Constrained Nonlinear Optimisation

Georgy Gimel'farb

COMPSCI 369 Computational Science

RECOMMENDED READING:

- G. Strang, *Computational Science and Engineering*. Wellesley-Cambridge Press, 2007: Section 2.6
- W. H. Press et al., *Numerical Recipes: The Art of Scientific Computing*. Cambridge Univ. Press, 2007: Section 15.5
- L. R. Foulds: *Optimization Techniques: An Introduction*. Springer-Verlag, 1981: Chapters 7, 8
- C. Woodford, C. Phillips: *Numerical Methods with Worked Examples*. Chapman & Hall,1997: Chapter 7.
- M. T. Heath: Scientific Computing: An Introductory Survey. McGraw-Hill, 2002: Chapter 6

Learning Outcomes on Nonlinear Optimisation

Be familiar with unconstrained and constrained optimisation:

- Recognise discrete and continuous optimisation problems.
- Understand the method of Lagrange for optimising a function of many variables subject to a system of equality constraints.
- Be able to implement a simple search for the maximum (or minimum) of a function of one variable.
- Understand unconstrained gradient optimisation.
- Understand dynamic programming (DP) and be familiar with DP solution of the edit-distance problem.
- Understand that heuristic optimisation strategies must be used when no good exact algorithm is known.

Understand how a wide range of scientific questions can be answered by combining optimisation and other numerical methods.

Constrained Optimisation: The Method of Lagrange

- Developed by French mathematician Lagrange in 1761!
- Equality constraints $\varphi_j(\mathbf{x}) = 0;\ j = 1, \ldots, m$ to define $\mathbf{X}$
- **Example**: Quadratic function with a linear constraint:

Maximise $f(x_1, x_2) = 25 - a_1 x_1^2 - a_2 x_2^2$ subject to $x_1 - x_2 = 1$

Lagrangian $F(x_1, x_2, \lambda) = 25 - a_1 x_1^2 - a_2 x_2^2 - \lambda \cdot (x_1 - x_2 - 1)$

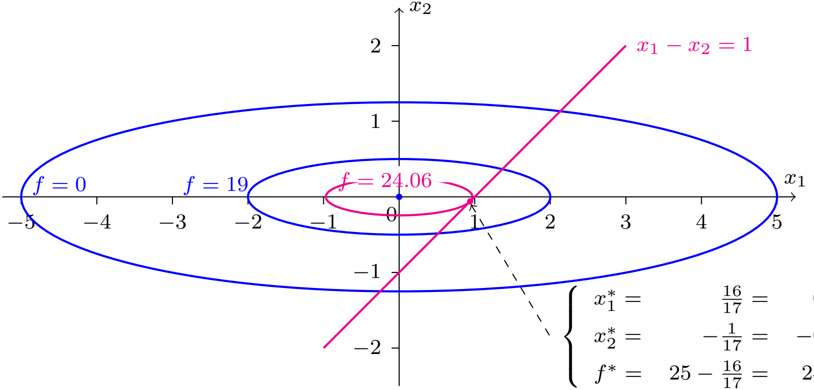
$$\rightarrow \max_{x_1, x_2} F(x_1, x_2, \lambda) \Rightarrow \nabla F(x_1, x_2, \lambda) = \mathbf{0}$$

$$\left\{\begin{array}{l} \frac{\partial F}{\partial x_1} = -2a_1 x_1 - \lambda = 0 \\ \frac{\partial F}{\partial x_2} = -2a_2 x_2 + \lambda = 0 \\ -x_1 + x_2 + 1 = 0 \end{array}\right. \Rightarrow \left\{\begin{array}{l} x_1 = -\frac{\lambda}{2a_1} \\ x_2 = \frac{\lambda}{2a_2} \\ -\frac{\lambda}{2a_1} - \frac{\lambda}{2a_2} = -1 \end{array}\right. \Rightarrow \begin{array}{l} \lambda^* = -\frac{2a_1 a_2}{a_1 + a_2} \\ x_1^* = \frac{a_2}{a_1 + a_2} \\ x_2^* = -\frac{a_1}{a_1 + a_2} \end{array}$$

Stationary point $f(x_1^*, x_2^*) = F(x_1^*, x_2^*, \lambda^*) = 25 - \frac{a_1 a_2}{a_1 + a_2}$

Constrained Optimisation: The Method of Lagrange

Example (cont): $a_1 = 1$; $a_2 = 16 \Rightarrow f(x_1, x_2) = 25 - x_1^2 - 16x_2^2$
$\Rightarrow \max\limits_{x_1,x_2} f(x_1, x_2) = 25 - x_1^2 - 16x_2^2$ subject to $x_1 - x_2 = 1$



$$\begin{cases} x_1^* = & \frac{16}{17} = & 0.94 \\ x_2^* = & -\frac{1}{17} = & -0.06 \\ f^* = & 25 - \frac{16}{17} = & 24.06 \end{cases}$$

