# Advanced computational statistics, lecture 4

Frank Miller, Department of Computer and Information Science,
Linköping University

April 15, 2025

# Course schedule

- Topic 1: **Gradient based optimisation**
- Topic 2: **Stochastic gradient based optimisation**
- Topic 3: **Gradient free optimisation**
- Topic 4: **Optimisation with constraints**
- Topic 5: **EM algorithm and bootstrap**
- Topic 6: **Simulation of random variables**
- Topic 7: **Numerical and Monte Carlo integration; importance sampling**

Course homepage: http://www.adoptdesign.de/frankmillereu/adcompstat2025.html

Includes schedule, reading material, lecture notes, assignments

# Today’s schedule: Optimisation with constraints

- Equality constraints
  - Transformation to an unconstrained problem
  - Modification of iterative algorithm to handle constraints
  - Lagrange multipliers
- Inequality constraints
  - Karush–Kuhn–Tucker approach
  - penalty method
  - barrier method
- Subset constraint
- Combinatorial constrained optimisation

# Optimisation with equality constraints

- Optimisation problem:
  - $\boldsymbol{x}$ $p$-dimensional vector, $g$: $\mathbb{R}^p \to \mathbb{R}$ function
  - We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \max g(\boldsymbol{x})$
  - Subject to $h_i(\boldsymbol{x}^*) = 0,\ i = 1, \ldots, m$ (equality constraints)

# Optimisation with equality constraints

- Optimisation problem:
  - $\boldsymbol{x}$ $p$-dimensional vector, $g\colon \mathbb{R}^p \to \mathbb{R}$ function
  - We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \max g(\boldsymbol{x})$
  - Subject to $h_i(\boldsymbol{x}^*) = 0,\ i = 1, \dots, m$ (equality constraints)
- Approaches:
  - Transformation to an unconstrained problem (problem specific approach)
  - Modification of iterative algorithm to handle constraints (algorithm specific approach)
  - Lagrange multipliers (general approach)
- $\mathbb{S} = \{\boldsymbol{x} \in \mathbb{R}^p | h_i(\boldsymbol{x}) = 0,\ i = 1, \dots, m\}$ called <u>feasible points</u>

# Equality constraints: transformation

- Example: Cubic regression model for fertilizer-yield-relationship with fertilizer $x \in [0,1.2]$. Experiment planned with
  - proportion $w_1$ of observations using $x_1 = 0$,
  - proportion $w_2$ using $x_2 = 0.4$,
  - proportion $w_3$ using $x_3 = 0.8$,
  - proportion $w_4$ using $x_4 = 1.2$.
- Note that $w_1 + w_2 + w_3 + w_4 = 1$.
- Information matrix $\boldsymbol{M}$ (proportional to inverse of covariance matrix for $\left(\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3\right)^T$):
  $\boldsymbol{M} = \boldsymbol{X}^T \operatorname{diag}(w_1, \dots, w_4)\boldsymbol{X} = \sum_{i=1}^{4} w_i \boldsymbol{f}(x_i)\,\boldsymbol{f}(x_i)^T$ with $\boldsymbol{f}(x) = (1, x, x^2, x^3)^T$
- The D-optimal design maximises
  $g(\boldsymbol{w}) = \det\left(\sum_{i=1}^{4} w_i \boldsymbol{f}(x_i)\,\boldsymbol{f}(x_i)^T\right)$ subject to $h_1(\boldsymbol{w}) = 1 - \sum_{i=1}^{4} w_i = 0$

# Equality constraints: transformation

- The D-optimal design maximises
  $g(\boldsymbol{w}) = \det\left(\sum_{i=1}^{4} w_i \boldsymbol{f}(x_i)\, \boldsymbol{f}(x_i)^T\right)$ subject to $h_1(\boldsymbol{w}) = 1 - \sum_{i=1}^{4} w_i = 0$
- Transformation: $1 - \sum_{i=1}^{4} w_i = 0 \;\; \Rightarrow \;\; w_4 = 1 - w_1 - w_2 - w_3$
  $\tilde{g}(w_1, w_2, w_3) = \det\left(\sum_{i=1}^{3} w_i \boldsymbol{f}(x_i)\, \boldsymbol{f}(x_i)^T + (1 - w_1 - w_2 - w_3)\boldsymbol{f}(x_4)\boldsymbol{f}(x_4)^T\right)$

- The <u>constrained</u> optimisation problem
  max. $g(w_1, w_2, w_3, w_4)$ subj. to $h_1(w_1, w_2, w_3, w_4) = 1 - \sum_{i=1}^{4} w_i = 0$
  is equivalent to the <u>unconstrained</u> optimisation problem
  maximise $\tilde{g}(w_1, w_2, w_3)$.
- Solution with unconstrained optimisation: $(w_1, w_2, w_3) = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4})$, $w_4 = 1 - \frac{3}{4} = \frac{1}{4}$

# Equality constraints: modification of algorithms

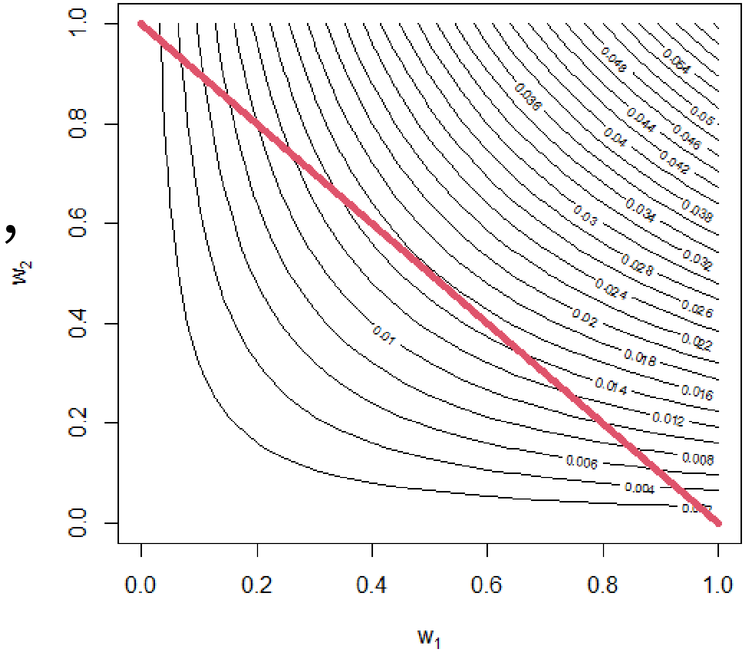
- Constrained optimisation problem:
  - $\boldsymbol{x}$ $p$-dimensional vector, $g$: $\mathbb{R}^p \to \mathbb{R}$ function
  - We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \max g(\boldsymbol{x})$
  - Subject to $\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b} = \boldsymbol{0}$, $\boldsymbol{A} \in \mathbb{R}^{m \times p}, \boldsymbol{b} \in \mathbb{R}^m$ (linear equality constraints)
- Example: Particle Swarm Optimisation (see L3)
- Movement of particle $i$ at iteration $t$+1:
  - $\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)} + \boldsymbol{v}_i^{(t+1)}$
  - $\boldsymbol{v}_i^{(t+1)} = w\boldsymbol{v}_i^{(t)} + c_1 R_1^{(t+1)} (\boldsymbol{p}_{\text{best}, i}^{(t)} - \boldsymbol{x}_i^{(t)}) + c_2 R_2^{(t+1)} (\boldsymbol{g}_{\text{best}}^{(t)} - \boldsymbol{x}_i^{(t)})$

**Scalar random variables (SPSO2011),** not random vectors (SPSO2007)

- $R_1^{(t+1)}$ and $R_2^{(t+1)}$ are uniformly distributed, `runif()`
- Ensure that $\boldsymbol{A}\boldsymbol{x}_i^{(0)} = \boldsymbol{b}$ and $\boldsymbol{A}\boldsymbol{v}_i^{(0)} = \boldsymbol{0}$, then $\boldsymbol{A}\boldsymbol{x}_i^{(t)} = \boldsymbol{b}$ for all $i$ and $t$

# Equality constraints: Lagrange multipliers

- Example: D-optimal design for quadratic regression without intercept. Experiment planned on $x \in [0,1]$ with
  - prop. $w_1$ of observations using $x_1 = 0.5$,
  - prop. $w_2$ using $x_2 = 1$,
  - $w_1 + w_2 = 1$.




