School of Mathematics

Alternating Direction Method of Multipliers (ADMM)

Jacek Gondzio
Email: J.Gondzio@ed.ac.uk
URL: http://www.maths.ed.ac.uk/~gondzio

Alternating Direction Method of Multipliers (ADMM)

- Exploits Duality
- Has inexpensive iterations
- Suitable for problems with loosely coupled variables
- Numerous applications:
  - machine learning/statistics (large data sets),
  - image processing,
  - decentralized optimization.

Dual Decomposition

Consider equality constrained optimization problem

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & Ax = b \end{array}$$

where $x \in \mathcal{R}^n$, $f : \mathcal{R}^n \mapsto \mathcal{R}$, $A \in \mathcal{R}^{m \times n}$, $b \in \mathcal{R}^m$. Usually $m \leq n$.
In this lecture f is convex. (In general it does not have to be.)
We associate Lagrange multipliers $y \in \mathcal{R}^m$ with equality constraints $Ax = b$ and write the **Lagrangian**:

$$L(x, y) \; = \; f(x) + y^T (Ax - b),$$

dual function: $$L_D(y) \; = \; \inf_x L(x, y)$$

and dual problem: $$\max_y L_D(y).$$

Having found the solution of the dual at $\hat{y}$, we recover

$$\hat{x} \; = \; \text{argmin}_x L(x, \hat{y}).$$

Dual Ascent Method

Apply a (simple) gradient method for the dual problem, i.e. make steps in direction of $\nabla L_D(y)$:

$$y^{k+1} = y^k + \alpha^k \nabla L_D(y^k).$$

Observe that $$\nabla L_D(y^k) = A\tilde{x} - b,$$

where $\tilde{x} = \text{argmin}_x L(x, y^k)$.

Dual Ascent Method:
repeat until optimality is reached:

$$\begin{array}{ll} x^{k+1} = \text{argmin}_x L(x, y^k) & \text{minimize in } x \\ y^{k+1} = y^k + \alpha^k (Ax^{k+1} - b) & \text{update Lagrange multipliers } y \end{array}$$

Theory:
Strong assumptions are required for such a simple method to work.

Dual Decomposition and Separable Objective

Suppose the objective function is **separable**:

$$f(x) = f_1(x_1) + f_2(x_2) + \cdots + f_p(x_p), \quad x = (x_1, x_2, \ldots, x_p).$$

Rewrite the problem in the following form:

$$\begin{array}{ll} \min & f_1(x_1) + f_2(x_2) + \cdots + f_p(x_p) \\ \text{s.t.} & A_1 x_1 + A_2 x_2 + \cdots + A_p x_p = b \end{array}$$

and observe that the Lagrangian is **separable** in x:

$$L(x,y) \;=\; L_1(x_1,y) + L_2(x_2,y) + \cdots + L_p(x_p,y) - y^T b,$$

where $L_i(x_i,y) = f_i(x_i) + y^T A_i x_i,\; i = 1, 2, \ldots, p.$
Hence the minimization in x may be split into p separate tasks:

$$x_i^{k+1} = \text{argmin}_{x_i} L_i(x_i, y^k),\; i = 1, 2, \ldots, p$$

which do not depend on each other, and may be executed *in parallel.*

Dual Decomposition in Separable Case

Dual Ascent Method (Separable Case):

repeat until optimality is reached:

$$x_i^{k+1} = \text{argmin}_{x_i} L_i(x_i, y^k), \quad i = 1, 2, \ldots, p,$$
$$y^{k+1} = y^k + \alpha^k (\textstyle\sum_{i=1}^p A_i x_i^{k+1} - b)$$

'Decomposition' because we decompose a large problem into pieces.
Two widely used decomposition schemes rely on such a framework:

- Dantzig-Wolfe Decomposition (1960)
- Benders Decomposition (1961)

$\longrightarrow$ essential tools for solving combinatorial optimization problems.
They have a weakness though: they may be slow.

