Prediction-Correction Methods for Time-Varying Convex Optimization

Geert Leus
collaborators: A. Simonetto, A. Mokhtari, A. Koppel, A. Ribeiro

EEMCS, Delft University of Technology, The Netherlands
The core

Define a smooth convex function \( f(x; t) : \mathbb{R}^m \times \mathbb{R}_+ \to \mathbb{R} \)
We want to solve the minimization problem

\[
\min_{x \in \mathbb{R}^m} f(x; t), \quad \text{for all } t \geq 0
\]

That is, we want to find the solution trajectory \( x^*(t) \)
The core

Define a smooth convex function $f(x; t) : \mathbb{R}^m \times \mathbb{R}_+ \rightarrow \mathbb{R}$

We want to solve the minimization problem

$$\min_{x \in \mathbb{R}^m} f(x; t), \quad \text{for all } t \geq 0$$

That is, we want to find the solution trajectory $x^*(t)$

Here we assume $f$ to be smooth (for each $x$ and uniformly in $t$):

- doubly differentiable, $m$-strongly convex, gradient $L$ Lipschitz
- higher derivatives (both in time and “space”) bounded
  (i.e., the mapping $t \mapsto x^*(t)$ is single-valued and locally Lipschitz)
Continuous-time optimization

Example problem settings:

- finding control gains
- target tracking
- statistical parameter estimation
The task

Sample the problem at time instances $t_k$, where $h = t_k - t_{k-1}$

$$\min_{x \in \mathbb{R}^m} f(x; t_k), \quad \text{for } k = 0, 1, 2, \ldots$$

1. Computational brute force?
The task

Sample the problem at time instances $t_k$, where $h = t_k - t_{k-1}$

$$\minimize_{x \in \mathbb{R}^m} f(x; t_k), \quad \text{for } k = 0, 1, 2, \ldots$$

1. Computational brute force?

2. Smarter choice: iterative solution-trackers: algorithms s.t.:
   - start at a certain $x_0$
   - generate a sequence $\{x_k\}$ such that
     $$\lim_{k \to \infty} \|x_k - x^*(t_k)\| \leq \delta(h)$$

We look at prediction-correction schemes
Some background

- Non-stationary opt. [Gupal, Kheisin, Nurminskii, etc., 1970s]
  Only prediction, require initial optimizer $x^*(t_0)$

- Parametric programming [Robinson, Rockafellar et.al., 1980s-2010]
  Perturbation analysis based on an initialization $x^*(t_0)$

- Distributed signal processing/control (2010-on)
  Dual decomp. [Jakubeic], ADMM [Boyd], DGD [Nedich]
  Only correction, arbitrary initial point $x_0$

- This work: track $x^*(t)$ up to $O(h^2)$ or $O(h^4)$ in some cases
Centralized methods


The dynamics inside iso-residual manifolds

At optimality we know that

\[ \nabla_{\mathbf{x}} f(\mathbf{x}^*(t); t) = 0 \]

and for any other vector \( \mathbf{x} \)

\[ \nabla_{\mathbf{x}} f(\mathbf{x}; t) = \mathbf{r}(t) \quad \text{where } \mathbf{r}(t) \text{ is the residual} \]
The dynamics inside iso-residual manifolds

At optimality we know that

\[ \nabla_x f(x^*(t); t) = 0 \]

and for any other vector \( x \)

\[ \nabla_x f(x; t) = r(t) \quad \text{where } r(t) \text{ is the residual} \]

We use this fact to generate (iso-residual) dynamics

\[ \nabla_x f(x; t) + \nabla_{xx} f(x; t) \delta x + \nabla_{tx} f(x; t) \delta t = r(t), \]

that is

\[ \dot{x} = - (\nabla_{xx} f(x; t))^{-1} \nabla_{tx} f(x; t) \]
The dynamics inside iso-residual manifolds

At optimality we know that

$$\nabla_x f(x^*(t); t) = 0$$

and for any other vector $x$

$$\nabla_x f(x; t) = r(t) \quad \text{where } r(t) \text{ is the residual}$$

Euler integration of $\dot{x} = -(\nabla_{xx} f(x; t))^{-1} \nabla_{tx} f(x; t)$ yields

$$x_{k+1|k} = x_k - (\nabla_{xx} f(x_k; t_k))^{-1} \nabla_{tx} f(x_k; t_k) h$$

With predicted variable $x_{k+1|k}$, correct using new info. $f(\cdot; t_{k+1})$
Predictor-corrector schemes