- $g(\boldsymbol{w}) = \det\left( w_1 \begin{pmatrix} \frac{1}{4} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{16} \end{pmatrix} + w_2 \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \right)$

- $h(\boldsymbol{w}) = 1 - w_1 - w_2$

# Equality constraints: Lagrange multipliers

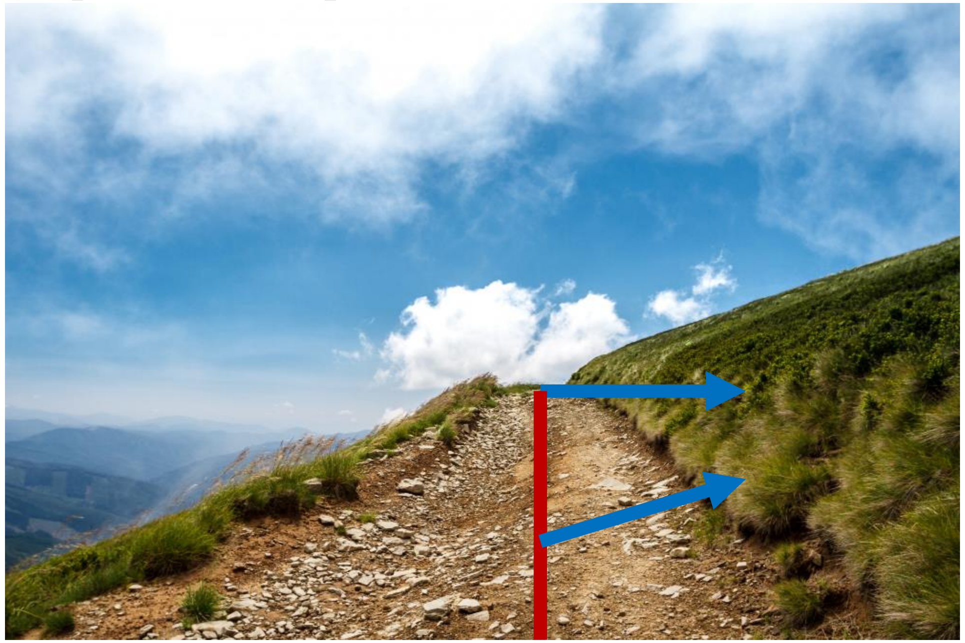
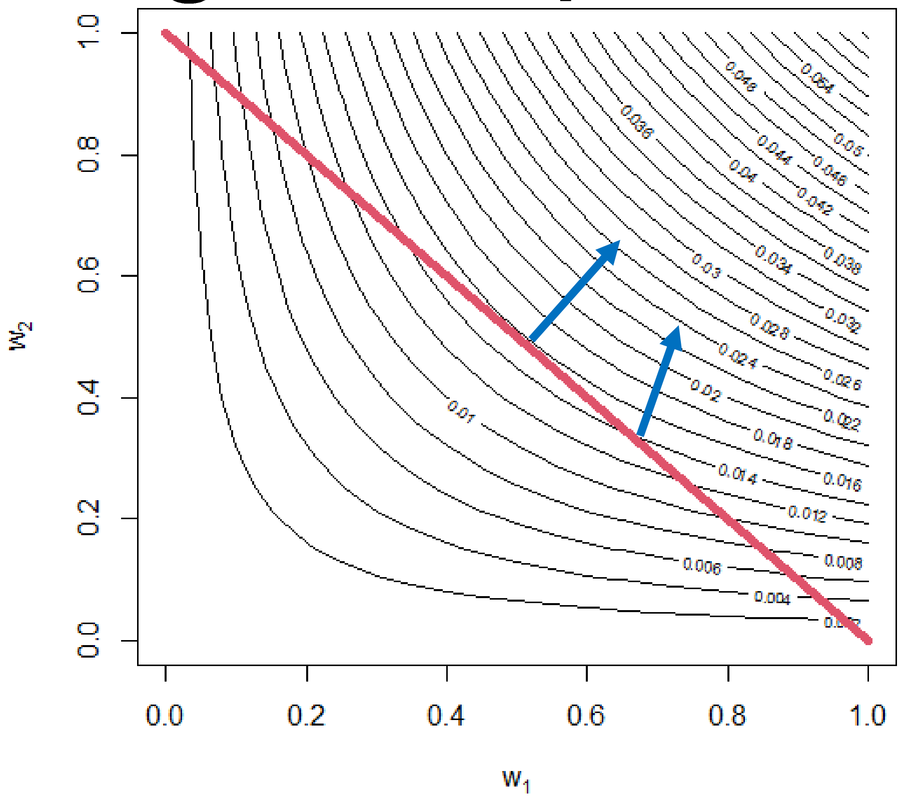


Image by cookie_studio on Freepik



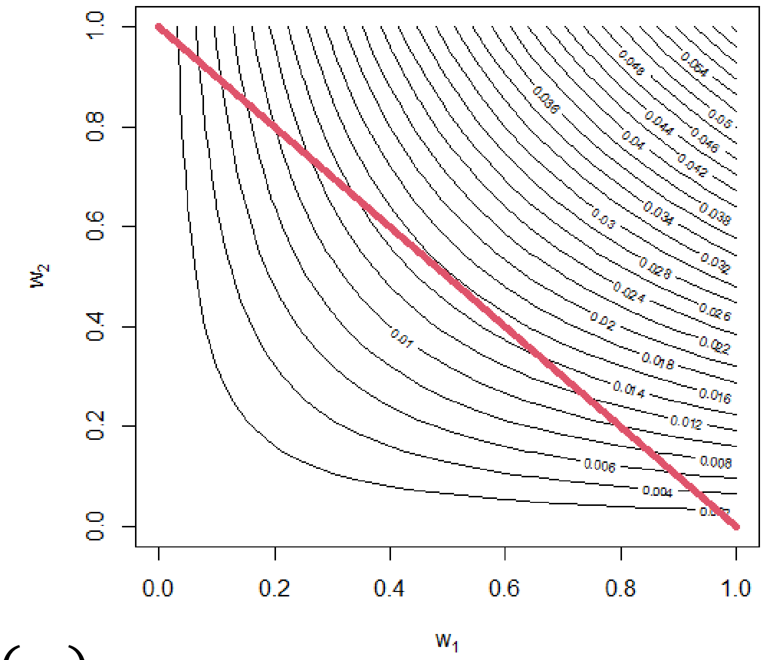

- Feasible points $\boldsymbol{w}$ ($h(\boldsymbol{w}) = 0$)
- Direction of steepest ascent, $g'(\boldsymbol{w})$

These two are orthogonal at constrained max.; direction orthogonal to feasible points is $h'(\boldsymbol{w})$

# Equality constraints: Lagrange multipliers

- $g(\boldsymbol{w}) = \det\left( w_1 \begin{pmatrix} \frac{1}{4} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{16} \end{pmatrix} + w_2 \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \right)$

- $h(\boldsymbol{w}) = 1 - w_1 - w_2$
- $g'(\boldsymbol{w})$ direction of steepest ascent
- $h'(\boldsymbol{w}) = (-1, -1)^T$ (orthogonal to feasible points)




- Condition for constrained maximum: $g'(\boldsymbol{w}) = \lambda\, h'(\boldsymbol{w})$
- $g'(\boldsymbol{w}) - \lambda\, h'(\boldsymbol{w}) = 0$
- Define $\mathcal{L}(\boldsymbol{w}, \lambda) = g(\boldsymbol{w}) - \lambda\, h(\boldsymbol{w})$ and determine stationary point

