Optimisation Combinatoire et Convexe

First Order Methods: Part II
First Order Methods: Part Two.

- **Centering.** Solve a centering problem at each iteration and compute a subgradient at the center to localize the solution.

- **Affine maximization.** Solve an affine maximization problem over the feasible set.

- **Partial optimization.** Solve the minimization problem over a subset of the variables.
Centering: Localization Methods
Localization methods

- Function $f : \mathbb{R}^n \to \mathbb{R}$ convex (and for now, differentiable)
- **problem:** minimize $f$
- **oracle model:** for any $x$ we can evaluate $f$ and $\nabla f(x)$ (at some cost)

Main assumption: **evaluating the gradient is very expensive.**

From $f(x) \geq f(x_0) + \nabla f(x_0)^T(x - x_0)$ we conclude

$$\nabla f(x_0)^T(x - x_0) \geq 0 \implies f(x) \geq f(x_0)$$

i.e., all points in halfspace $\nabla f(x_0)^T(x - x_0) \geq 0$ are **worse** than $x_0$
Localization methods

by evaluating $\nabla f$ we rule out a halfspace in our search for $x^*$:

$$x^* \in \{x \mid \nabla f(x_0)^T (x - x_0) \leq 0\}$$

idea: get one bit of info (on location of $x^*$) by evaluating $\nabla f$

for nondifferentiable $f$, can replace $\nabla f(x_0)$ with any subgradient $g \in \partial f(x_0)$
Localization methods

Suppose we have evaluated $\nabla f(x_1), \ldots, \nabla f(x_k)$ then we know $x^* \in \{x \mid \nabla f(x_i)^T (x - x_i) \leq 0\}$.

On the basis of $\nabla f(x_1), \ldots, \nabla f(x_k)$, we have localized $x^*$ to a polyhedron.

**Question:** what is a ‘good’ point $x_{k+1}$ at which to evaluate $\nabla f$?
Localization methods

Basic localization (or cutting-plane) algorithm:

1. after iteration $k - 1$ we know $x^* \in P_{k-1}$:

   $$P_{k-1} = \{ x \mid \nabla f(x^{(i)})^T (x - x^{(i)}) \leq 0, \; i = 1, \ldots, k - 1 \}$$

2. evaluate $\nabla f(x^{(k)})$ (or $g \in \partial f(x^{(k)})$) for some $x^{(k)} \in P_{k-1}$

3. $P_k := P_{k-1} \cap \{ x \mid \nabla f(x^{(k)})^T (x - x^{(k)}) \leq 0 \}$
Localization methods

- $\mathcal{P}_k$ gives our uncertainty of $x^*$ at iteration $k$
- want to pick $x^{(k)}$ so that $\mathcal{P}_{k+1}$ is as small as possible
- clearly want $x^{(k)}$ near center of $C^{(k)}$
Example: bisection on $\mathbb{R}$

- $f : \mathbb{R} \to \mathbb{R}$
- $\mathcal{P}_k$ is interval
- obvious choice: $x^{(k+1)} := \text{midpoint}(\mathcal{P}_k)$

<table>
<thead>
<tr>
<th>bisection algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>given</strong> interval $C = [l, u]$ containing $x^*$</td>
</tr>
<tr>
<td>repeat</td>
</tr>
<tr>
<td>1. $x := (l + u)/2$</td>
</tr>
<tr>
<td>2. evaluate $f'(x)$</td>
</tr>
<tr>
<td>3. if $f'(x) &lt; 0$, $l := x$; else $u := x$</td>
</tr>
</tbody>
</table>
Example: bisection on $\mathbb{R}$

\[
\text{length}(\mathcal{P}_{k+1}) = u_{k+1} - l_{k+1} = \frac{u_k - l_k}{2} = \frac{1}{2} \text{length}(\mathcal{P}_k)
\]

and so \(\text{length}(\mathcal{P}_k) = 2^{-k} \text{length}(\mathcal{P}_0)\)
Example: bisection on $\mathbb{R}$

**interpretation:**

- $\text{length}(\mathcal{P}_k)$ measures our uncertainty in $x^*$
- Uncertainty is halved at each iteration; get exactly one bit of info about $x^*$ per iteration
- # steps required for uncertainty (in $x^*$) $\leq \epsilon$:

$$\log_2 \frac{\text{length}(\mathcal{P}_0)}{\epsilon} = \log_2 \frac{\text{initial uncertainty}}{\text{final uncertainty}}$$