Constrained Optimisation: The Method of Lagrange

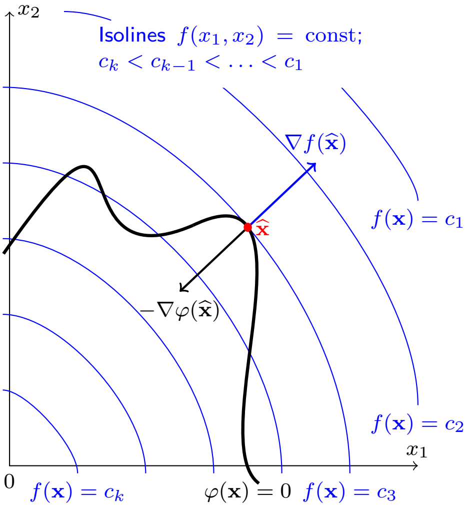
- Maximise $f(\mathbf{x})$ subject to a system of equality constraints

$$\boldsymbol{\varphi}(\mathbf{x}) = \mathbf{0} \Leftrightarrow \begin{bmatrix} \varphi_1(\mathbf{x}) \\ \vdots \\ \varphi_m(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

- Lagrangian $F(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \boldsymbol{\lambda}^\mathsf{T}\boldsymbol{\varphi}(\mathbf{x})$ where $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_m]^\mathsf{T}$ is a vector of **Lagrange multipliers**
- Its stationary points correspond to solutions of the system of nonlinear equations $\nabla F(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}$, i.e. $\frac{\partial F}{\partial \mathbf{x}} = \mathbf{0}$ and $\frac{\partial F}{\partial \boldsymbol{\lambda}} = \mathbf{0}$:

$$\begin{cases} \frac{\partial F}{\partial x_i} = 0; & i = 1, \ldots, n \\ \frac{\partial F}{\partial \lambda_j} = -\varphi_j(\mathbf{x}) = 0; & j = 1, \ldots, m \end{cases}$$

Geometric Interpretation



- Local maxima of $f(\mathbf{x})$ subject to $\varphi(\mathbf{x}) = 0$ are only at points $\widehat{\mathbf{x}}$ where the isolines $f(\mathbf{x}) = \mathrm{const}$ are tangent to the curve $\varphi(\mathbf{x}) = 0$
  - Otherwise the value of $f(\mathbf{x})$ could have been inreased by moving along the curve $\varphi(\mathbf{x}) = 0$
- Thus, at the points of local maxima the gradients of the functions f and φ lie along the same straight line: $\nabla f(\widehat{\mathbf{x}}) = \nabla \varphi(\widehat{\mathbf{x}})$
  - Lagrangian $f(\mathbf{x}) - \lambda \varphi(\mathbf{x})$ yields just the same condition in this example

Interpreting the Lagrangian: Duality

Primal problem: Maximise $f(\mathbf{x})$ subject to the system of constraints $\boldsymbol{\varphi}(\mathbf{x}) = \mathbf{0} \Leftrightarrow \max_{\mathbf{x}}\{f(\mathbf{x})|\boldsymbol{\varphi}(\mathbf{x}) = 0\}$

- Let the system $\nabla F(\mathbf{x}, \boldsymbol{\lambda}) = \nabla f(\mathbf{x}) - \boldsymbol{\lambda}^\mathsf{T} \nabla \boldsymbol{\varphi}(\mathbf{x}) = \mathbf{0}$ have a unique solution for any $\boldsymbol{\lambda}$ in the vicinity of $\boldsymbol{\lambda}^*$
- Then the maximiser in $\mathbf{x}$ can be considered as a function of $\boldsymbol{\lambda}$:

$$\max_{\mathbf{x}} F(\mathbf{x}, \boldsymbol{\lambda}) = F(\widehat{\mathbf{x}}, \boldsymbol{\lambda}) = h(\boldsymbol{\lambda}) \;\Rightarrow\; h(\boldsymbol{\lambda}^*) = F(\mathbf{x}^*, \boldsymbol{\lambda}^*) = f(\mathbf{x}^*)$$

  - It can be proven that the Lagrangian $F(\mathbf{x}, \boldsymbol{\lambda})$ has a relative minimum in $\boldsymbol{\lambda}^*$ if $\mathbf{x}$ is defined for each $\boldsymbol{\lambda}$ by the constraints $\nabla f(\mathbf{x}) - \boldsymbol{\lambda}^\mathsf{T} \nabla \boldsymbol{\varphi}(\mathbf{x}) = \mathbf{0}$

Dual problem: Minimise $F(\mathbf{x}, \boldsymbol{\lambda})$ in $\boldsymbol{\lambda}$ subject to the system of constraints $\nabla f(\mathbf{x}) - \boldsymbol{\lambda}^\mathsf{T} \nabla \boldsymbol{\varphi}(\mathbf{x}) = \mathbf{0}$

- Thus, $f(\mathbf{x}^*) = \min\limits_{\boldsymbol{\lambda}} \max\limits_{\mathbf{x}} F(\mathbf{x}, \boldsymbol{\lambda})$, and the maximiser $\mathbf{x}^*$ solves a minimax problem

Duality of the Lagrangian: An Example

$$\max_{x_1,x_2} f(x_1,x_2) = 25 - a_1x_1^2 - a_2x_2^2 \text{ subject to } x_1 - x_2 = 1$$