# Equality constraints: Lagrange multipliers

- Constrained optimisation problem:
  - $\boldsymbol{x}$ $p$-dimensional vector, $g$: $\mathbb{R}^p \to \mathbb{R}$ function
  - We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \max g(\boldsymbol{x})$
  - Subject to $h_i(\boldsymbol{x}^*) = 0,\ i = 1, \dots, m$ (equality constraints)
- Lagrange:
  Let $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = g(\boldsymbol{x}) - \boldsymbol{\lambda}^T \boldsymbol{h}(\boldsymbol{x})$, $\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), \dots, h_m(\boldsymbol{x}))^T$, $\boldsymbol{\lambda} \in \mathbb{R}^m$ and $g, h_1, \dots, h_m$ are continuously differentiable. If $g$ has a local maximum at some point $\boldsymbol{x}^*$ with $\boldsymbol{h}(\boldsymbol{x}^*) = \boldsymbol{0}$ (i.e. in the constrained maximisation problem) and at which the gradients of $h_1, \dots, h_m$ are linearly independent, then there exists a $\boldsymbol{\lambda}$ such that gradient $\mathcal{L}'(\boldsymbol{x}^*, \boldsymbol{\lambda}) = \boldsymbol{0}$ (i.e. stationary point in the unconstrained problem).

# Equality constraints: Lagrange multipliers

- Constrained optimisation problem:
  - $\boldsymbol{x}$ $p$-dimensional vector, $g$: $\mathbb{R}^p \to \mathbb{R}$ function
  - We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \max g(\boldsymbol{x})$
  - Subject to $h_i(\boldsymbol{x}^*) = 0,\ i = 1, \dots, m$ (equality constraints)
- Unconstrained problem:
  Search stationary point $(\boldsymbol{x}^*, \boldsymbol{\lambda})$ of $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = g(\boldsymbol{x}) - \boldsymbol{\lambda}^T \boldsymbol{h}(\boldsymbol{x})$.
- Note:
  - $\frac{\partial}{\partial \lambda_i}\mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\lambda}) = 0$ ensures $h_i(\boldsymbol{x}^*) = 0$
  - $\frac{\partial}{\partial x_i}\mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\lambda}) = 0$ ensures that gradient $g'(\boldsymbol{x}^*)$ is orthogonal to the set $\mathbb{S}$ of feasible points at $\boldsymbol{x} = \boldsymbol{x}^*$

# Equality constraints: Comparison

- Recall example about D-optimal design for quadratic regression without intercept; optimal values for $w_1$ and $w_2$ are of interest ($p = 2, m = 1$).

In general:

- Transformation method: optimise over $w_1$ — dim = $p - m$,
- Modification of algorithm: optimise over $(w_1, w_2)$ — dim = $p$,
- Lagrange multiplier method: search space is $(w_1, w_2, \lambda)$ — dim = $p + m$

- If transformation method possible and not too complicated, it has potential to deliver results fastest
- Transformation and modification methods require creativity; Lagrange can be applied generally

# Optimisation with inequality constraints

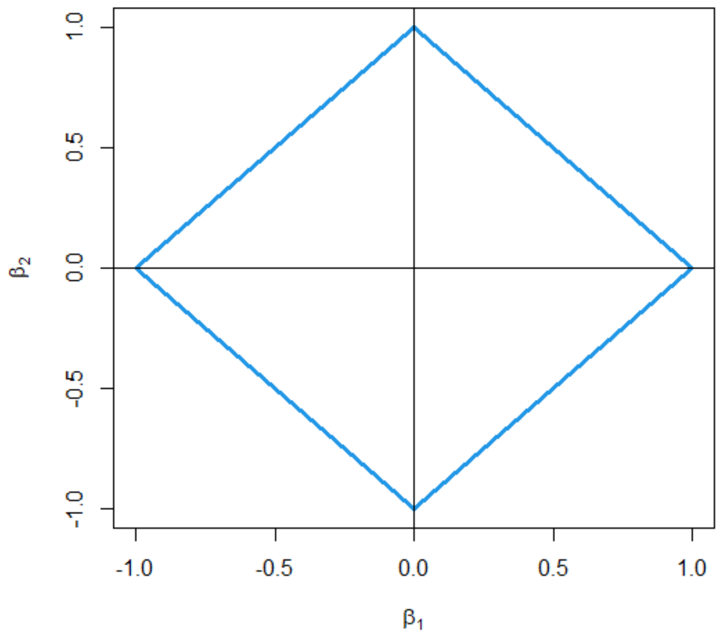
- Constrained optimisation problem:
  - $\boldsymbol{x}$ $p$-dimensional vector, $g$: $\mathbb{R}^p \rightarrow \mathbb{R}$ function
  - We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \max g(\boldsymbol{x})$
  - Subject to $h_i(\boldsymbol{x}^*) = 0,\ i = 1, \dots, m$
  - and $q_i(\boldsymbol{x}^*) \leq 0,\ i = 1, \dots, n$ (inequality constraints)
- Set of feasible points $\mathbb{S} = \{\boldsymbol{x} \in \mathbb{R}^p | h_i(\boldsymbol{x}) = 0,\ i = 1, \dots, m; q_i(\boldsymbol{x}) \leq 0,\ i = 1, \dots, n\}$

- An inequality constraint $q_\mathrm{i}(\boldsymbol{x})$ is called active, if $q_\mathrm{i}(\boldsymbol{x}^*) = 0$
- If it is not active ($q_\mathrm{i}(\boldsymbol{x}^*) < 0$), $\boldsymbol{x}^*$ is a local optimum of the unconstrained optimisation problem

# Inequality constraints – lasso example

- Lasso's objective function to minimise:
  $g(\widehat{\boldsymbol{\beta}}) = \left\|\boldsymbol{X}\widehat{\boldsymbol{\beta}} - \boldsymbol{y}\right\|^2 + \lambda \sum_{i=1}^{p} \left|\widehat{\beta}_i\right|$

- Alternatively, one can solve the constrained problem:
  minimise: $g(\widehat{\boldsymbol{\beta}}) = \left\|\boldsymbol{X}\widehat{\boldsymbol{\beta}} - \boldsymbol{y}\right\|^2$
  subject to $\left\|\widehat{\boldsymbol{\beta}}\right\|_1 = \sum_{i=1}^{p} \left|\widehat{\beta}_i\right| \leq t$




- For $p = 2$ and $t = 1$, the set of feasible points $\mathbb{S} = \{\widehat{\boldsymbol{\beta}} \in \mathbb{R}^p \,|\, \sum_{i=1}^{p} |\widehat{\beta}_i| \leq t\}$ is inside of the blue area

# Optimisation with inequality constraints

- Approaches to handle inequality constraints:
  - Generalisation of Lagrange multipliers (Karush–Kuhn–Tucker approach)
  - penalty method
  - barrier method (also called: interior-point method)

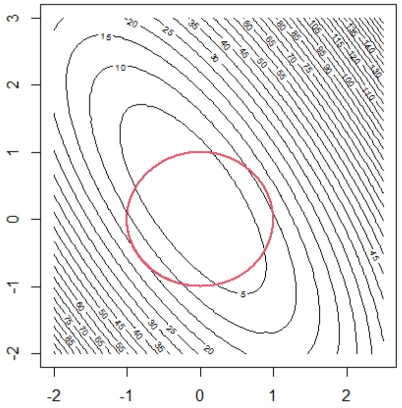
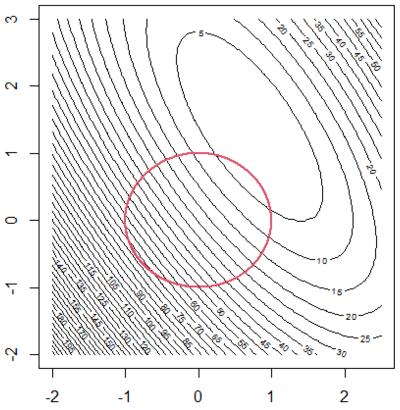
# Inequality constraints: Karush-Kuhn-Tucker appr.