**question:**

- Can bisection be extended to $\mathbb{R}^n$?
- Or is it special since $\mathbb{R}$ is linear ordering?
Center of gravity algorithm

Take $x^{(k+1)} = \text{CG}(\mathcal{P}_k)$ (center of gravity)

$$\text{CG}(\mathcal{P}_k) = \frac{\int_{\mathcal{P}_k} x \, dx}{\int_{\mathcal{P}_k} dx}$$

**Theorem.** If $C \subseteq \mathbb{R}^n$ convex, $x_{\text{cg}} = \text{CG}(C)$, $g \neq 0$,

$$\text{vol} \left( C \cap \{x \mid g^T(x - x_{\text{cg}}) \leq 0\} \right) \leq (1 - 1/e) \text{vol}(C) \approx 0.63 \text{ vol}(C)$$

(independent of dimension $n$)

hence in CG algorithm, $\text{vol}(\mathcal{P}_k) \leq 0.63^k \text{ vol}(\mathcal{P}_0)$
Center of gravity algorithm

- $\text{vol}(\mathcal{P}_k)^{1/n}$ measures uncertainty (in $x^*$) at iteration $k$
- Uncertainty reduced at least by $0.63^{1/n}$ each iteration
- From this can prove $f(x^{(k)}) \rightarrow f(x^*)$ (later)
- Max. # steps required for uncertainty $\leq \epsilon$:

$$1.51n \log_2 \frac{\text{initial uncertainty}}{\text{final uncertainty}}$$

(cf. bisection on $\mathbb{R}$)
Center of gravity algorithm

advantages of CG-method

- guaranteed convergence
- number of steps proportional to dimension $n$, log of uncertainty reduction

disadvantages

- finding $x^{(k+1)} = \text{CG}(\mathcal{P}_k)$ is harder than original problem
- $\mathcal{P}_k$ becomes more complex as $k$ increases
  (removing redundant constraints is harder than solving original problem)

(but, can modify CG-method to work)
**Analytic center cutting-plane method**

**analytic center** of polyhedron $\mathcal{P} = \{ z \mid a_i^T z \leq b_i, \ i = 1, \ldots, m \}$ is

$$AC(\mathcal{P}) = \arg\min_z - \sum_{i=1}^{m} \log(b_i - a_i^T z)$$

**ACCPM** is localization method with next query point $x^{(k+1)} = AC(\mathcal{P}_k)$ (found by Newton’s method)
let $x^*$ be analytic center of $\mathcal{P} = \{z \mid a_i^T z \leq b_i, \ i = 1, \ldots, m\}$

let $H^*$ be Hessian of barrier at $x^*$,

$$H^* = -\nabla^2 \sum_{i=1}^{m} \log(b_i - a_i^T z) \bigg|_{z=x^*} = \sum_{i=1}^{m} \frac{a_i a_i^T}{(b_i - a_i^T x^*)^2}$$

then, $\mathcal{P} \subseteq \mathcal{E} = \{z \mid (z - x^*)^T H^* (z - x^*) \leq m^2\}$ (not hard to show)
Let $E^{(k)}$ be outer ellipsoid associated with $x^{(k)}$

A lower bound on optimal value $p^*$ is

$$p^* \geq \inf_{z \in E^{(k)}} \left( f(x^{(k)}) + g^{(k)}T(z - x^{(k)}) \right)$$

$$= f(x^{(k)}) - m_k \sqrt{g^{(k)}T H^{(k)} - g^{(k)}}$$

($m_k$ is number of inequalities in $\mathcal{P}_k$)

Gives simple stopping criterion $\sqrt{g^{(k)}T H^{(k)} - g^{(k)}} \leq \epsilon / m_k$
Best objective and lower bound

since ACCPM isn’t a descent a method, we keep track of best point found, and
best lower bound