- Lagrangian $F(x_1,x_2,\lambda) = 25 - a_1x_1^2 - a_2x_2^2 - \lambda(x_1 - x_2 - 1)$
- $\nabla F(x_1,x_2,\lambda) = 0 \Rightarrow \widehat{\mathbf{x}}$ as a function of λ : $\begin{cases} \widehat{x}_1 & = & -\frac{\lambda}{2a_1} \\ \widehat{x}_2 & = & \frac{\lambda}{2a_2} \end{cases}$
- $$\begin{aligned} h(\lambda) &= F(\widehat{x}_1,\widehat{x}_2,\lambda) \\ &= 25 - a_1\frac{\lambda^2}{4a_1^2} - a_2\frac{\lambda^2}{4a_2^2} - \lambda\left(-\frac{\lambda}{2a_1} - \frac{\lambda}{2a_2} - 1\right) \\ &= 25 + \lambda^2\frac{a_1+a_2}{4a_1a_2} + \lambda \end{aligned}$$
- $\lambda^* = \arg\min_\lambda \left\{25 + \frac{a_1+a_2}{4a_1a_2}\lambda^2 + \lambda\right\} \Rightarrow \lambda^* = -\frac{2a_1a_2}{a_1+a_2}$
- $x_1^* = \frac{a_2}{a_1+a_2}$ and $x_2^* = -\frac{a_1}{a_1+a_2}$

Using the Method of Lagrange

System $\nabla F(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}$ is well determined:

- $n + m$ equations with $n + m$ unknowns $\mathbf{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$
- As long as these equations are independent, any constrained stationary point $\mathbf{x}^*$ of $f(\mathbf{x})$ produces a unique set of values $\boldsymbol{\lambda}^*$
- At such a point,

$$\nabla_{\mathbf{x}} F(\mathbf{x}^*, \boldsymbol{\lambda}^*) \equiv \nabla f(\mathbf{x}) - (\boldsymbol{\lambda}^*)^{\mathsf{T}} \nabla \boldsymbol{\varphi}(\mathbf{x}) = \mathbf{0}$$

i.e. $\nabla f(\mathbf{x}) - \sum_{j=1}^{m} \lambda_j \nabla \varphi_j(\mathbf{x}) = \mathbf{0}$, and $\boldsymbol{\varphi}(\mathbf{x}^*) = \mathbf{0}$

Individual components λ_j are called **sensitivity coefficients**

- The coefficients specify the rate of changes of the optimum value $F(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ when the constraints $\boldsymbol{\varphi}(\mathbf{x}^*)$ are perturbed

Using the Method of Lagrange

Although the system $\nabla F(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}$ is well determined, it is non-linear (in $\mathbf{x}$)

- In general case it may not be easy to solve this system although this may be possible in simple cases

Even if the system can be solved analytically, the solution could be found also using a numerical search

- The numerical search might be easier to use than attempt to find the analytical solution
- Because no second-order information is taken into account, the system $\nabla F(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}$ is satisfied at a constrained minimiser, maximiser, or saddle point

Nonlinear Systems of Equations

- Linear models such as $\mathbf{Ku} = \mathbf{f}$ or $\mathbf{Mu}'' + \mathbf{Ku} = \mathbf{0}$ are excellent approximations in many problems
- But many more applied problems result in nonlinear equations
- The basic nonlinear model of ν equations in ν unknowns:

$$\mathbf{g}(\mathbf{u}) = \mathbf{0} \;\Leftrightarrow\; \begin{cases} g_1(u_1, \ldots, u_\nu) = 0 \\ \vdots \\ g_\nu(u_1, \ldots, u_\nu) = 0 \end{cases}$$

To obtain a solution of $\mathbf{g}(\mathbf{u}) = \mathbf{0}$

- Start from an initial guess $\mathbf{u}^{[0]}$
- At each iteration t: improve $\mathbf{u}^{[t-1]}$ to $\mathbf{u}^{[t]}$ being closer to the solution $\mathbf{u}^*$ such that $\mathbf{g}(\mathbf{u}^*) = \mathbf{0}$

Newton's (or Newton–Raphson) Method

If the Jacobian $\nu \times \nu$ matrix of the first derivatives is known:

$$\mathbf{J}(\mathbf{u}) = \begin{bmatrix} \frac{\partial g_1}{\partial u_1}(\mathbf{u}) & \dots & \frac{\partial g_1}{\partial u_\nu}(\mathbf{u}) \\ \vdots & \ddots & \vdots \\ \frac{\partial g_\nu}{\partial u_1}(\mathbf{u}) & \dots & \frac{\partial g_\nu}{\partial u_\nu}(\mathbf{u}) \end{bmatrix}$$

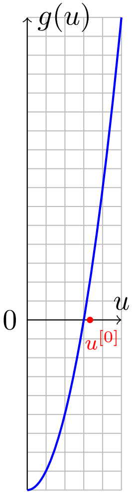
then linear approximation to $\mathbf{g}(\mathbf{u})$ around $\mathbf{u}^{[0]} = (u_1^{[0]}, \dots, u_\nu^{[0]})$ is:

$$\begin{array}{ll} \mathbf{g}(\mathbf{u}) \approx \mathbf{g}_{\mathsf{ap}:0}(\mathbf{u}) = \mathbf{g}(\mathbf{u}^{[0]}) + \mathbf{J}(\mathbf{u}^{[0]})\left(\mathbf{u} - \mathbf{u}^{[0]}\right) & \\ \mathbf{u}^{[1]} \quad \Longleftarrow \quad \mathbf{g}_{\mathsf{ap}:0}(\mathbf{u}^{[1]}) = \mathbf{g}(\mathbf{u}^{[0]}) + \mathbf{J}(\mathbf{u}^{[0]})\left(\mathbf{u}^{[1]} - \mathbf{u}^{[0]}\right) = \mathbf{0} & \end{array}$$