Method of Multiplers

An identifiable weakness of Dual Decomposition is the difficulty to satisfy the constraint $Ax = b$. This may be addressed by giving this constraint a more prominent role and adding to the Lagrangian the quadratic penalty of the constraint violation.

Define the **Augmented Lagrangian**:

$$L_\rho(x, y) \; = \; f(x) + y^T(Ax - b) + \frac{\rho}{2}\|(Ax - b)\|^2,$$

where ρ is a weight of the penalty.

This gives the **Method of Multipliers** (Hestenes, 1969, Powell, 1969):

repeat until optimality is reached:

$$x^{k+1} = \text{argmin}_x L_\rho(x, y^k)$$
$$y^{k+1} = y^k + \rho(Ax^{k+1} - b)$$

The method is similar to the dual ascent.
It minimizes $L_\rho(x, y^k)$ instead of $L(x, y^k)$
and uses a fixed dual update stepsize ρ instead of α.

Convergence of the Method of Multiplers

If the objective function in the optimization problem

$$\min \; f(x) \quad \text{s.t.} \quad Ax = b$$

is differentiable, then the optimality conditions are:

$$\begin{aligned} \nabla f(\hat{x}) + A^T \hat{y} &= 0 \qquad \textit{dual feasibility} \\ A\hat{x} &= b \qquad \textit{primal feasibility} \end{aligned}$$

Observe that since x^{k+1} minimizes $L_\rho(x, y^k)$, the dual update

$$y^{k+1} = y^k + \rho(Ax^{k+1} - b)$$

ensures that (x^{k+1}, y^{k+1}) is *dual feasible*. Indeed:

$$\begin{aligned} 0 = \nabla_x L_\rho(x^{k+1}, y^k) &= \nabla_x f(x^{k+1}) + A^T(y^k + \rho(Ax^{k+1} - b)) \\ &= \nabla_x f(x^{k+1}) + A^T y^{k+1}. \end{aligned}$$

However, the *primal feasibility* is attained only in the limit

$$Ax^{k+1} - b \to 0.$$

Method of Multiplers vs Dual Decomposition

Method of Multiplers:

- converges under more relaxed assumptions
  (f can be nondifferentiable)
- deals better with the primal feasibility $Ax - b$
  (presence of $\|Ax - b\|^2$ in the Augmented Lagrangian helps)

but

- the quadratic penalty $\|Ax - b\|^2$ in L_ρ destroys separability
  $\rightarrow$ cannot be used in *decomposition*.

Alternating Direction Method of Multipliers (ADMM)

ADMM offers a compromise:

- enjoys some of the benefits of the *method of multipliers*
- is well-suited to *decomposition*

Gabay and Mercier (1976), Glowinski and Marrocco (1975).

Alternating Direction Method of Multipliers (ADMM)

Consider a problem in the following form:

$$\begin{array}{ll} \min & f_1(x_1) + f_2(x_2) \\ \text{s.t.} & A_1 x_1 + A_2 x_2 = b, \end{array}$$

where $f_1 : \mathcal{R}^{n_1} \mapsto \mathcal{R}$, and $f_2 : \mathcal{R}^{n_2} \mapsto \mathcal{R}$ are convex functions (do not have to be differentiable), $A_i \in \mathcal{R}^{m \times n_i}$, $i = 1, 2$, $b \in \mathcal{R}^m$. Observe that the objective is *separable*, but the constraint links x_1 and x_2.

Write down the associated **Augmented Lagrangian**:

$$\begin{aligned} L_\rho(x_1, x_2, y) = f_1(x_1) + f_2(x_2) + y^T(A_1x_1 + A_2x_2 - b) \\ + \frac{\rho}{2}\|(A_1x_1 + A_2x_2 - b)\|^2. \end{aligned} \tag{1}$$

Alternating Direction Method of Multipliers (ADMM)

repeat until optimality is reached:

$$
\begin{aligned}
x_1^{k+1} &= \text{argmin}_{x_1}\ L_\rho(x_1, x_2^k, y^k) && \text{minimize in } x_1 \\
x_2^{k+1} &= \text{argmin}_{x_2}\ L_\rho(x_1^{k+1}, x_2, y^k) && \text{minimize in } x_2 \\
y^{k+1} &= y^k + \rho(A_1 x_1^{k+1} + A_2 x_2^{k+1} - b) && \text{update multipliers } y
\end{aligned}
$$

Observe that the optimization in x_1 uses the “old” x_2^k, but the optimization in x_2 uses already the “new” (updated) x_1^{k+1}.

