_k$ (successful step)

Value of Δ_{k+1} depends on ρ_k:

$$\begin{aligned} \rho_k \leq \mu &\Longrightarrow \Delta_{k+1} = \tfrac{1}{2}\Delta_k, \\ \mu < \rho_k < \eta &\Longrightarrow \Delta_{k+1} = \Delta_k, \\ \rho_k \geq \eta &\Longrightarrow \Delta_{k+1} = 2\Delta_k \end{aligned}$$

Figure 4 illustrates the trust region subproblem

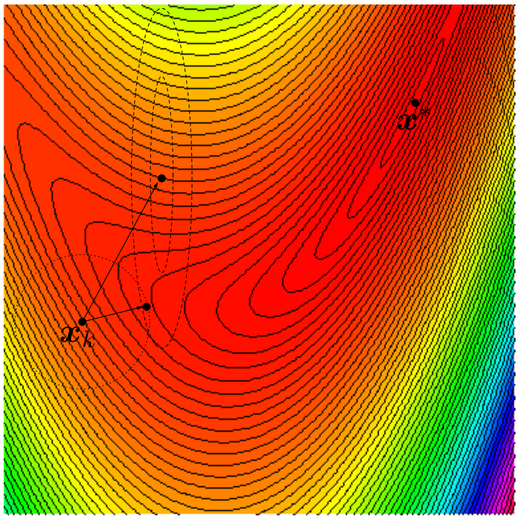


Figure 4: Trust region and line search step. The ellipses are level curves of the quadratic model; the circle defines the trust region

Minimizing implicit functions

- Common in engineering and natural science applications that f is not explicitly given but through a simulation:

$$\xrightarrow{\boldsymbol{x} \in \mathbb{R}^n} \boxed{\text{Simulation}} \longrightarrow \boldsymbol{y} \in \mathbb{R}^m$$

- Wish is to minimize a function of both $\boldsymbol{x}$ and $\boldsymbol{y}$: $f(\boldsymbol{x}, \boldsymbol{y})$; find the vector $\boldsymbol{x}$ that gives the best *response* $\boldsymbol{y}$ for f
- The form of the response $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x})$ from the input $\boldsymbol{x}$ is normally unknown
- Cannot differentiate $\boldsymbol{x} \mapsto f(\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x}))$
- Two distinct possibilities!

- (1) *Numerical differentiation* of f by using a difference formula:
- Let $\boldsymbol{e}_i = (0, 0, \ldots, 0, 1, 0, \ldots, 0)^{\mathrm{T}}$ be the unit vector in $\mathbb{R}^n$. Then,

$$\begin{aligned} f(\boldsymbol{x} + \alpha \boldsymbol{e}_i) &= f(\boldsymbol{x}) + \alpha \boldsymbol{e}_i^{\mathrm{T}} \nabla f(\boldsymbol{x}) + (\alpha^2/2) \boldsymbol{e}_i^{\mathrm{T}} \nabla^2 f(\boldsymbol{x}) \boldsymbol{e}_i + \ldots \\ &= f(\boldsymbol{x}) + \alpha \partial f(\boldsymbol{x}) / \partial x_i + (\alpha^2/2) \partial^2 f(\boldsymbol{x}) / \partial x_i^2 + \ldots \end{aligned}$$

- So, for small $\alpha > 0$,

$$\frac{\partial f(\boldsymbol{x})}{\partial x_i} \approx \frac{f(\boldsymbol{x} + \alpha \boldsymbol{e}_i) - f(\boldsymbol{x})}{\alpha} \quad \text{(forward difference)}$$

$$\frac{\partial f(\boldsymbol{x})}{\partial x_i} \approx \frac{f(\boldsymbol{x} + \alpha \boldsymbol{e}_i) - f(\boldsymbol{x} - \alpha \boldsymbol{e}_i)}{2\alpha} \quad \text{(central difference)}$$

- Value of α typically set to a function of the machine precision; if too large, we get a bad approximation of the partial derivative, while a too small value might result in numerical cancellation
- May work well *if* the simulation is *accurate*, otherwise bad derivative information. Requires *cheap* simulations!

- (2) *Derivative-free methods* are available. (Not counting subgradient methods, because they demand f to be convex!) Either builds explicit *models* $\hat{f}$ of the objective function by evaluating f at test points, or evaluates f at grid points that are moved around, shrunk or expanded. *Names: Nelder–Mead, Pattern search*
- Check hand-out!
- Alternative: create explicit algebraic (e.g., polynomial) model $\tilde{f}$ based on visited points $\boldsymbol{x}_k$; solve this problem with gradient methods; evaluate its optimum in the real problem (i.e., perform a simulation); update $\tilde{f}$ with the new information. $\Longrightarrow$ minimizes the number of simulations!

Conjugate gradient methods

- Algorithm for strictly convex quadratic programs to minimize $f(\boldsymbol{x}) := \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{q}^{\mathrm{T}}\boldsymbol{x}$, that is, solve $\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{q}$
- Non-quadratic extensions available
- Basic scheme:

$$\boldsymbol{p}_0 = -\nabla f(\boldsymbol{x}_0); \tag{3a}$$

$$\boldsymbol{p}_k = -\nabla f(\boldsymbol{x}_k) + \beta_k \boldsymbol{p}_{k-1}, \qquad k = 1, 2, \ldots, n-1, \tag{3b}$$

where

$$\beta_k = \frac{\nabla f(\boldsymbol{x}_k)^{\mathrm{T}} \nabla f(\boldsymbol{x}_k)}{\nabla f(\boldsymbol{x}_{k-1})^{\mathrm{T}} \nabla f(\boldsymbol{x}_{k-1})} \tag{3c}$$

- Use exact line search

- Crucial properties:

(a) Step k minimizes f over a k-dimensional manifold; after at most n steps we find the optimal solution

(b) All directions $\boldsymbol{p}_k$ are “conjugate”, that is, $\boldsymbol{p}_i^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{p}_j = 0$ for $i \neq j$ (Type of orthogonality; generation a type of Gram–Schmidt procedure applied to the negative gradients)

(c) Need only store the previous gradient to get new direction

(d) Strictly better convergence than steepest descent

(e) Direction vector $\boldsymbol{p}_i$ is an eigenvector corresponding to a largest eigenvalue λ_i not yet found

(f) Meaning: Takes care of most difficult part of the problem first. Less sensitive to the size of the condition number $\kappa(\boldsymbol{Q}) := \lambda_n/\lambda_1$ than steepest descent