Each iteration $t = 1, 2, \dots$ chooses $\mathbf{u}^{[t+1]}$ such that $\mathbf{g}_{\mathsf{ap}:k}\left(\mathbf{u}^{[t+1]}\right) = \mathbf{0}$:

$$\mathbf{J}(\mathbf{u}^{[t]})\left(\mathbf{u}^{[t+1]} - \mathbf{u}^{[t]}\right) = -\mathbf{g}\left(\mathbf{u}^{[t]}\right)$$

Newton's and Modified Newton Method



Example ($g(u) = u^2 - 9$ with Jacobian $J(u) = g' = 2u$)

The exact root $u^* = 3$; $u^{[0]} = 3\frac{1}{3}$ so that $J^{[0]} = 6\frac{2}{3}$

Newton's method (NM) updates to $J^{[t]} = 2u^{[t]}$ at each step

$$2u^{[t]}\left(u^{[t+1]} - u^{[t]}\right) = 9 - \left(u^{[t]}\right)^2$$

while the **modified Newton method** (MNM) stays with the original $J^{[0]} = 2u^{[0]}$:

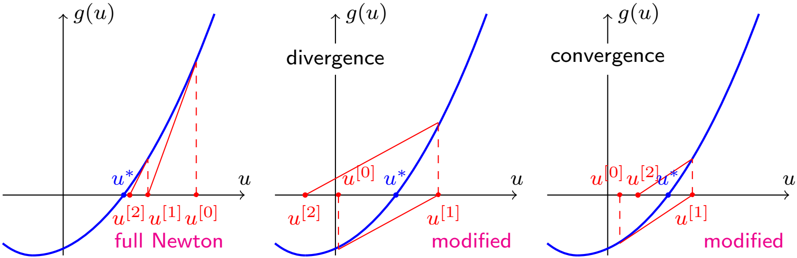
$$2u^{[0]}\left(u^{[t+1]} - u^{[t]}\right) = 9 - \left(u^{[t]}\right)^2$$

t	**NM**: Error $\lvert u^{[t]} - u^*\rvert$	**MNM**: Error $\lvert u^{[t]} - u^*\rvert$
0	0.**3**3333333333333	0.**3**3333333333333
1	0.0**1**666666666667	0.0**1**666666666667
2	0.0000**4**604051565	0.00**1**62500000000
3	0.000000000**3**5328	0.000**1**6210390625
4	0.00000000000000	0.0000**1**620644897

Newton's and Modified Newton Method

Major difficulties:

- It may be impractical to compute exactly each $\mathbf{J}(\mathbf{u}^{[t]})$ and each $\boldsymbol{\Delta}\mathbf{u}^{[t]} = \mathbf{u}^{[t+1]} - \mathbf{u}^{[t]}$
- The full Newton step $\boldsymbol{\Delta}\mathbf{u}^{[t]}$ may be dangerously large
- The modified method is simpler (as the Jacobian is computed only once) but converges slower and diverges more frequently



Fixed Point Iterations

Newton's and modified Newton methods can be represented as a **fixed point iteration**

$$\mathbf{u}^{[t+1]} = H(\mathbf{u}^{[t]})$$

- Iteration for Newton's method is $H(\mathbf{u}) = \mathbf{u} - \mathbf{J}^{-1}(\mathbf{u})\mathbf{g}(\mathbf{u})$
- Iteration for the modified NM is $H(\mathbf{u}) = \mathbf{u} - \mathbf{J}^{-1}(\mathbf{u}^{[0]})\mathbf{g}(\mathbf{u})$

Fixed point when the values $\mathbf{u}^{[t]}$ approach a limit $\mathbf{u}^*$:

$$\mathbf{u}^* = H(\mathbf{u}^*)$$

- It can be proven that the modified NM has linear convergence
  - The new error $\mathbf{e}^{[t+1]} = \mathbf{u}^* - \mathbf{u}^{[t+1]}$ multiplies the old error $\mathbf{e}^{[t]}$ by a constant factor c: $\mathbf{e}^{[t+1]} \approx c\mathbf{e}^{[t]}$
- The full Newton's method has thus the quadratic convergence rate because this factor is zero

Quadratic Model for Deriving the Newton's Method

Unconditional minimisation of a function $f(\mathbf{x})$ of n variables

- **Quadratic model**: from a truncated Taylor series expansion of $f(\mathbf{x})$ about $\mathbf{x}^{[t]}$:

  $$f(\mathbf{x}^{[t]} + \boldsymbol{\delta}) \approx q_t(\boldsymbol{\delta}) = f(\mathbf{x}^{[t]}) + \mathbf{g}_t^{\mathsf{T}}\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^{\mathsf{T}}\mathbf{H}_t\boldsymbol{\delta}$$

  where $\boldsymbol{\delta} = \mathbf{x} - \mathbf{x}^{[t]}$,
  $\mathbf{g}_t = \nabla f(\mathbf{x}^{[t]})$ is the gradient of f at point $\mathbf{x}^{[t]}$, and
  $\mathbf{H}_t$ is the Hessian of f (i.e. the Jacobian of $\mathbf{g}$) in that point
- **Newton's method**: Iterate $\mathbf{x}^{[t+1]} = \mathbf{x}^{[t]} + \boldsymbol{\delta}_t$ where $\boldsymbol{\delta}_t$ is the minimiser of $q_t(\boldsymbol{\delta})$: $\boldsymbol{\delta}_t = \arg\min\limits_{\boldsymbol{\delta}} q_t(\boldsymbol{\delta})$
  - The method is only well defined if $\mathbf{H}_t$ is positive definite since $q_t(\boldsymbol{\delta})$ has a unique minimiser only in these circumstances
  - In this case $\boldsymbol{\delta}_t$ is defined by the condition $\nabla q_t(\boldsymbol{\delta}_t) = \mathbf{0}$, i.e. $\mathbf{g}_t + \mathbf{H}_t\boldsymbol{\delta}_t = \mathbf{0}$, or $\mathbf{H}_t\boldsymbol{\delta}_t = -\mathbf{g}_t$