Convergence of the ADMM If the functions f_1 and f_2 in the objective are differentiable, then the optimality conditions are:

$$\begin{aligned} \nabla f_1(\hat{x}_1) + A_1^T \hat{y} &= 0 && \textit{1st dual feasibility} \\ \nabla f_2(\hat{x}_2) + A_2^T \hat{y} &= 0 && \textit{2nd dual feasibility} \\ A_1\hat{x}_1 + A_2\hat{x}_2 &= b && \textit{primal feasibility} \end{aligned}$$

Note that since x_2^{k+1} minimizes $L_\rho(x_1^{k+1}, x_2, y^k)$, the dual update

$$y^{k+1} = y^k + \rho(Ax^{k+1} - b)$$

guarantees that $(x_1^{k+1}, x_2^{k+1}, y^{k+1})$ satisfies the 2nd dual feasibility constraint. Indeed:

$$\begin{aligned} 0 &= \nabla_{x_2} f_2(x_2^{k+1}) + A_2^T y^k + \rho A_2^T (A_1 x_1^{k+1} + A_2 x_2^{k+1} - b) \\ &= \nabla_{x_2} f_2(x_2^{k+1}) + A_2^T (y^k + \rho(Ax^{k+1} - b)) \\ &= \nabla_{x_2} f_2(x_2^{k+1}) + A_2^T y^{k+1}. \end{aligned}$$

However, the *1st dual feasibility* and *primal feasibility* are attained only in the limit (at convergence).

Convergence of the ADMM

Consider a problem in the following form:

$$\begin{array}{ll} \min & F(x_1, x_2) = f_1(x_1) + f_2(x_2) \\ \text{s.t.} & A_1 x_1 + A_2 x_2 = b, \end{array}$$

where $f_1 : \mathcal{R}^{n_1} \mapsto \mathcal{R}$, and $f_2 : \mathcal{R}^{n_2} \mapsto \mathcal{R}$ are convex functions (do not have to be differentiable), $A_i \in \mathcal{R}^{m \times n_i}$, $i = 1, 2$, $b \in \mathcal{R}^m$.

Convergence of the ADMM

Theorem. Suppose f_1 and f_2 are closed convex functions, and γ is any constant which satisfies $\gamma > 2\|\hat{y}\|_2$. Then

$$F(x_1^t, x_2^t) - F(\hat{x}_1, \hat{x}_2) \leq \frac{\|x_2^0 - \hat{x}_2\|^2_{\rho A_2^T A_2} + (\gamma + \|y^0\|_2)^2/\rho}{2(t+1)}$$

$$\|A_1 x_1^t + A_2 x_2^t - b\|_2 \leq \frac{\|x_2^0 - \hat{x}_2\|^2_{\rho A_2^T A_2} + (\gamma + \|y^0\|_2)^2/\rho}{\gamma(t+1)},$$

where $x_1^t := \frac{1}{t+1}\sum_{k=1}^{t+1} x_1^k$, $x_2^t := \frac{1}{t+1}\sum_{k=1}^{t+1} x_2^k$,
and for $C \succeq 0$, we define $\|u\|_C^2 := u^T C\, u$.

Theoretical convergence of ADMM is *slow*:
$\mathcal{O}(1/t)$ convergence rate and $\mathcal{O}(1/\epsilon)$ iteration complexity.
(To compare: IPMs enjoy $\mathcal{O}(\log(1/\epsilon))$ iteration complexity.)