**Best function value** so far: \( u_k = \min_{i=1,\ldots,k} f(x^{(k)}) \)

**Best lower bound** so far: \( l_k = \max_{i=1,\ldots,k} f(x^{(k)}) - m_k \sqrt{g^{(k)T} H^{(k)} - 1} g^{(k)} \)

can stop when \( u_k - l_k \leq \epsilon \)
Basic ACCPM

given polyhedron $\mathcal{P}$ containing $x^*$

repeat
1. compute $x^*$, the analytic center of $\mathcal{P}$, and $H^*$
2. compute $f(x^*)$ and $g \in \partial f(x^*)$
3. $u := \min\{u, f(x^*)\}$
   
   $l := \max\{l, f(x^*) - m \sqrt{g^T H^* - 1} g\}$
4. add inequality $g^T(z - x^*) \leq 0$ to $\mathcal{P}$

until $u - l < \epsilon$

here $m$ is number of inequalities in $\mathcal{P}$
Dropping constraints

ACCPM adds an inequality to $\mathcal{P}$ each iteration, so centering gets harder, more storage as algorithm progresses.

Schemes for **dropping constraints** from $\mathcal{P}^{(k)}$:

- remove all redundant constraints (expensive)
- remove some constraints known to be redundant
- remove constraints based on some relevance ranking
Dropping constraints in ACCPM

$x^*$ is AC of $\mathcal{P} = \{x \mid a_i^T x \leq b_i, \ i = 1, \ldots, m\}$, $H^*$ is barrier Hessian at $x^*$

define (ir)relevance measure $\eta_i = \frac{b_i - a_i^T x^*}{\sqrt{a_i^T H^* a_i}}$

- $\eta_i/m$ is normalized distance from hyperplane $a_i^T x = b_i$ to outer ellipsoid
- if $\eta_i \geq m$, then constraint $a_i^T x \leq b_i$ is redundant

common ACCPM constraint dropping schemes:

- drop all constraints with $\eta_i \geq m$ (guaranteed to not change $\mathcal{P}$)
- drop constraints in order of irrelevance, keeping constant number, usually $3n - 5n$
Example

PWL objective, \( n = 10 \) variables, \( m = 100 \) terms

simple ACCPM: \( f(x^{(k)}) \) and lower bound \( f(x^{(k)}) - m \sqrt{g^{(k)T} H^{(k)} - 1 g^{(k)}} \)
ACCPM with constraint dropping

\[ u_k - p^* \]

\[ u_k - l_k \]

- no dropping
- drop \( \eta_i > m \)
- keep \( 3n \)
ACCPM with constraint dropping

number of inequalities in $\mathcal{P}$:

\[ k \]

. . . constraint dropping actually improves convergence (!)
Challenges in cutting-plane methods:

- can be difficult to compute appropriate next query point
- localization polyhedron grows in complexity as algorithm progresses

can get around these challenges . . .

**ellipsoid method** is another approach

- developed in 70s by Shor and Yudin
- used in 1979 by Khachian to give polynomial time algorithm for LP
Ellipsoid algorithm

**idea:** localize $x^*$ in an **ellipsoid** instead of a **polyhedron**

1. at iteration $k$ we know $x^* \in \mathcal{E}^{(k)}$

2. set $x^{(k+1)} := \text{center}(\mathcal{E}^{(k)})$; evaluate $\nabla f(x^{(k+1)})$ (or $g^{(k)} \in \partial f(x^{(k+1)})$)

3. hence we know

$$x^* \in \mathcal{E}^{(k)} \cap \{z \mid \nabla f(x^{(k+1)})^T(z - x^{(k+1)}) \leq 0\}$$

(a half-ellipsoid)

4. set $\mathcal{E}^{(k+1)} := \text{minimum volume ellipsoid covering} \ \mathcal{E}^{(k)} \cap \{z \mid \nabla f(x^{(k+1)})^T(z - x^{(k+1)}) \leq 0\}$
Ellipsoid algorithm

compared to cutting-plane method:

- localization set doesn’t grow more complicated
- easy to compute query point
- but, we add unnecessary points in step 4
Properties of ellipsoid method

- reduces to bisection for $n = 1$
- simple formula for $E^{(k+1)}$ given $E^{(k)}$, $\nabla f(x^{(k+1)})$
- $E^{(k+1)}$ can be larger than $E^{(k)}$ in diameter (max semi-axis length), but is always smaller in volume
- $\text{vol}(E^{(k+1)}) < e^{-\frac{1}{2n}} \text{vol}(E^{(k)})$
  (note that volume reduction factor depends on $n$)
Example
Updating the ellipsoid

\[ E(x, A) = \{ z \mid (z - x)^T A^{-1} (z - x) \leq 1 \} \]
Updating the ellipsoid