Quadratic Model for Deriving the Newton's Method

- Therefore, the iteration t of Newton's method can be written
  (a) Solve $\mathbf{H}_t\boldsymbol{\delta} = -\mathbf{g}_t$ for $\boldsymbol{\delta} = \boldsymbol{\delta}_t$
  (b) Set $\mathbf{x}^{[t+1]} = \mathbf{x}^{[t]} + \boldsymbol{\delta}_t$
  - Step (a): Solving the $n \times n$ system of linear equations
  - Most convenient solution: by factoring $\mathbf{H}_t = \mathbf{LDL}^\mathsf{T}$
  - This allows also for checking the positive definite condition
  - Computational complexity: $O(n^3)$ multiplications per iteration
- Quasi-Newton methods: only first derivatives required
  - The inverse matrix $\mathbf{H}_t^{-1}$ is *approximated* by a symmetric positive definite matrix $\mathbf{G}_t$
  - It is always positive definite whereas $\mathbf{H}_t$ may be indefinite
  - The matrix $\mathbf{G}_{t+1}$ is calculated from $\mathbf{G}_t$ using the second derivative information gained on the iteration t
  - Computational complexity: $O(n^2)$ multiplications per iteration

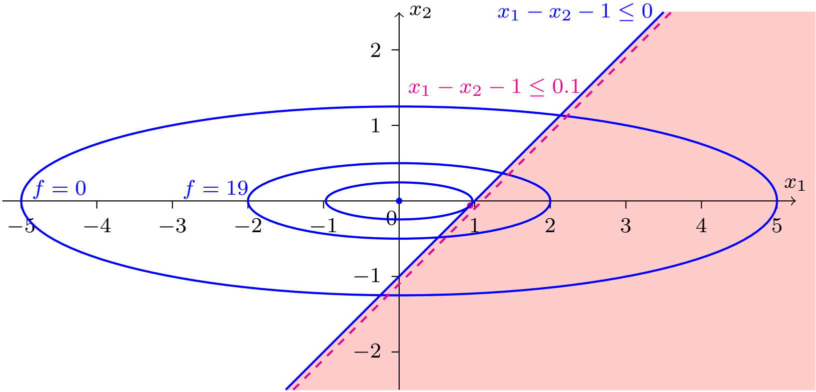
Optimisation with Inequality Constraints

Maximise $f(\mathbf{x})$ subject to a system of inequality constraints $\varphi_j(\mathbf{x}) \leq 0;\ j = 1, \ldots, m$

- Adding nonnegative slack variables s_j: $\varphi_j(\mathbf{x}) + s_j = 0$; $s_j \geq 0;\ j = 1, \ldots, m$, to transform to equality constraints
- Forming the Lagrangian $F(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \sum\limits_{j=1}^{m} \lambda_j \left(\varphi_j(\mathbf{x}) + s_j\right)$
    - λ_j gives the rate $\frac{\partial f}{\partial \varphi_j}$ of change of f w.r.t. $\varphi_j;\ j = 1, \ldots, m$
- If $\varphi_j(\mathbf{x}) \leq 0$ is changed to $\varphi_j(\mathbf{x}) \leq \varepsilon$ (here, $\varepsilon \geq 0$ is a small nonnegative number) then:
    - The set of feasible solutions for the original problem is no smaller (as the constraints are relaxed)
    - Thus the optimal solution $f(\mathbf{x}^*)$ is no lesser than for the original problem, hence $\lambda_j \geq 0$

Changes of an Inequality Constraint: $\varphi_j(\mathbf{x}) \leq \varepsilon$

$$\max_{x_1, x_2} \left\{ f(x_1, x_2) = 25 - x_1^2 - 16x_2^2 \right\} \text{ subject to } x_1 - x_2 - 1 \leq 0$$



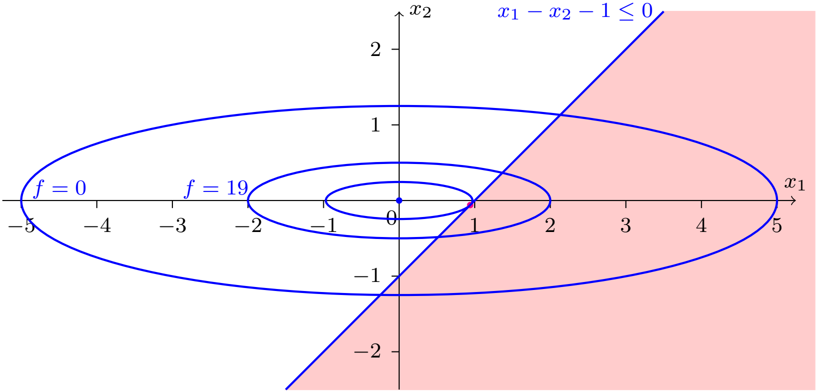
$x_1 - x_2 - 1 \leq 0.1$ relaxes the constraint, so that the set of feasible solutions is no smaller