ADMM: From 2 blocks to p blocks

Consider a problem in the following form:

$$\begin{array}{ll} \min & \sum_{i=1}^{p} f_i(x_i) \\ \text{s.t.} & \sum_{i=1}^{p} A_i x_i = b, \end{array}$$

where $f_i : \mathcal{R}^{n_i} \mapsto \mathcal{R}$, $i = 1, 2, \ldots, p$, are convex functions (do not have to be differentiable), $A_i \in \mathcal{R}^{m \times n_i}$, $i = 1, 2, \ldots, p$, $b \in \mathcal{R}^m$. Observe that the objective is *separable*, but the constraint links all the variables x_i.

Write down the associated **Augmented Lagrangian**:

$$L_\rho(x_1, x_2, \ldots, x_p, y) = \sum_{i=1}^{p} f_i(x_i) + y^T (\sum_{i=1}^{p} A_i x_i - b) + \frac{\rho}{2} \|(\sum_{i=1}^{p} A_i x_i - b)\|^2.$$

ADMM: From 2 blocks to p blocks

Multiple block version of **ADMM**:
repeat until optimality is reached:

$$
\begin{array}{lll}
x_1^{k+1} & = \operatorname{argmin}_{x_1} L_\rho(x_1, x_2^k, \ldots, x_p^k, y^k) & \text{minimize in } x_1 \\
x_2^{k+1} & = \operatorname{argmin}_{x_2} L_\rho(x_1^{k+1}, x_2, \ldots, x_p^k, y^k) & \text{minimize in } x_2 \\
\vdots & \quad \vdots & \\
x_p^{k+1} & = \operatorname{argmin}_{x_p} L_\rho(x_1^{k+1}, x_2^{k+1}, \ldots, x_p, y^k) & \text{minimize in } x_p \\
y^{k+1} & = y^k + \rho(\sum_{i=1}^p A_i x_i^{k+1} - b) & \text{update } y
\end{array}
$$

Comments on Convergence

While (under suitable assumptions) the 2-block ADMM is proved to converge, the p-block version does not have to converge, see:
C. Chen, B. He, Y. Ye, X. Yuan, "The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent", Mathematical Prog A 155 (2016) pp. 57–79.
Example with *null* objective:

$$\begin{array}{ll} \min & 0 \\ \text{s.t.} & A_1 x_1 + A_2 x_2 + A_3 x_3 = 0, \end{array}$$

where

$$A_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}.$$

Observe that $A = [A_1, A_2, A_3]$ is nonsingular. Since the right-hand-side $b = 0$, the feasible set contains a single element $\hat{x}_1 = \hat{x}_2 = \hat{x}_3 = 0$. Since the objective is null, the optimal Lagrange multiplier $\hat{y} = 0$. The 3-block ADMM is *divergent* for this problem.

Applications

ADMM is particularly attractive when the independent minimizations in x_i are significantly easier than the minimization of the aggregate objective $\sum_{i=1}^{p} f_i(x_i)$.

Sometimes a non-separable problem is converted to a separable one, just to be able to apply ADMM because of its attractive features, namely, its ability to make independent optimizations in x_i.

Example: Consider a non-separable problem

$$\begin{array}{ll} \min & f_1(x) + f_2(x) \\ \text{s.t.} & Ax = b, \end{array}$$

in which both functions f_1 and f_2 depend on the same variable x.

We create a copy of variable x and rewrite the above problem as:

$$\begin{array}{ll} \min & f_1(x) + f_2(z) \\ \text{s.t.} & Ax = b \\ & x - z = 0 \end{array}$$

in a form suitable for ADMM.