Centralized algorithms for trajectory tracking

- start at $x_0$, fix step-size $\gamma > 0$, sampling rate $h = t_{k+1} - t_k$

  1. Prediction step: predict $x_{k+1|k}$ based on objective at time $t_k$

     $x_{k+1|k} = x_k - h (\nabla_x f(x_k; t_k))^{-1} \nabla_t f(x_k; t_k)$

  2. Correction step: update prediction via objective at time $t_{k+1}$

     $x_{k+1} = x_{k+1|k} - \gamma \nabla_x f(x_{k+1|k}; t_{k+1})$

     Gradient correction $\rightarrow$ Gradient trajectory tracking (GTT)

- return
Predictor-corrector schemes

Centralized algorithms for trajectory tracking

- start at $x_0$, fix step-size $\gamma > 0$, sampling rate $h = t_{k+1} - t_k$

1. Prediction step: predict $x_{k+1|k}$ based on objective at time $t_k$

$$x_{k+1|k} = x_k - h \left( \nabla xx f(x_k; t_k) \right)^{-1} \nabla tx f(x_k; t_k)$$

2. Correction step: update prediction via objective at time $t_{k+1}$

$$x_{k+1} = x_{k+1|k} - \left( \nabla xx f(x_{k+1|k}; t_{k+1}) \right)^{-1} \nabla x f(x_{k+1|k}; t_{k+1})$$

Newton correction $\rightarrow$ Newton trajectory tracking (NTT)

- return
Main result 1

Theorem 1. (idea)

1. By using a gradient-based correction we obtain, (state-of-the-art)

\[
\lim_{k \to \infty} \| x_k - x^*(t_k) \| = O(h)
\]
Main result 1

Theorem 1. (idea)

1. By using a gradient-based correction we obtain, (state-of-the-art)

\[ \lim_{k \to \infty} \| x_k - x^*(t_k) \| = O(h) \]

2. By using a prediction and gradient-based correction there exist small enough \( h \) and \( \gamma \) such that

\[ \lim_{k \to \infty} \| x_k - x^*(t_k) \| = O(h^2) \]

3. By using a prediction and Newton’s-based correction there exist small enough \( h \) such that

\[ \lim_{k \to \infty} \| x_k - x^*(t_k) \| = O(h^4) \]
Main result 1

Theorem 1. (idea)

1. By using a gradient-based correction we obtain, (state-of-the-art)

\[ \lim_{k \to \infty} \| x_k - x^*(t_k) \| = O(h) \]

4. By using a prediction and gradient/Newton’s-based correction for all \( h \), there exist a \( \gamma < 2/L \) such that

\[ \lim_{k \to \infty} \| x_k - x^*(t_k) \| = O(h) \]
Additional result

**Approximate time derivative**

What if we do not know $\nabla_{tx} f(x_k; t_k)$?
Additional result

**Approximate time derivative**

What if we do not know $\nabla_{tx} f(x_k; t_k)$?

No problem: use backward derivative approximation:

$$\nabla_{tx} f(x_k; t_k) \approx \frac{\nabla_x f(x_k; t_k) - \nabla_x f(x_{k-1}; t_{k-1})}{h}$$

and we still have $O(h^2)$ error, and $O(h^4)$ for Newton correction
Numerical result: simple example

\[ \min_{x \in \mathbb{R}} f(x; t) := \frac{1}{2} (x - \cos(\omega t))^2 + \frac{\kappa}{2} \sin^2(\omega t) \exp(\mu x^2) \]

- Prediction-correction methods yield accurate solution trajectory tracking
- Better to use Newton correction step
**Numerical result: target tracking**

\[
\minimize_{x \in \mathbb{R}^2} f(x; t) := \frac{1}{2} \left( \|x - y(t)\|^2 + \mu_1 \exp(\mu_2 \|x - b\|^2) \right),
\]

- Track moving planar object \( y(t) \in \mathbb{R}^2 \), stay close to base-station \( b \in \mathbb{R}^2 \)

- Simple integral control \( \rightarrow \) effectively track reference path
- Can only use Newton correction step for sufficiently large sampling interval
- When can use NTT (\( h \) large enough), much better tracking performance
Networked problems


Networked setting

$N$ nodes and an undirected graph $G = (V, E)$, $V = \{1, \ldots, N\}$

$$\min \limits_{y^1 \in \mathbb{R}^m, \ldots, y^N \in \mathbb{R}^m} F(y; t) := f(y, t) + g(y; t)$$