(for $n > 1$) minimum volume ellipsoid containing

$$\mathcal{E} \cap \{ z \mid g^T(z - x) \leq 0 \}$$

is given by

$$x^+ = x - \frac{1}{n+1} A\tilde{g}$$

$$A^+ = \frac{n^2}{n^2 - 1} \left( A - \frac{2}{n+1} A\tilde{g}\tilde{g}^T A \right)$$

where $\tilde{g} \triangleq g / \sqrt{g^T A g}$
As in the ACCPM case, we can get **error bounds** on the current iterate.

\[ x^* \in \mathcal{E}_k, \text{ so} \]

\[
f(x^*) \geq f(x^{(k)}) + \nabla f(x^{(k)})^T (x^* - x^{(k)})
\]

\[
\geq f(x^{(k)}) + \inf_{x \in \mathcal{E}(k)} \nabla f(x^{(k)})^T (x - x^{(k)})
\]

\[
= f(x^{(k)}) - \sqrt{\nabla f(x^{(k)})^T A^{(k)} \nabla f(x^{(k)})}
\]

**simple stopping criterion:**

\[
\sqrt{\nabla f(x^{(k)})^T A^{(k)} \nabla f(x^{(k)})} \leq \epsilon
\]
Stopping criterion

\[ f(x^{(k)}) - \sqrt{\nabla f(x^{(k)})^T A^{(k)} \nabla f(x^{(k)})} \]

\[ f^* \]

\[ f(x^{(k)}) \]

Ax. d'Aspremont. M1 ENS.
Basic ellipsoid algorithm

ellipsoid described as $\mathcal{E}(x, A) = \{ z \mid (z - x)^T A^{-1} (z - x) \leq 1 \}$

given ellipsoid $\mathcal{E}(x, A)$ containing $x^*$, accuracy $\epsilon > 0$

repeat

1. evaluate $\nabla f(x)$ (or $g \in \partial f(x)$)
2. if $\sqrt{\nabla f(x)^T A \nabla f(x)} \leq \epsilon$, return($x$)
3. update ellipsoid
   3a. $\tilde{g} := \nabla f(x) / \sqrt{\nabla f(x)^T A \nabla f(x)}$
   3b. $x := x - \frac{1}{n+1} A \tilde{g}$
   3c. $A := \frac{n^2}{n^2 - 1} \left( A - \frac{2}{n+1} A \tilde{g} \tilde{g}^T A \right)$

properties:

- can propagate Cholesky factor of $A$; get $O(n^2)$ update
- not a descent method
- often slow but robust in practice
Interpretation

- change coordinates so uncertainty \((\mathcal{E})\) is unit ball
- take gradient (or subgradient) step with fixed length \(1/(n + 1)\)

**properties:**

- can propagate Cholesky factor of \(A\); get \(O(n^2)\) update
- **not** a descent method
- often slow but robust in practice
Proof of convergence

assumptions:

- $f$ is Lipschitz: $|f(y) - f(x)| \leq G\|y - x\|
- $\mathcal{E}^{(0)}$ is ball with radius $R$

suppose $f(x^{(i)}) > f^* + \epsilon$, $i = 0, \ldots, k$, then

$$f(x) \leq f^* + \epsilon \implies x \in \mathcal{E}^{(k)}$$

since at iteration $i$ we only discard points with $f \geq f(x^{(i)})$, then from Lipschitz condition,

$$\|x - x^*\| \leq \epsilon/G \implies f(x) \leq f^* + \epsilon \implies x \in \mathcal{E}^{(k)}$$

so $B = \{x \mid \|x - x^*\| \leq \epsilon/G\} \subseteq \mathcal{E}^{(k)}$, hence $\text{vol}(B) \leq \text{vol}(\mathcal{E}^{(k)})$, so

$$\beta_n(\epsilon/G)^n \leq e^{-k/2n} \text{vol}(\mathcal{E}^{(0)}) = e^{-k/2n} \beta_n R^n$$