Example: ADMM for ℓ_1-regularized Least Squares

Recall $\boldsymbol{\ell_1}$-regularized least squares

$$\min \quad \tau\|x\|_1 + \frac{1}{2}\|Ax - b\|_2^2,$$

where $A \in \mathcal{R}^{m \times n}$, $b \in \mathcal{R}^m$. Usually $m \geq n$ (and often $m \gg n$).
This problem may be cast in a form suitable for ADMM:

$$\begin{array}{ll} \min & \tau\|z\|_1 + \tfrac{1}{2}\|Ax - b\|_2^2 \\ \text{s.t.} & \qquad x - z = 0. \end{array}$$

Write down the associated **Augmented Lagrangian**:

$$L_\rho(x, z, y) = \tau\|z\|_1 + \frac{1}{2}\|Ax - b\|_2^2 + y^T(x - z) + \frac{\rho}{2}\|x - z\|_2^2.$$

Example: ADMM for ℓ_1-regularized Least Squares

With such Augmented Lagrangian:

$$L_\rho(x,z,y) = \tau\|z\|_1 + \frac{1}{2}\|Ax-b\|_2^2 + y^T(x-z) + \frac{\rho}{2}\|x-z\|_2^2.$$

Minimization in x exploits the differentiability of L_ρ in x:

$$\nabla_x L_\rho(x,z,y) = A^T(Ax-b) + \rho(x-z) + y = 0,$$

which gives

$$x = (A^TA + \rho I)^{-1}(A^Tb + \rho z - y).$$

Minimization in z requires:

$$\min_z \left(\tau\|z\|_1 + \frac{\rho}{2}\|z - x - y/\rho\|_2^2\right),$$

and is perfectly separable into n independent coordinates:

$$\min_{z_i} \left(\tau|z_i| + \frac{\rho}{2}(z_i - x_i - y_i/\rho)^2\right), \quad i = 1, 2, \ldots n.$$

Soft Thresholding

In ℓ_1-regularized least squares (and in many other applications) there is a need to perform a one-dimensional update of z_i:

$$z_i^+ := \text{argmin}_{z_i} \left(\tau |z_i| + \frac{\rho}{2}(z_i - u)^2 \right),$$

Although the first term is not differentiable because it involves the absolute value, we can easily compute a closed-form solution:

$$z_i^+ := S_{\tau/\rho}(u),$$

where the *soft thresholding operator* S is defined as:

$$S_{\tau/\rho}(u) = \begin{cases} u - \tau/\rho & \text{if} \quad u > \tau/\rho \\ 0 & \text{if} \quad |u| \leq \tau/\rho \\ u + \tau/\rho & \text{if} \quad u < -\tau/\rho, \end{cases}$$

or equivalently:

$$S_{\tau/\rho}(u) = (u - \tau/\rho)_+ - (-u - \tau/\rho)_+,$$

where $v_+ := max(v, 0)$.

Example: ADMM for ℓ_1-regularized Least Squares

ℓ_1-regularized least squares problem cast in a form suitable for ADMM:

$$\begin{array}{ll} \min & \tau\|z\|_1 + \frac{1}{2}\|Ax-b\|_2^2 \\ \text{s.t.} & x - z = 0. \end{array}$$

where $A \in \mathcal{R}^{m\times n}$, $b \in \mathcal{R}^m$. Usually $m \geq n$ (and often $m \gg n$).

ADMM:

repeat until optimality is reached:

$$\begin{aligned} x^{k+1} &= (A^TA + \rho I)^{-1}(A^Tb + \rho z^k - y^k) \\ z^{k+1} &= S_{\tau/\rho}(x^{k+1} + y^k/\rho) \\ y^{k+1} &= y^k + \rho(x^{k+1} - z^{k+1}), \end{aligned}$$

where $S_{\tau/\rho}(.)$ is the *soft thresholding* operator:

$$S_{\tau/\rho}(u) = (u - \tau/\rho)_+ - (-u - \tau/\rho)_+.$$

The optimization in z is split component-wise and enjoys a trivial solution.

Example: ADMM for Consensus optimization

Consider a non-separable problem

$$\min \ \sum_{i=1}^{p} f_i(x)$$

in which all functions $f_i, i = 1, 2, \dots p$ depend on the same variable x. In machine learning, f_i might be the loss function for the i-th block of training data.