Optimisation with Inequality Constraints

- **Inactive constraint** φ_j (i.e. $s_j > 0$):
  - Constrained stationary point lies in the region $\varphi_j(\mathbf{x}) < 0$
  - Thus, the constraint plays no role, so in the stationary condition $\lambda_j = 0$
- **Active constraint** φ_j (i.e. $s_j = 0$):
  - Constrained stationary point lies in the boundary $\varphi_j(\mathbf{x}) = 0$
  - This case is analogous to the equality constraint – with $\lambda \neq 0$
  - The sign of the multiplier is crucial because f will only be at a maximum if the rate of change $\lambda_j \geq 0$
- In both the cases: $\lambda_j \varphi_j(\mathbf{x}) = 0$; $j = 1, \ldots, m$
  - These intuitive and nonrigorous considerations are formalised in the Kuhn-Tucker theory being instrumental for optimisation with inequality constraints

Inactive Constraint

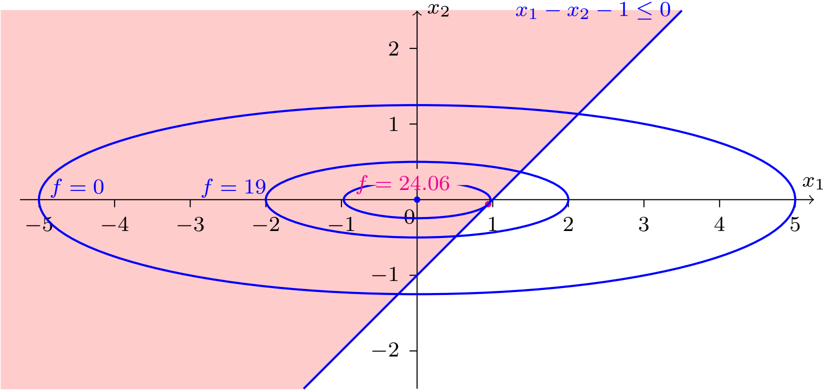
$$\max_{x_1,x_2} \left\{ f(x_1,x_2) = 25 - x_1^2 - 16x_2^2 \right\} \text{ subject to } x_1 - x_2 - 1 \le 0$$



Solution: $x_1^* = x_2^* = 0$; $f^* = f(x_1^*, x_2^*) = 25$; $\lambda^* = 0$

Active Constraint

$$\max_{x_1,x_2} \left\{ f(x_1,x_2) = 25 - x_1^2 - 16x_2^2 \right\} \text{ subject to } -x_1 + x_2 + 1 \le 0$$



Solution: $x_1^* = \frac{16}{17} \approx 0.94$; $x_2^* = -\frac{1}{17} \approx 0.06$;
$f^* = f(x_1^*, x_2^*) = 25 - \frac{16}{17} \approx 24.06$; $\lambda^* = \frac{32}{17} \approx 1.88$

Karush-Kuhn-Tucker (KKT) Theorem: Maximisation

Theorem (KKT – a nonrigorous outline)

A local maximum $\mathbf{x}^*$ *of* $f(\mathbf{x})$ *subject to constraints* $\varphi_j(\mathbf{x}) \leq 0$; $j = 1, \ldots, m$, *where* f *and all* φ_j *have continuous first derivatives and the constraints form a well-behaved boundary, meets the following necessary conditions:* $\nabla f(\mathbf{x}^*) - \sum_{j=1}^{m} \lambda_j^* \nabla \varphi_j(\mathbf{x}^*) = \mathbf{0}$ *and*

$$\begin{array}{rcl} \varphi_j(\mathbf{x}^*) & \leq & 0; \quad j = 1, \ldots, m \\ \lambda_j^* \varphi_j(\mathbf{x}^*) & = & 0; \quad j = 1, \ldots, m \\ \lambda_j^* & \geq & 0; \quad j = 1, \ldots, m \end{array}$$

for some set of real numbers $(\lambda_1^*, \ldots, \lambda_m^*)$

The boundary of the region $\mathbf{X}$ formed by the constraints $\boldsymbol{\varphi}(\mathbf{x})$ is well-behaved if from any point $\mathbf{x} \in \mathbf{X}$ one can travel along a continuous differentiable curve for a positive distance and remain in this region

KKT Theorem: Minimisation

Theorem (KKT – a nonrigorous outline)

A local minimum $\mathbf{x}^*$ *of* $f(\mathbf{x})$ *subject to constraints* $\varphi_j(\mathbf{x}) \leq 0$*;* $j = 1, \ldots, m$, *where* f *and all* φ_j *have continuous first derivatives and the constraints form a well-behaved boundary, meets the following necessary conditions:* $\nabla f(\mathbf{x}^*) - \sum_{j=1}^{m} \lambda_j^* \nabla \varphi_j(\mathbf{x}^*) = \mathbf{0}$ *and*

$$\begin{array}{rcl} \varphi_j(\mathbf{x}^*) & \leq & 0;\;\; j = 1, \ldots, m \\ \lambda_j^* \varphi_j(\mathbf{x}^*) & = & 0;\;\; j = 1, \ldots, m \\ \lambda_j^* & \leq & 0;\;\; j = 1, \ldots, m \end{array}$$

for some set of real numbers $(\lambda_1^*, \ldots, \lambda_m^*)$

- The KKT conditions are also sufficient if the feasible region $\mathbf{X}$ defined by the constraints $\boldsymbol{\varphi}(\mathbf{x}) \leq \mathbf{0}$ is convex, i.e. when functions $\varphi_j(\mathbf{x})$ are convex for all $j = 1, \ldots, m$