($\beta_n$ is volume of unit ball in $\mathbb{R}^n$), therefore $k \leq 2n^2 \log(RG/\epsilon)$
conclusion: for $K > 2n^2 \log(RG/\epsilon)$,

$$\min_{i=0,\ldots,K} f(x^{(i)}) \leq f^* + \epsilon$$
Interpretation of complexity

since $x^* \in \mathcal{E}_0 = \{x \mid \|x - x^{(0)}\| \leq R\}$, our prior knowledge of $f^*$ is

$$f^* \in [f(x^{(0)}) - GR, f(x^{(0)})]$$

our prior uncertainty in $f^*$ is $GR$

after $k$ iterations our knowledge of $f^*$ is

$$f^* \in \left[\min_{i=0,\ldots,k} f(x^{(i)}) - \epsilon, \min_{i=0,\ldots,k} f(x^{(i)})\right]$$

posterior uncertainty in $f^*$ is $\leq \epsilon$

iterations required:

$$2n^2 \log \frac{RG}{\epsilon} = 2n^2 \log \frac{\text{prior uncertainty}}{\text{posterior uncertainty}}$$

efficiency: $0.72/n^2$ bits per gradient evaluation (degrades with $n$)
Inequality constrained problems

minimize \( f_0(x) \)
subject to \( f_i(x) \leq 0, \ i = 1, \ldots, m \)

**same idea:** maintain ellipsoids \( \mathcal{E}^{(k)} \) that

- contain \( x^* \)
- decrease in volume to zero
**case 1:** $x^{(k)}$ feasible, i.e., $f_i(x^{(k)}) \leq 0$, $i = 1, \ldots, m$

- then do usual update of $E^{(k)}$ based on $\nabla f_0(x^{(k)})$
- rules out halfspace of points with larger function value than current point

**case 2:** $x^{(k)}$ infeasible, say, $f_j(x^{(k)}) > 0$;

- then $\nabla f_j(x^{(k)})^T(x - x^{(k)}) \geq 0 \implies f_j(x) > 0 \implies x$ infeasible so update $E^{(k)}$ based on $\nabla f_j(x^{(k)})$
- rules out halfspace of infeasible points
Affine Maximization: Frank-Wolfe
Classical first order methods for solving

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in C,
\end{align*}
\]

in \( x \in \mathbb{R}^n \), with \( C \subset \mathbb{R}^n \) convex, relied on the assumption that the following subproblem could be solved efficiently

\[
\begin{align*}
\text{minimize} & \quad y^T x + d(x) \\
\text{subject to} & \quad x \in C,
\end{align*}
\]

in the variable \( x \in \mathbb{R}^n \), where \( d(x) \) is a strongly convex function.

The method detailed here assumes instead that the affine minimization subproblem

\[
\begin{align*}
\text{minimize} & \quad d^T x \\
\text{subject to} & \quad x \in C
\end{align*}
\]

can be solved efficiently for any \( y \in \mathbb{R}^n \).
Algorithm.

- Choose $x_0 \in C$.
- For $k = 1, \ldots, k_{\text{max}}$ iterate
  1. Compute $\nabla f(x_k)$
  2. Solve
     \[
     \begin{align*}
     \text{minimize} & \quad x^T \nabla f(y_k) \\
     \text{subject to} & \quad x \in C
     \end{align*}
     \]
     in $x \in \mathbb{R}^n$, call the solution $x_d$.
  3. Update the current point
     \[
     x_{k+1} = x_k + \frac{2}{k+2}(x_d - x_k)
     \]

Note that all iterates are feasible.
**Complexity.** Assume that $f$ is differentiable. Define the curvature $C_f$ of the function $f(x)$ as

$$C_f \triangleq \sup_{s, x \in \mathcal{M}, \alpha \in [0, 1], \ y = x + \alpha (s - x)} \frac{1}{\alpha^2} (f(y) - f(x) - \langle y - x, \nabla f(x) \rangle).$$

The Franke-Wolfe algorithm will then produce an $\epsilon$ solution after

$$N_{\max} = \frac{4C_f}{\epsilon}$$

iterations.
- **Stopping criterion.** At each iteration, we get a lower bound on the optimum as a byproduct of the affine minimization step. By convexity

\[
f(x_k) + \nabla f(x_k)^T(x_d - x_k) \leq f(x), \quad \text{for all } x \in C
\]

and finally, calling \( f^* \) the optimal value of problem, we obtain

\[
f(x_k) - f^* \leq \nabla f(x_k)^T(x_k - x_d).
\]

This allows us to bound the suboptimality of iterate at no additional cost.
Partial Minimization: Coordinate Descent
Coordinate Descent

We seek to solve

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in C
\end{align*}
\]

in the variable \( x \in \mathbb{R}^n \), with \( C \subset \mathbb{R}^n \) convex.