We create p copies of variable x, call them x_i, add new constraints $x_i = z, \forall i$, and then rewrite the above problem as:

$$\begin{array}{ll} \min & \sum_{i=1}^{p} f_i(x_i) \\ \text{s.t.} & x_i - z = 0, \quad i = 1, 2, \dots, p \end{array}$$

in a form suitable for ADMM. In this problem:

x_i are the *local* variables, z is a *global* variable and the constraint $x_i - z = 0$ forces all (independent) sub-problems to agree on a common value z, i.e., to reach a **consensus**.

Example: ADMM for Consensus optimization (cont'd)

Write down the associated **Augmented Lagrangian**:

$$L_\rho(x_1,x_2,\ldots,x_p,z,y_1,y_2,\ldots,y_p)=\sum_{i=1}^{p}\left(f_i(x_i)+y_i^T(x_i-z)+\frac{\rho}{2}\|x_i-z\|^2\right).$$

ADMM:

repeat until optimality is reached:

$$\begin{aligned} x_i^{k+1} &= \mathrm{argmin}_{x_i}\left(f_i(x_i)+(y_i^k)^T(x_i-z^k)+\tfrac{\rho}{2}\|x_i-z^k\|^2\right),\ i=1..p \\ z^{k+1} &= \tfrac{1}{p}\textstyle\sum_{i=1}^{p}(x_i^{k+1}+y_i^k/\rho) \\ y_i^{k+1} &= y_i^k+\rho(x_i^{k+1}-z^{k+1}), \quad i=1..p \end{aligned}$$

Observe that averaging is performed in the update of z.

Example: ADMM for QP

Consider a convex quadratic programming problem

$$\begin{array}{ll} \min & \frac{1}{2} x^T H\, x + c^T x \\ \text{s.t.} & Ax = b, \end{array}$$

where $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathcal{R}^n$, $x_i \in \mathcal{R}^{n_i}$, $n_1 + n_2 = n$, $H = \begin{bmatrix} H_{11} & H_{21}^T \\ H_{21} & H_{22} \end{bmatrix} \in \mathcal{R}^{n \times n}$ is a symmetric pos. definite matrix, $A = [A_1, A_2] \in \mathcal{R}^{m \times n}$, $b \in \mathcal{R}^m$.

Write down the associated **Augmented Lagrangian**:

$$\begin{aligned} L_\rho(x_1, x_2, y) &= \frac{1}{2} x^T H\, x + c^T x + y^T (Ax - b) + \frac{\rho}{2} \|(Ax - b)\|^2 \\ &= \frac{1}{2} \left[x_1^T, x_2^T \right] \begin{bmatrix} H_{11} + \rho A_1^T A_1 & H_{21}^T + \rho A_1^T A_2 \\ H_{21} + \rho A_2^T A_1 & H_{22} + \rho A_2^T A_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \\ &\quad + (c_1 + A_1^T y + \rho A_1^T b)^T x_1 + (c_2 + A_2^T y + \rho A_2^T b)^T x_2 + \\ &\quad + \frac{\rho}{2} b^T b - b^T y. \end{aligned}$$

Example: ADMM for QP (continued)

Recall the general **ADMM**:

repeat until optimality is reached:

$$
\begin{array}{lll}
x_1^{k+1} = \operatorname{argmin}_{x_1} L_\rho(x_1, x_2^k, y^k) & & \text{minimize in } x_1 \\
x_2^{k+1} = \operatorname{argmin}_{x_2} L_\rho(x_1^{k+1}, x_2, y^k) & & \text{minimize in } x_2 \\
y^{k+1} = y^k + \rho(A_1 x_1^{k+1} + A_2 x_2^{k+1} - b) & & \text{update multipliers } y
\end{array}
$$