$f$ is sum of local functions, and $g$ is induced by network structure:

$$f(y; t) := \sum_{i \in V} f^i(y^i; t) , \quad g(y; t) := \sum_{i \in V} g^{i,i}(y^i; t) + \sum_{(i,j) \in E} g^{i,j}(y^i, y^j; t)$$

where we stack all node variables $y = (y^1; \ldots; y^N)$
Continuous-time networked optimization

→ Approximate distributed opt (Cooperative automotive control)
→ Estimation of spatial fields (Predict wind energy generation)
→ Wireless network resource allocation (Internet of Things)
Continuous-time networked optimization

Technical conditions
We assume here $f$ and $g$ to be smooth (for each $y$ and uniformly in $t$):

- $f^i$ and $g^{i,j}$ doubly differentiable, $f^i$ $m$-strongly convex, for $f^i$ gradient $L$ Lipschitz
- higher derivatives (both in time and “space”) bounded
- the hessian of $g$ upper bounded by $M$ and *diagonally dominant*
Prediction-correction methods in decentralized settings

\[
\min_{y_1 \in \mathbb{R}^m, \ldots, y_N \in \mathbb{R}^m} F(y; t) := f(y; t) + g(y; t)
\]

A not-so-small issue:
When we write the prediction step the Hessian of \( F(y; t) \) appears

\[
y_{k+1|k} = y_k - (\nabla_{yy} F(y_k; t_k))^{-1} \nabla_{ty} F(y_k; t_k) h
\]

Does not have the same sparsity structure as \( G \)
Hessian inverse approximation

**Idea:** under the given assumptions (especially diagonally dominance), we can compute an approximate prediction direction in $K$ rounds of communication

$$(\nabla_{yy} F(y_k; t_k))^{-1} \nabla_{ty} F(y_k; t_k) \approx d_{k,K}$$

in a distributed way, with error w.r.t to true inverse bounded as

$$e_k \leq \left( \frac{L/2}{m + L/2} \right)^{K+1}$$

Related to Lipschitz constant $L$ and $m$-strong convexity parameter
**Hessian inverse approximation**

**Catch:** Define $D_k := \nabla_{yy} f(y_k; t_k) + \text{diag} [\nabla_{yy} g(y_k; t_k)]$, 

$B_k := \text{diag} [\nabla_{yy} g(y_k; t_k)] - \nabla_{yy} g(y_k; t_k)$.

Then may write Hessian as matrix difference

$$\nabla_{yy} F(y_k; t_k) = D_k - B_k,$$

Conjugate $B_k$ by $D_k^{-1/2}$, and factor out common $D_k^{1/2}$:

$$\nabla_{yy} F(y_k; t_k) = D_k^{1/2} (I - D_k^{-1/2} B_k D_k^{-1/2}) D_k^{1/2}$$
Hessian inverse approximation

**Catch:** Using factored Hessian

\[ \nabla_{yy} F(y_k; t_k) = D_k^{1/2} (I - D_k^{-1/2} B_k D_k^{-1/2}) D_k^{1/2}, \]

Compute inverse, using the fact that \( D_k \succ 0 \)

\[
(\nabla_{yy} F(y_k; t_k))^{-1} = D_k^{-1/2} \left( I - D_k^{-1/2} B_k D_k^{-1/2} \right)^{-1} D_k^{-1/2}
\]

(Taylor expansion)

\[
= D_k^{-1/2} \sum_{\tau=0}^{\infty} \left( D_k^{-1/2} B_k D_k^{-1/2} \right)^\tau D_k^{-1/2}
\]

\((K\text{-truncated series})\)

\[
\approx D_k^{-1/2} \sum_{\tau=0}^{K} \left( D_k^{-1/2} B_k D_k^{-1/2} \right)^\tau D_k^{-1/2}
\]
Hessian inverse approximation

Recursion procedure for communication budget $K$:

- Node $i$ initializes its prediction direction at time $t_k$
  \[ d_{k,(0)}^i = -(D_k^{ii})^{-1} \nabla_{ty} F^i(y_k; t_k) \]

- Then recurses over $\tau = 0, \ldots, K \rightarrow \text{info. exchange } j \in N^i$
  \[ d_{k,(\tau+1)}^i = -(D_k^{ii})^{-1} \left( \sum_{j \in N^i} B_k^{ij} d_{k,(\tau)}^j + \nabla_{ty} F^i(y_k; t_k) \right) , \]

where local mixed partial derivative w.r.t. node $i$ is
\[
\nabla_{ty} F^i(y_k; t_k) = \nabla_{ty} f^i(y_k^i; t_k) + \nabla_{ty} g^{i,i}(y_k^i; t_k) \\
+ \sum_{j \in N^i} \nabla_{ty} g^{i,j}(y_k^i, y_k^j; t_k). 
\]
Distributed algorithm