- Our main assumption here is that \( C \) is a product of simpler sets. We rewrite the problem

\[
\begin{align*}
\text{minimize} & \quad f(x_1, \ldots, x_p) \\
\text{subject to} & \quad x_i \in C_i, \quad i = 1, \ldots, p
\end{align*}
\]

where \( C = C_1 \times \ldots \times C_p \).

- This helps if the minimization subproblems

\[
\min_{x_i \in C_i} f(x_1, \ldots, x_i, \ldots, x_p)
\]

can be solved very efficiently (or in closed-form).
Coordinate Descent

Algorithm. The algorithm simply computes the iterates $x^{(k+1)}$ as

$$
x^{(k+1)}_i = \arg\min_{x_i \in C_i} f(x_1^{(k)}, \ldots, x_i^{(k)}, \ldots, x_p^{(k)})
$$

$$
x^{(k+1)}_j = x_j^{(k)}, \quad j \neq i
$$

for a certain $i \in [1, p]$, cycling over all indices in $[1, p]$.

Convergence.

- Complexity analysis similar to coordinate-wise gradient descent (or steepest descent in $\ell_1$ norm).
- Need $f(x)$ strongly convex to get linear complexity bound.
- Few clean results outside of this setting.
Coordinate Descent

Example.

- Consider the box constrained minimization problem

\[
\begin{align*}
\text{minimize} & \quad x^T A x + b^T x \\
\text{subject to} & \quad \|x\|_\infty \leq 1
\end{align*}
\]

in the variable \(x \in \mathbb{R}^n\). We assume \(A > 0\).

- The set \(\|x\|_\infty \leq 1\) is a box, i.e. a product of intervals.

- Each minimization subproblem means solving a second order equation.

- The dual is

\[
\min_{y \in \mathbb{R}^n} \ (b + y)^T A^{-1} (b + y) - 4\|y\|_1
\]

which can be interpreted as a penalized regression problem in the variable \(y \in \mathbb{R}^n\).
Partial Minimization:

Dykstra, alternating projection
We focus on a simple **feasibility problem**

$$\text{find } x \in C_1 \cap C_2$$

in the variable $x \in \mathbb{R}^n$ with $C_1, C_2 \subset \mathbb{R}^n$ two convex sets.

We assume now that the projection problems on $C_i$ are easier to solve

$$\begin{align*}
\text{minimize} & \quad \|x - y\|_2 \\
\text{subject to} & \quad x \in C_i
\end{align*}$$

in $x \in \mathbb{R}^n$. 

A. d’Aspremont. M1 ENS.
Algorithm (alternating projection)

- Choose $x_0 \in \mathbb{R}^n$.
- **For** $k = 1, \ldots, k_{\text{max}}$ **iterate**
  
  1. Project on $C_1$

    $$x_{k+1/2} = \arg\min_{x \in C_1} \|x - x_k\|_2$$

  2. Project on $C_2$

    $$x_{k+1} = \arg\min_{x \in C_2} \|x - x_{k+1/2}\|_2$$

**Convergence.** We can show $\text{dist}(x_k, C_1 \cap C_2) \to 0$. Linear convergence provided some additional regularity assumptions.
Dykstra, alternating projection

Algorithm (Dykstra)

- Choose \( x_0, z_0 \in \mathbb{R}^n \).
- For \( k = 1, \ldots, k^{max} \) iterate
  1. Project on \( C_1 \)
     \[ x_{k+1/2} = \arg\min_{x \in C_1} \| x - z_k \|_2 \]
  2. Update
     \[ z_{k+1/2} = 2x_{k+1/2} - z_k \]
  3. Project on \( C_2 \)
     \[ x_{k+1} = \arg\min_{x \in C_2} \| x - z_{k+1/2} \|_2 \]
  4. Update
     \[ z_{k+1} = z_k + x_{k+1} - x_{k+1/2} \]

Convergence. Usually faster than simple alternating projection.