Quadratic Programming: Primal Problem

Maximisation of a quadratic function subject to linear inequality constraints

Primal problem:

$$\max_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{A}\mathbf{x} + \mathbf{b}^{\mathsf{T}}\mathbf{x} \quad \text{subject to} \quad \varphi(\mathbf{x}) = \mathbf{C}\mathbf{x} + \mathbf{d} \leq 0$$

- $\mathbf{A}$ – a symmetric negative definite $n \times n$ matrix (f is strictly concave)
- $\mathbf{x}$ and $\mathbf{b}$ – column vectors of n components each
- $\mathbf{C}$ – an arbitrary $m \times n$ matrix
- $\mathbf{d}$ – a column vector of m components

Quadratic Programming: Equivalent Minimax Problem

Equivalent minimax solution (by the Kuhn-Tucker theory):

Vector $\mathbf{x}^*$ of n components is a maximiser of

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} + \mathbf{b}^\mathsf{T}\mathbf{x} \quad \text{subject to} \quad \varphi(\mathbf{x}) = \mathbf{C}\mathbf{x} + \mathbf{d} \leq 0$$

if and only if $\mathbf{x}^*$ and some vector $\boldsymbol{\lambda}^*$ of m components solve the minimax problem

$$\min_{\boldsymbol{\lambda}} \max_{\mathbf{x}} \left\{ F(\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} + \mathbf{b}^\mathsf{T}\mathbf{x} - \boldsymbol{\lambda}^\mathsf{T}\left(\mathbf{C}\mathbf{x} + \mathbf{d}\right) \right\}$$

subject to $\boldsymbol{\lambda}^* \geq \mathbf{0}$

Minimax solution

For any $\boldsymbol{\lambda}$, the maximiser $\widehat{\mathbf{x}}$ equates the partial derivatives to zero:

$$\frac{\partial F(\mathbf{x},\boldsymbol{\lambda})}{\partial \mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b} - \mathbf{C}^\mathsf{T}\boldsymbol{\lambda} = \mathbf{0} \Rightarrow \widehat{\mathbf{x}} = \mathbf{A}^{-1}\left(\mathbf{C}^\mathsf{T}\boldsymbol{\lambda} - \mathbf{b}\right)$$

$$F(\widehat{\mathbf{x}},\boldsymbol{\lambda}) = \tfrac{1}{2}\boldsymbol{\lambda}^\mathsf{T}\underbrace{\left(-\mathbf{C}\mathbf{A}^{-1}\mathbf{C}^\mathsf{T}\right)}_{\mathbf{G}}\boldsymbol{\lambda} + \underbrace{\left(\mathbf{b}^\mathsf{T}\mathbf{A}^{-1}\mathbf{C}^\mathsf{T} - \mathbf{d}^\mathsf{T}\right)}_{\mathbf{g}^\mathsf{T}}\boldsymbol{\lambda} - \tfrac{1}{2}\mathbf{b}^\mathsf{T}\mathbf{A}^{-1}\mathbf{b}$$

Thus the primal maximisation problem and its minimax equivalent are solved using the much simpler **dual problem**:

$$\min_{\boldsymbol{\lambda}}\left\{\psi(\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\lambda}^\mathsf{T}\mathsf{G}\boldsymbol{\lambda} + \mathbf{g}^\mathsf{T}\boldsymbol{\lambda}\right\} \text{ subject to } \boldsymbol{\lambda} \geq \mathbf{0}$$

- The minimiser $\boldsymbol{\lambda}^* \Rightarrow$ the maximiser $\mathbf{x}^* = \mathbf{A}^{-1}\left(\mathbf{C}^\mathsf{T}\boldsymbol{\lambda}^* - \mathbf{b}\right)$
    - For a unique solution of the primal problem, $\mathbf{A}$ must be a symmetric negative definite matrix of rank n so that the matrix $\mathbf{G}$ is symmetric positive definite

Hildreth–D'Ezopo QP Algorithm

One-at-a-time search (by alternating variables)

Step $t = 1, 2, \ldots$:

The new approximation $\boldsymbol{\lambda}^{[t+1]}$ is found from the current $\boldsymbol{\lambda}^{[t]}$ by changing a single component $i = 1 + ((t-1) \mod m)$:

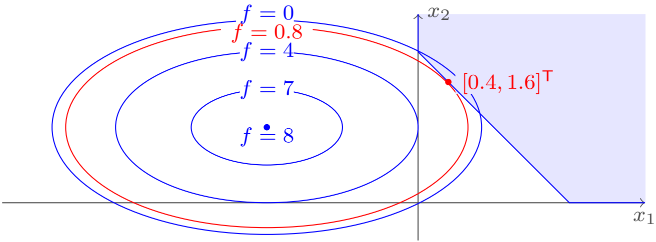
$$\lambda_i^{[t+1]} = \max\left\{0, u_i^{[t+1]}\right\} \text{ where}$$

$$u_i^{[t+1]} = -\frac{1}{G_{ii}}\left(\sum_{j=1}^{i-1} G_{ij}\lambda_j^{[t+1]} + \sum_{j=i+1}^{m} G_{ij}\lambda_j^{[t]} + g_i\right)$$