- Constrained optimisation problem:
  - $\boldsymbol{x}$ $p$-dimensional vector, $g\colon \mathbb{R}^p \to \mathbb{R}$ function
  - We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \max g(\boldsymbol{x})$
  - Subject to $h_i(\boldsymbol{x}^*) = 0,\ i = 1, \dots, m$
  - and $q_i(\boldsymbol{x}^*) \leq 0,\ i = 1, \dots, n$ (inequality constraints)

- Karush–Kuhn–Tucker (KKT) approach uses generalised Lagrangian $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = g(\boldsymbol{x}) - \boldsymbol{\lambda}^T \boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{\mu}^T \boldsymbol{q}(\boldsymbol{x})$ with $\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), \dots, h_m(\boldsymbol{x}))^T, \boldsymbol{\lambda} \in \mathbb{R}^{\boldsymbol{m}}, \boldsymbol{q}(\boldsymbol{x}) = (q_1(\boldsymbol{x}), \dots, q_n(\boldsymbol{x}))^T, \boldsymbol{\mu} \in \mathbb{R}^{\boldsymbol{n}}$
- Instead of above constrained optimisation, search stationary point $(\boldsymbol{x}^*, \boldsymbol{\lambda}, \boldsymbol{\mu} \geq \boldsymbol{0})$ of $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = g(\boldsymbol{x}) - \boldsymbol{\lambda}^T \boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{\mu}^T \boldsymbol{q}(\boldsymbol{x})$.
  For $\boldsymbol{x}^*$ being a solution of the constrained problem, following condition required:
  "for all $i = 1, \dots, n$: $q_i(\boldsymbol{x}^*) = 0$ or $\mu_i = 0$"

# Inequality constraints: KKT, example

- **Constrained LS-<u>min</u>imisation**:
  - $\boldsymbol{x}$ $p$-dim., $g$: $\mathbb{R}^p \to \mathbb{R}$, $g(\boldsymbol{x}) = \left|\left|\boldsymbol{Ax} - \boldsymbol{b}\right|\right|_2^2$ $\quad\quad\quad\quad$ $\|\boldsymbol{x}\|_2 \leq 1$
  - $g(\boldsymbol{x}) = \min g(\boldsymbol{x})$ subject to $q_1(\boldsymbol{x}) = ||\boldsymbol{x}||_2^2 - 1 \leq 0$ (inequality constraint)
- Generalised Lagrangian (KKT): $\mathcal{L}(\boldsymbol{x}, \mu) = \left|\left|\boldsymbol{Ax} - \boldsymbol{b}\right|\right|_2^2 + \mu(\|\boldsymbol{x}\|_2^2 - 1)$ with $\mu \geq 0$
- $\frac{\partial}{\partial x}\mathcal{L}(\boldsymbol{x}, \mu) = \boldsymbol{A^T A x} - \boldsymbol{A^T b} + 2\mu\boldsymbol{x}$; setting this to 0 gives $\boldsymbol{x} = (\boldsymbol{A^T A} + 2\mu\boldsymbol{I})^{-1}\,\boldsymbol{A^T b}$
- $\frac{\partial}{\partial \mu}\mathcal{L}(\boldsymbol{x}, \mu) = 1 - \|\boldsymbol{x}\|_2^2$

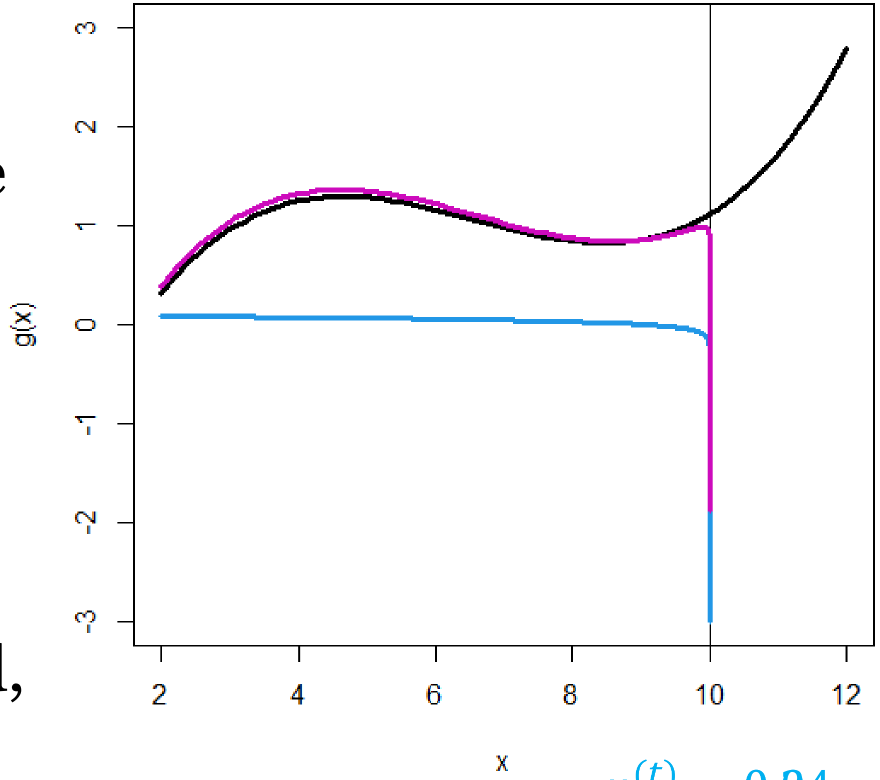
# Inequality constraints: penalty and barrier methods

- Constrained optimisation problem:
  - $\boldsymbol{x}$ $p$-dimensional vector, $g$: $\mathbb{R}^p \to \mathbb{R}$ function
  - We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \max g(\boldsymbol{x})$
  - Subject to $q_i(\boldsymbol{x}^*) \leq 0,\ i = 1, \dots, n$ (inequality constraints)
- Idea: Modify $g$ to $\tilde{g}$ such that the algorithm finds only local maxima which fulfil $q_i(\boldsymbol{x}^*) \leq 0,\ i = 1, \dots, n$, even if optimisation done unconstrained

- Penalty methods: Set $\tilde{g} = g$ on $\mathbb{S} = \{\boldsymbol{x} | q_i(\boldsymbol{x}) \leq 0,\ i = 1, \dots, n\}$ and add a (negative) penalty if $q_i(\boldsymbol{x}) > 0$ for some $i$
- Barrier methods: Set $\tilde{g} = -\infty$ if $q_i(\boldsymbol{x}) > 0$ for some $i$ and $g$ is modified on $\mathbb{S} = \{\boldsymbol{x} | q_i(\boldsymbol{x}) \leq 0,\ i = 1, \dots, n\}$

# Inequality constraints: Barrier method

- Example: maximise $g(x)$ on range $x \leq 10$
- Add barrier function $\mu^{(t)} b(x)$
- $\tilde{g}(x) = g(x) + \mu^{(t)} b(x)$ should be small close to 10 for $x < 10$, and $-\infty$ for $x > 10$
- Log barrier: $b(x) = \log(10 - x)$
- Solve maximisation for $\tilde{g}(x)$
- Adapt barrier with smaller $\mu^{(t)}$
- If $\mu^{(t)} \to 0$, local maxima of $g$ can be detected, both at the boundary and in the interior