Decentralized Prediction-Correction

- initialize $y^i_0$, fix step-size $\gamma > 0$, sampling rate $h = t_{k+1} - t_k$ 
- generate a sequence $\{y^i_k\}$ as:
  1. Generate approx. prediction direction with $K$ comm. rounds 
  2. Prediction step: via info. at time $t_k$, predict optimizer at $t_{k+1}$

$$y^i_{k+1|k} = y^i_k - hd^i_{k,(K)}$$

- Gradient correction $\rightarrow$ Decentralized Gradient tracking (DeGT)

$$y^i_{k+1} = y^i_{k+1|k} - \gamma \nabla y F^i(y^i_{k+1|k}; t_{k+1})$$

- return
Distributed algorithm

Decentralized Prediction-Correction

1. Initialize $y^i_0$, fix step-size $\gamma > 0$, sampling rate $h = t_{k+1} - t_k$

2. Generate a sequence $\{y^i_k\}$ as:
   1. Generate approx. prediction direction with $K$ comm. rounds
   2. Prediction step: via info. at time $t_k$, predict optimizer at $t_{k+1}$

$$y^i_{k+1|k} = y^i_k - h d^i_{k,(K)}$$

3. Newton correction $\rightarrow$ Decentralized Newton tracking (DeNT)

$$y^i_{k+1} = y^i_{k+1|k} - \gamma c^i_{k+1,K'}$$

$c^i_{k+1,K'}$ is approximate Newton step $\rightarrow$ via same Hessian approx. technique on updated objective $F(y^i_{k+1|k}; t_{k+1})$

4. Return
Main result 2

Theorem 2. (idea)

1. By using a prediction and gradient-based correction there exist small enough $h$ and $\gamma$ such that

$$\lim_{k \to \infty} \|y_k - y^*(t_k)\| = O(h^2) + \left(\frac{L/2}{m + L/2}\right)^{K+1} O(h)$$
Main result 2

Theorem 2. (idea)

1. By using a prediction and gradient-based correction there exist small enough $h$ and $\gamma$ such that

$$\lim_{k \to \infty} \|y_k - y^*(t_k)\| = O(h^2) + \left(\frac{L/2}{m + L/2}\right)^{K+1} O(h)$$

2. Define $\rho = \left(\frac{L/2}{m + L/2}\right)$, then select communication budget $K$

$$K \geq \left\lceil \frac{\log h}{\log \rho} - 1 \right\rceil,$$

then we obtain more accurate tracking performance

$$\lim_{k \to \infty} \|y_k - y^*(t_k)\| = O(h^2)$$
Setting

We consider a WSN estimating the intensity of a two dimensional spatial circular wave

\[
c(\xi; t) = \frac{\cos(2\pi \omega (t - \|\xi - \xi_0\|/v))}{4\pi \|\xi - \xi_0\|},
\]

Each sensor node at \(\xi_i\) estimates the intensity of the wave as

\[
\hat{c}(\xi_i; t) = c(\xi_i; t) + \eta_i, \quad \eta_i \sim N(0, q),
\]

\[
\text{minimize } F(y; t) := \sum_{i \in V} \left( \frac{1}{2q} \|y_i - \hat{c}(\xi_i; t)\|^2 + \frac{\beta}{q} \sum_{j \in N_i} \frac{w_{ij}}{2} \|y_i - y_j\|^2 \right)
\]
Numerical result

- Performance comparable to centralized methods
- Orders of magnitude better than running methods
- Accuracy vs. comms. tradeoff: larger $K \rightarrow$ better accuracy
Conclusions

- Time-varying convex opt: brute force approach intractable
- Discrete-time sampling via prediction-correction
- Predict where optimal trajectory will be at next time
- Once we observe info. at next sample time, make correction
- Correction step $\rightarrow$ Gradient or Newton steps (GTT or NTT)

- Theorem: $\mathbf{x}^*(t)$ up to $O(h^2)$ or better
- Prediction-correction $\rightarrow$ better tracking w.r.t existing methods

- Decentralized extensions via approximated Hessian inverse
- Comparable tracking guarantees to centralized methods
- Numerical examples $\rightarrow$ utility in control & comms. domain