- The iterations provably converge to the minimum of $\psi(\boldsymbol{\lambda})$
- $u_i^{[t+1]}$ is the value of λ_i that sets $\frac{\partial\psi}{\partial\lambda_i} = 0$ when the preceding and subsequent components have values in $\boldsymbol{\lambda}^{[t+1]}$ and $\boldsymbol{\lambda}^{[t]}$, respectively
- Termination: when changes between $\boldsymbol{\lambda}^{[t+1]}$ and $\boldsymbol{\lambda}^{[t]}$ are very small

Quadratic Programming: An Example

Maximise $f(x_1, x_2) = 8 - (x_1 + 2)^2 - 4(x_2 - 1)^2$ subject to $x_1 \geq 0$, $x_2 \geq 0$ and $x_1 + x_2 \geq 2$



Minimax representation with $\mathbf{x} = [x_1, x_2]^\mathsf{T}$; $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \lambda_3]^\mathsf{T}$: $F(\mathbf{x}, \boldsymbol{\lambda}) =$

$$\frac{1}{2}\mathbf{x}^\mathsf{T} \underbrace{\begin{bmatrix} -2 & 0 \\ 0 & -8 \end{bmatrix}}_{\mathbf{A}} \mathbf{x} + \mathbf{x}^\mathsf{T} \underbrace{\begin{bmatrix} -4 \\ 8 \end{bmatrix}}_{\mathbf{b}} - \lambda^\mathsf{T} \left(\underbrace{\begin{bmatrix} -1 & -1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}}_{\mathbf{C}} \mathbf{x} + \underbrace{\begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{d}} \right)$$

- $\mathbf{A}^{-1} = \begin{bmatrix} -\frac{1}{2} & 0 \\ 0 & -\frac{1}{8} \end{bmatrix} \Rightarrow \hat{\mathbf{x}} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{8} & 0 & \frac{1}{8} \end{bmatrix} \boldsymbol{\lambda} - \begin{bmatrix} 2 \\ -1 \end{bmatrix}$

Quadratic Programming: An Example (cont.)

$$\mathbf{G} = -\begin{bmatrix} -1 & -1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} -\frac{1}{2} & 0 \\ 0 & -\frac{1}{8} \end{bmatrix} \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} \frac{5}{8} & \frac{1}{2} & \frac{1}{8} \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{8} & 0 & \frac{1}{8} \end{bmatrix}$$

$$\mathbf{g}^\mathsf{T} = [-4 \ \ 8] \begin{bmatrix} -\frac{1}{2} & 0 \\ 0 & -\frac{1}{8} \end{bmatrix} \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & -1 \end{bmatrix} - [2 \ \ 0 \ \ 0] = [-3 \ \ -2 \ \ 1]$$

- Initialisation: $\boldsymbol{\lambda}^{[0]} = \mathbf{0} \Rightarrow \mathbf{x}^{[0]} = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$; $f(\mathbf{x}^{[0]}) = 8$
- Iteration 1: $\boldsymbol{\lambda}^{[1]} = [4.8 \ \ 0 \ \ 0]^\mathsf{T}$:

$$\begin{aligned} \lambda_1^{[1]} &= \max\left\{0, -\tfrac{8}{5}\left(\tfrac{1}{2}\cdot 0 + \tfrac{1}{8}\cdot 0 - 3\right)\right\} &&= \max\{0, 4.8\} = 4.8 \\ \lambda_2^{[1]} &= \max\left\{0, -\tfrac{2}{1}\left(\tfrac{1}{2}\cdot \tfrac{24}{5} + 0\cdot 0 - 2\right)\right\} &&= \max\{0, -0.8\} = 0 \\ \lambda_3^{[1]} &= \max\left\{0, -\tfrac{8}{1}\left(\tfrac{1}{8}\cdot \tfrac{24}{5} + 0\cdot 0 + 1\right)\right\} &&= \max\{0, -12.8\} = 0 \end{aligned}$$

- Iteration 2(halt): $\boldsymbol{\lambda}^{[2]} = \boldsymbol{\lambda}^{[1]}$: $\boldsymbol{\lambda}^* = [4.8 \ \ 0 \ \ 0]^\mathsf{T} \Rightarrow \mathbf{x}^* = [0.4 \ \ 1.6]^\mathsf{T}$
  $\Rightarrow f(\mathbf{x}^*) = 0.8$
  - One active constraint (i.e. with $\lambda_i > 0$): $x_1 + x_2 \geq 2$

Common Types of Optimisation Problems (by M. J. Dinneen)

$f : \mathbb{R}^n \to \mathbb{R}$, $g_i : \mathbb{R}^n \to \mathbb{R}$, $h_j : \mathbb{R}^n \to \mathbb{R}$ – general functions of $\mathbf{x} \in \mathbb{R}^n$

Opimisation: Minimise $f(\mathbf{x})$ subject to

$g_i(\mathbf{x}) \geq 0$; $1 \leq i \leq p$ (inequality constraints) and
$h_j(\mathbf{x}) = 0$; $1 \leq j \leq q$ (equality constraints)

Hierarchy of optimization problems:

1. nonlinear programming; no constraints on functions
2. convex programming; f is convex, g_i concave and h_j linear
3. linear programming; f, g_i and h_j linear
4. integer programming; f, g_i and h_j linear and integer valued