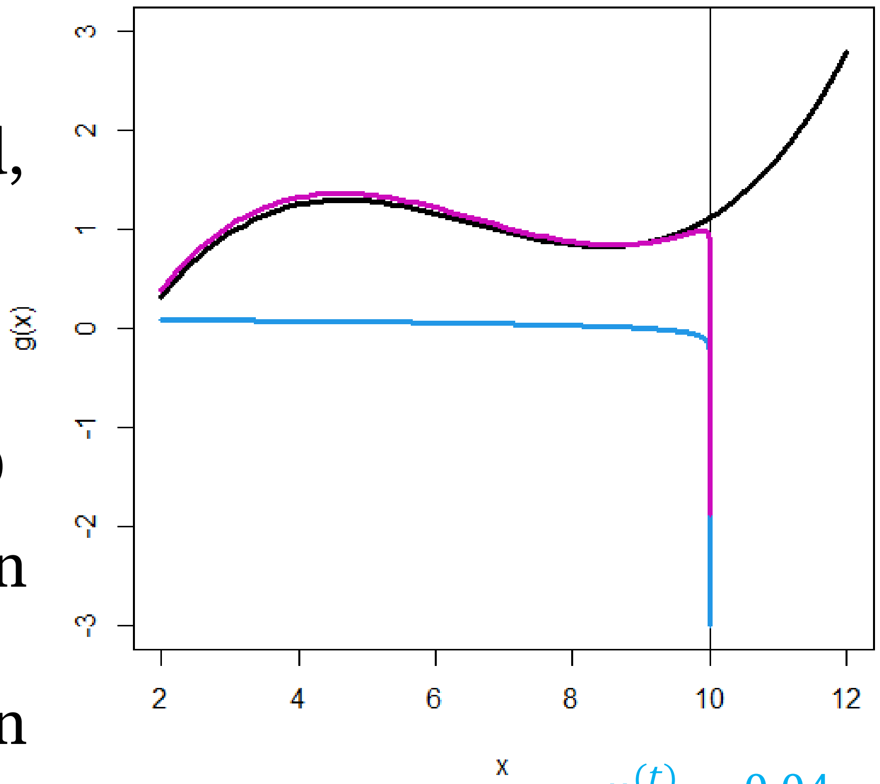



$\mu^{(t)} = 0.04$

Two 2d-animations: http://apmonitor.com/me575/index.php/Main/InteriorPointMethod

# Inequality constraints: Barrier method

- Example: maximise $g(x)$ on range $x \leq 10$
- Adapt barrier with smaller $\mu^{(t)}$
- If $\mu^{(t)} \to 0$, local maxima of $g$ can be detected, both <u>at the boundary</u> and <u>in the interior</u>
- Use a sequence $\mu^{(1)} > \mu^{(2)} > \cdots > \mu^{(k)} > \cdots$ with $\mu^{(t)} \to 0$:
  - Solution for optimisation with $\mu^{(1)}$ is $x^{(*1)}$
  - Use $x^{(*1)}$ as starting value for optimisation with $\mu^{(2)}$; solution is $x^{(*2)}$
  - Use $x^{(*2)}$ as starting value for optimisation with $\mu^{(3)}$; solution is $x^{(*3)}$
  - …




$\mu^{(t)} = 0.04$

# Linear inequality constraints: R-function `constrOptim`

- Constrained optimisation problem:
  - $\boldsymbol{x}$ $p$-dimensional vector, $g\colon\ \mathbb{R}^p \to \mathbb{R}$ function
  - We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \max g(\boldsymbol{x})$
  - Subject to $\boldsymbol{U}\boldsymbol{x}^* - \boldsymbol{c} \geq \boldsymbol{0}$, $\boldsymbol{U} \in \mathbb{R}^{n\times p}, \boldsymbol{c} \in \mathbb{R}^n$ (linear inequality constraints; rows of $\boldsymbol{U}$ are $\boldsymbol{u}_i^T$)

- The R-function **`constrOptim`** uses log barrier functions
- **`constrOptim`** calls repeatedly **`optim`** for function $\tilde{g}$ with barrier; barrier adapted between iterations: $\mu^{(t)}$ decreases
- E.g: $\tilde{g}(\boldsymbol{x}) = g(\boldsymbol{x}) + \mu^{(t)} \sum_{i=1}^{n} \log(\boldsymbol{u}_i^T \boldsymbol{x} - c_i)$ (for maximisation; $g(\boldsymbol{x}) - \mu^{(t)}$ ... for minimisation)

# Linear inequality constraints: barrier method

- Example: Quadratic regression for fertilizer-yield-relationship with fertilizer $x \in [0,1.2]$. Experiment planned with
  - proportion $w_i$ of observations using $x_i \in [0,1.2]$ (can be chosen by experimenter), $i = 1,2,3$; $w_3 = 1 - w_1 - w_2$.
- Parameters to be optimised: $\boldsymbol{y} = (x_1, x_2, x_3, w_1, w_2)^T$
- D-optimal design maximises $g(\boldsymbol{y}) = \det\left(\sum_{i=1}^{3} w_i \boldsymbol{f}(x_i)\, \boldsymbol{f}(x_i)^T\right)$ subject to $x_i \geq 0,\ 1.2 - x_i \geq 0,\ i = 1,2,3,\ w_1 \geq 0,\ w_2 \geq 0,\ 1 - w_1 - w_2 \geq 0$
- Construct $\boldsymbol{U}$ and $\boldsymbol{c}$ such that constraints can be written as $\boldsymbol{U}\boldsymbol{y} - \boldsymbol{c} \geq \boldsymbol{0}$

# Linear inequality constraints: barrier method

- $\boldsymbol{y} = (x_1, x_2, x_3, w_1, w_2)^T$, $w_3 = 1 - w_1 - w_2$
- D-optimal design maximises $g(\boldsymbol{y}) = \det\left(\sum_{i=1}^{3} w_i \boldsymbol{f}(x_i)\, \boldsymbol{f}(x_i)^T\right)$ subject to $x_i \geq 0$, $1.2 - x_i \geq 0$, $i = 1,2,3$, $w_1 \geq 0$, $w_2 \geq 0$, $1 - w_1 - w_2 \geq 0$
- $\boldsymbol{U}\boldsymbol{y} - \boldsymbol{c} \geq \boldsymbol{0}$ with

$$\boldsymbol{U} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 \end{pmatrix}, \boldsymbol{c} = \begin{pmatrix} 0 \\ -1.2 \\ 0 \\ -1.2 \\ 0 \\ -1.2 \\ 0 \\ 0 \\ -1 \end{pmatrix}$$

# Linear inequality constraints: R-function `constrOptim`

- R-code:
- ```
  U       <- matrix(0, nrow=9, ncol=5)
  U[1,1] <- U[3,2] <- U[5,3] <- U[7,4] <- U[8,5] <- 1
  U[2,1] <- U[4,2] <- U[6,3] <- U[9,4] <- U[9,5] <- -1
  d       <- c(rep(c(0, -1.2), 3), 0, 0, -1)

  startv <- c(0.2, 0.3, 0.4, 0.2, 0.2)

  # Nelder-Mead as inner optimisation method:
  res    <- constrOptim(startv, f=g, grad=NULL, ui=U, ci=d,
                        control=list(fnscale=-1))
  round(res$par, 3)
  ```

Python: scipy.optimize.minimize
Julia: optimize! in JuMP, using Ipopt
Matlab: fmincon