For convex QP the first two tasks have closed form solutions

$$\nabla_{x_1} L_\rho = (H_{11} + \rho A_1^T A_1) x_1 + (H_{21}^T + \rho A_1^T A_2) x_2 + (c_1 + A_1^T y + \rho A_1^T b) = 0$$

$$\nabla_{x_2} L_\rho = (H_{21} + \rho A_2^T A_1) x_1 + (H_{22} + \rho A_2^T A_2) x_2 + (c_2 + A_2^T y + \rho A_2^T b) = 0$$

hence **ADMM for QP** repeats the following steps:

$$x_1^{k+1} = -(H_{11} + \rho A_1^T A_1)^{-1} \Big((H_{21}^T + \rho A_1^T A_2) x_2^k + (c_1 + A_1^T y + \rho A_1^T b) \Big)$$

$$x_2^{k+1} = -(H_{22} + \rho A_2^T A_2)^{-1} \Big((H_{21} + \rho A_2^T A_1) x_1^{k+1} + (c_2 + A_2^T y + \rho A_2^T b) \Big)$$

$$y^{k+1} = y^k + \rho(A_1 x_1^{k+1} + A_2 x_2^{k+1} - b)$$

Relation between ADMM and Gauss-Seidel

Consider a large system of linear equations $Qx = r$ which involves a positive definite matrix Q that is decomposed into $p \times p$ blocks:

$$\begin{bmatrix} Q_{11} & Q_{12} & \cdots & Q_{1p} \\ Q_{21} & Q_{22} & \cdots & Q_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{p1} & Q_{p2} & \cdots & Q_{pp} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_p \end{bmatrix}$$

(blocks may have different sizes).

Gauss-Seidel Method repeats the following steps:

$$\begin{aligned} x_1^{k+1} &= Q_{11}^{-1}(r_1 - Q_{12}x_2^k - \ldots - Q_{1p}x_p^k) \\ x_2^{k+1} &= Q_{22}^{-1}(r_2 - Q_{21}x_1^{k+1} - Q_{23}x_3^k - \ldots - Q_{2p}x_p^k) \\ \vdots & \quad \vdots \\ x_p^{k+1} &= Q_{pp}^{-1}(r_p - Q_{p1}x_1^{k+1} - Q_{p2}x_2^{k+1} - \ldots - Q_{p,p-1}x_{p-1}^{k+1}). \end{aligned}$$

S. Cipolla, J. Gondzio, ADMM and inexact ALM: the QP case.

`https://www.maths.ed.ac.uk/~gondzio/reports/ADMMandIALM.html`

(Block) Gauss-Seidel Method

Consider the following **splitting** of the matrix

$$\begin{bmatrix} Q_{11} & Q_{12} & \cdots & Q_{1p} \\ Q_{21} & Q_{22} & \cdots & Q_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{p1} & Q_{p2} & \cdots & Q_{pp} \end{bmatrix} = \underbrace{\begin{bmatrix} Q_{11} & 0 & \cdots & 0 \\ Q_{21} & Q_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ Q_{p1} & Q_{p2} & \cdots & Q_{pp} \end{bmatrix}}_{L} + \underbrace{\begin{bmatrix} 0 & Q_{12} & \cdots & Q_{1p} \\ 0 & 0 & \cdots & Q_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}}_{U},$$

and rearrange the equation

$$Qx = (L+U)x = r \quad \Leftrightarrow \quad Lx = r - Ux \quad \Leftrightarrow \quad x = L^{-1}(r - Ux).$$

Gauss-Seidel Method is a *fixed point iteration*:

$$x^{k+1} = L^{-1}(r - Ux^k).$$

Gauss-Seidel iteration overwrites the approximate solution with the new value as soon as it is computed:

$$x_i^{k+1} = Q_{ii}^{-1}(r_i - \textstyle\sum_{j<i} Q_{ij} x_j^{k+1} - \sum_{j>i} Q_{ij} x_j^k),$$

ADMM for QP acts as a **Gauss-Seidel iteration**.

Final Remarks

Alternating Direction Method of Multipliers (ADMM)

- is suitable for problems with loosely coupled variables
- has inexpensive iterations
  hence is attractive for very large scale optimization
- may be slow, but
  is often sufficiently fast when appropriately tuned
- has numerous applications due to its ‘decoupling’ ability:
  - machine learning/statistics (large data sets),
  - image processing,
  - decentralized optimization.

Modern Techniques of Large Scale Optimization for Data Science

Thank you for your attention!