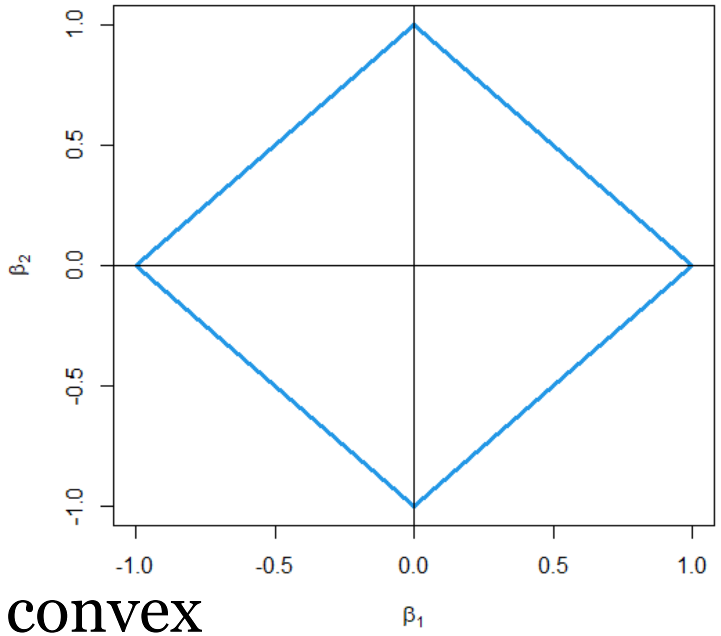
- Result: `0.000 0.597 1.200 0.331 0.333`
- Note: In this case, the solution can also be calculated algebraically (optimal design theory)
  ```

# Linear inequality constraints: barrier method

- Limitations of barrier method (Lange, 2010, page 301):
  - Iterations within iterations necessary
  - No obvious choice how fast $\mu^{(t)}$ should go to 0
  - A too small value $\mu^{(t)}$ can lead to numerical instability

# Optimisation with a subset constraint

- Optimisation problem with closed and convex subset constraint:
  - $\boldsymbol{x}$ $p$-dimensional vector, $g$: $\mathbb{R}^p \rightarrow \mathbb{R}$ continuously differentiable function
  - We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \max g(\boldsymbol{x})$
  - Subject to $\boldsymbol{x} \in \Omega$ with $\Omega$ being a closed and convex set
- Set of feasible points $\mathbb{S} = \{\boldsymbol{x} \in \Omega \subseteq \mathbb{R}^p\}$



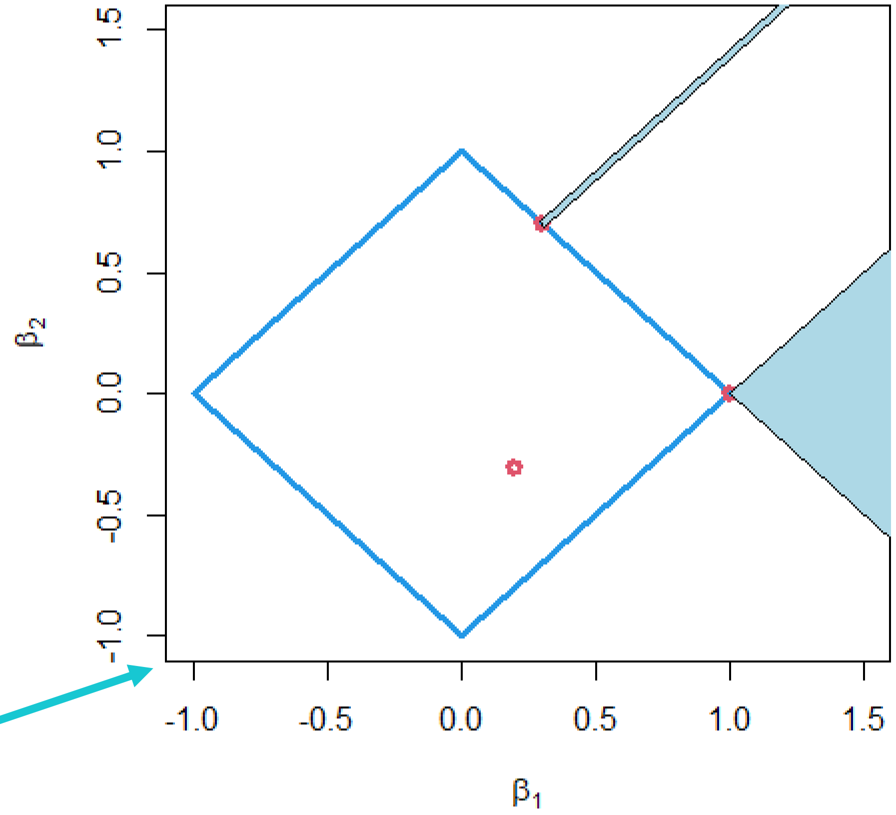

- Note that in the constrained lasso example, $\Omega = \left\{\hat{\beta} \middle| \|\widehat{\boldsymbol{\beta}}\|_1 = \sum_{i=1}^{p} |\widehat{\beta}_i| \leq t\right\}$ which is closed and convex

# Optimisation with a subset constraint

- Definition: Let $\Omega$ be a closed and convex set. The normal cone at $\boldsymbol{x} \in \Omega$ is defined as
  $N_\Omega(\boldsymbol{x}) =$
  $\{\boldsymbol{d} \in \mathbb{R}^p \,|\, \boldsymbol{d}^\mathrm{T}(\boldsymbol{y} - \boldsymbol{x}) \leq 0 \text{ for all } \boldsymbol{y} \in \Omega\}$

- Note: $\boldsymbol{d}^\mathrm{T}\boldsymbol{z} \leq 0$ means that the angle between $\boldsymbol{d}$ and $\boldsymbol{z}$ is at least 90 degrees

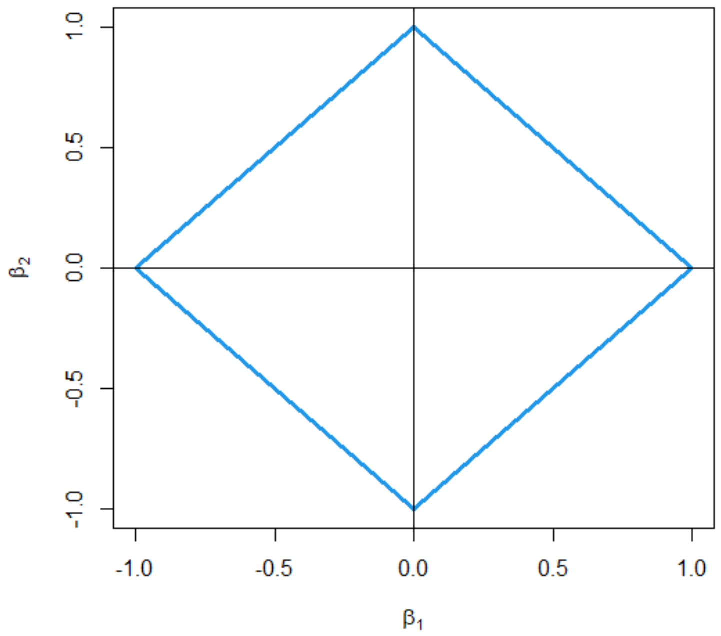
- Some examples for $N_\Omega(\boldsymbol{x})$ ($\boldsymbol{x}$ = red dots)



# Optimisation with a subset constraint

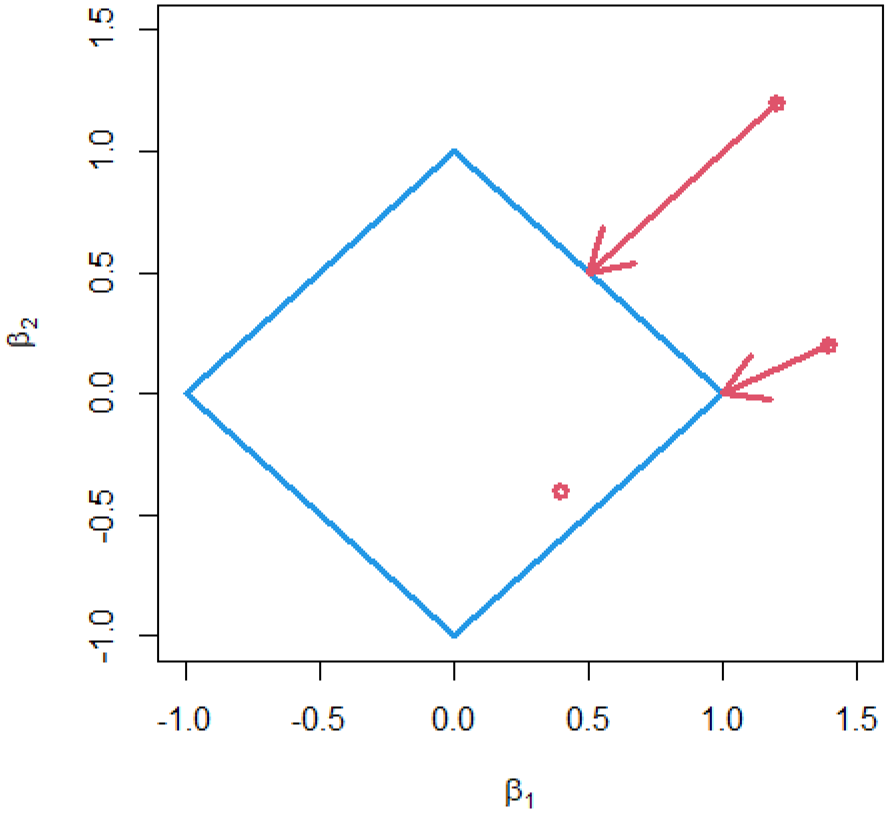
- Theorem: If $\boldsymbol{x}^* \in \Omega$ is a local maximum in the optimisation problem with a closed and convex subset constraint, then $g'(\boldsymbol{x}^*) \in N_\Omega(\boldsymbol{x}^*)$.

- Corollary: If $g$ is a concave function and we consider the optimisation problem with a closed and convex subset constraint, then:
  $\boldsymbol{x}^* \in \Omega$ is a local maximum $\Leftrightarrow g'(\boldsymbol{x}^*) \in N_\Omega(\boldsymbol{x}^*)$.



# Optimisation with a subset constraint

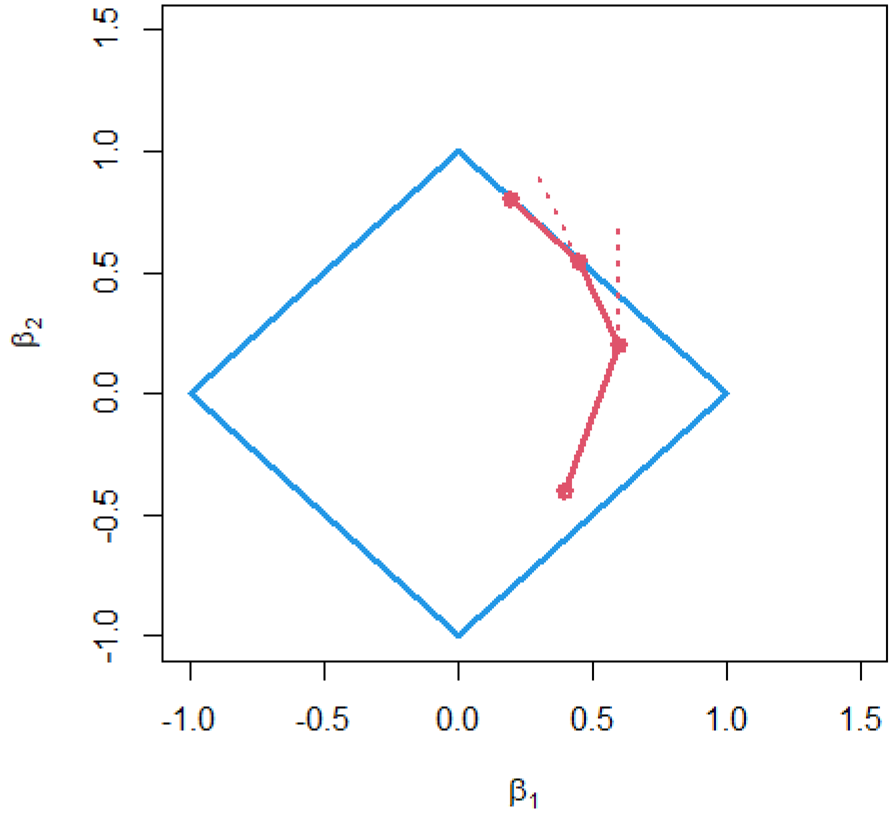
- Definition: For a closed and convex set $\Omega$, we define the Euclidian projection as
$$P_\Omega(\boldsymbol{x}) = \arg\min_{\mathbf{z}\in\Omega}\{||\boldsymbol{z} - \boldsymbol{x}||\}$$



# Optimisation with a subset constraint

Projected gradient algorithm

- Start with some $\boldsymbol{x}^{(0)} \in \Omega$.
- For given $\boldsymbol{x}^{(k)}$, compute next iteration $\boldsymbol{x}^{(k+1)}$ as:
  - $\boldsymbol{x}^{(k+1)} = P_\Omega\left(\boldsymbol{x}^{(k)} + \alpha_k g'\left(\boldsymbol{x}^{(k)}\right)\right)$
- Until $\boldsymbol{x}^{(k)}$ and $\boldsymbol{x}^{(k+1)}$ are close and fulfil a stopping criterion



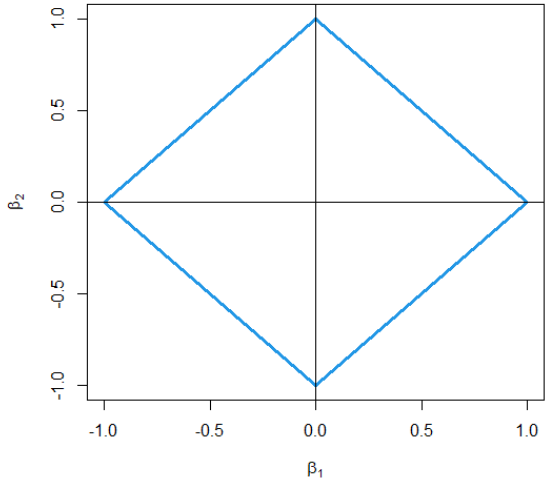

- If $g$ is Lipschitz-smooth, one can choose $\alpha_k = \frac{1}{L}$, otherwise apply back-tracking

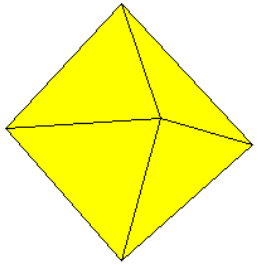
# Optimisation with a subset constraint

- The projected gradient algorithm generalizes the steepest ascent/descent algorithm to handle a subset constraint

- In the projected gradient algorithm, the Euclidian projection is computed,
$$P_\Omega\left(\boldsymbol{x}^{(k)} + \alpha_k g'\left(\boldsymbol{x}^{(k)}\right)\right)$$
- A requirement for the algorithm is that this computation is feasible and not a more complicated minimisation problem than optimising $g$ itself …

# Euclidian projection for lasso



- In the 2d-lasso-case, the Euclidian projection can be computed in an ad-hoc way
- For lasso in higher dimensions, one can do Euclidian projection onto the $L_1$-norm ball (see Condat, 2016; Duchi et al., 2008; Held et al., 1974) and some R-code on the course homepage

Condat L (2016). Fast projection onto the simplex and the ℓ1 ball. Mathematical Programming, Series A, 158, 575–585.
Duchi J, Shalev-Shwartz S, Singer Y, Chandra T (2008). Efficient Projections onto the ℓ1-Ball for Learning in High Dimensions. International Conference on Machine Learning (ICML).
Held M, Wolfe P, Crowder H (1974). Validation of subgradient optimization. Mathematical Programming 6, 62–88.

- If only some coordinates, $I \subseteq \{1, \ldots, p\}$, are regularised,

$$\Omega = \{\boldsymbol{b} \mid \textstyle\sum_{i \in I} |b_i| \leq t\}$$

- Then, we can apply Euclidian projection onto $L_1$-norm in the smaller space $\mathbb{R}^{|I|}$ (for coordinates $i \in I$ only), and keep $b_i$ for $i \notin I$ when computing $P_\Omega(\boldsymbol{b})$

# Optimisation with a subset constraint

Frank-Wolfe algorithm

- Start with some $\boldsymbol{x}^{(0)} \in \Omega$.
- For given $\boldsymbol{x}^{(k)}$, compute next iteration $\boldsymbol{x}^{(k+1)}$ as:
  - $\overline{\boldsymbol{x}}^{(k)} = \text{argmax}_{\overline{\boldsymbol{x}} \in \Omega}\, \boldsymbol{g}'\left(\boldsymbol{x}^{(k)}\right)^T \overline{\boldsymbol{x}}$
  - $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k\left(\overline{\boldsymbol{x}}^{(k)} - \boldsymbol{x}^{(k)}\right)$

argmin instead of argmax
for a minimisation problem

- Until $\boldsymbol{x}^{(k)}$ and $\boldsymbol{x}^{(k+1)}$ are close and fulfil a stopping criterion

- Sublinear convergence is ensured for convex, L-smooth function $g$ and $\Omega$ closed bounded convex set when steplength $\alpha_k = 2/(k+2)$ is used, see Theorem 7.9 of Wright and Recht (2022).

# Combinatorial constrained optimisation

# Recall L3 and Exercise 3.3: Maximising information of experimental designs

- Regression model $\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}$, $\mathrm{Cov}(\widehat{\boldsymbol{\beta}}) = (\boldsymbol{X}^T\boldsymbol{X})^{-1} \cdot const$
- Example: cubic regression,

$$y_i = \beta_0 + \beta_1 w_i + \beta_2 w_i^2 + \beta_3 w_i^3 + \varepsilon_i, \quad \boldsymbol{X} = \begin{pmatrix} 1 & w_1 & w_1^2 & w_1^3 \\ 1 & w_2 & w_2^2 & w_2^3 \\ \dots & \dots & \dots & \dots \\ 1 & w_n & w_n^2 & w_n^3 \end{pmatrix}$$

  $w_i$ can be chosen in [-1, 1], but practical circumstances require here a distance between design points of 0.05
- Therefore, we allow design points {-1, -0.95, -0.9, …, 1} a~~nd at most one observation can be done at each point~~
  $\mathbb{S} = \{(n_1, \dots, n_{41}), n_i \in \mathbb{N}_0\}$
- A design can be represented by a vector in ~~$\mathbb{S} = \{0, 1\}^{41}$~~ where 0 means that no observation is done at a design point and 1 means that one observation is made there
- Each observation has a cost; and we want to minimise the ~~penalized~~ D-optimality

$$\cancel{\#observations * 0.2} - \log\left(\det\left(\boldsymbol{X}^T\boldsymbol{X}\right)\right)$$

  for a given total sample size $n$

# Constrained optimisation to determine design

- Regression model $\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}$, $\text{Cov}\left(\widehat{\boldsymbol{\beta}}\right) = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-\mathbf{1}} \cdot const$
- Example: cubic regression,

  $$y_i = \beta_0 + \beta_1 w_i + \beta_2 w_i^2 + \beta_3 w_i^3 + \varepsilon_i, \quad \boldsymbol{X} = \begin{pmatrix} 1 & w_1 & w_1^2 & w_1^3 \\ 1 & w_2 & w_2^2 & w_2^3 \\ \dots & \dots & \dots & \dots \\ 1 & w_n & w_n^2 & w_n^3 \end{pmatrix}$$

  $w_i$ can be chosen in [-1, 1], but practical circumstances require here a distance between design points of 0.05; hence, we allow design points {-1, -0.95, -0.9, …, 1}
- A design can be represented (coded) in different ways, e.g.,
  - by a vector in $\mathbb{S}$ =$\{(n_1, \dots, n_{41}), n_i \in \mathbb{N}_0\}$ with $n_1$ being number of observations made at $w_i$
  - by a (sorted) vector $(w_1, \dots, w_n)$ in $\{-1, -0.95, -0.9, \dots, 1\}^n$

# Constrained optimisation to determine design

- Regression model $\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}$, $\mathrm{Cov}(\widehat{\boldsymbol{\beta}}) = (\boldsymbol{X}^T\boldsymbol{X})^{-1} \cdot const$
- Example: cubic regression,

  $$y_i = \beta_0 + \beta_1 w_i + \beta_2 w_i^2 + \beta_3 w_i^3 + \varepsilon_i, \quad \boldsymbol{X} = \begin{pmatrix} 1 & w_1 & w_1^2 & w_1^3 \\ 1 & w_2 & w_2^2 & w_2^3 \\ \dots & \dots & \dots & \dots \\ 1 & w_n & w_n^2 & w_n^3 \end{pmatrix}$$

  $w_i$ can be chosen in [-1, 1], but practical circumstances require here a distance between design points of 0.05; hence, we allow design points {-1, -0.95, -0.9, …, 1}
- We use the representation as vectors in $\mathbb{S} = \{(n_1, \dots, n_{41}), n_i \in \mathbb{N}_0\}$ with $n_1$ being number of observations made at $w_i$
- We have a restricted budget allowing for $n$ observations, i.e. $\sum_{i=1}^{41} n_i = n$.
- We want to minimise the D-criterion $-\log\left(\det(\boldsymbol{X}^T\boldsymbol{X})\right)$

# Constrained optimisation to determine design

- We can easily adjust the simulated annealing algorithm for combinatorial optimisation to handle the equality constraint $\sum_{i=1}^{41} n_i = n$:
  - Start with a design fulfilling the constraint
  - Define neighbourhood of a design such that all neighbours fulfil restriction (proposal distribution has probability 1 on designs with $\sum_{i=1}^{41} n_i = n$)
  - An intuitive possibility is to **exchange** observations:

    (2, 0, 0, 4, 5, 0, 0, 0, 3, 1, 0, ..., 0, 4) →
    (2, 1, 0, 4, 4, 0, 0, 0, 3, 1, 0, ..., 0, 4)
  - Search randomly a location (here of the 41 $w_i$'s) which has $n_i > 0$ where an observation is removed and another location where one is added

# Constrained optimisation to determine design

- Start design fulfilling constraint

```
des     <- rep(0, 41)
indices <- 1:41
for (i in 1:n){
  ind      <- sample(indices, size=1)
  des[ind] <- des[ind]+1
}
```

- Determine randomly a neighbour (exchanging points of observation)

```
irem         <- sample(indices[des>0], size=1)
iadd         <- sample(indices, size=1)
desnew       <- des
desnew[irem] <- desnew[irem]-1
desnew[iadd] <- desnew[iadd]+1
```

# Constrained optimisation to determine design

- A design can be represented (coded) in different ways, e.g.,
  - by a vector in $\mathbb{S} = \{(n_1, \dots, n_{41}), n_i \in \mathbb{N}_0\}$ with $n_1$ being number of observations made at $w_i$
  - by a (sorted) vector $(w_1, \dots, w_{41})$ in $\{-1, -0.95, -0.9, \dots, 1\}^{41}$
- We can translate a design **des** coded in the first way to a vector **xv** of design points (second way) as follows:
  - ```
    w  <- seq(-1, 1, by=0.05)
    xv <- rep(w, des)
    ```

Python: `xv = np.repeat(w, des)`
Julia: `xv = repeat(w, inner = des)`
Matlab: `xv = repelem(w, des);`

- The design matrix **X** is then:
  - `X <- cbind(rep(1, sum(des)), xv, xv^2, xv^3)`

Python: `X = np.column_stack((np.ones_like(xv), xv, xv**2, xv**3))`
Julia: `X = hcat(ones(length(xv)), xv, xv.^2, xv.^3)`
Matlab: `X = [ones(length(xv), 1), xv', (xv').^2, (xv').^3];`
    ```