Optimization and other Applied Mathematics Challenges for Complex Energy Systems

Mihai Anitescu

Mathematics and Computer Science Division

Argonne National Laboratory

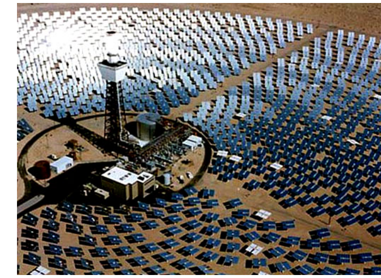
IMSE Illinois 2012

1. Motivation: Management of Energy Systems under Ambient Conditions Uncertainty

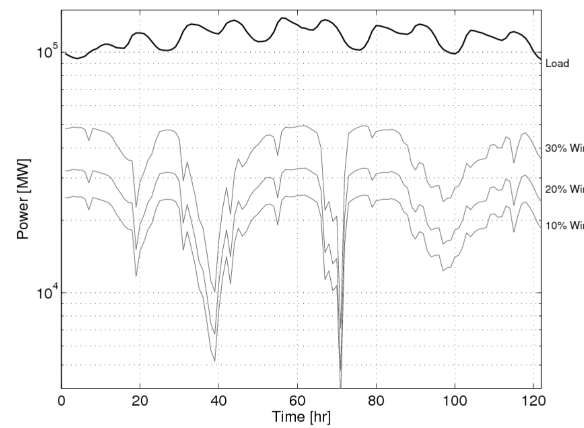
Ambient Condition Effects in Energy Systems

Operation of Energy Systems is Strongly Affected by Ambient Conditions

- **Power Grid Management:** Predict Spatio-Temporal Demands *(Douglas, et.al. 1999)*
- **Power Plants:** Generation levels affected by air humidity and temperature *(General Electric)*
- **Petrochemical:** Heating and Cooling Utilities *(ExxonMobil)*
- **Buildings:** Heating and Cooling Needs *(Braun, et.al. 2004)*
- (Focus) **Next Generation Energy Systems** assume a major renewable energy penetration: Wind + Solar + Fossil *(Beyer, et.al. 1999)*



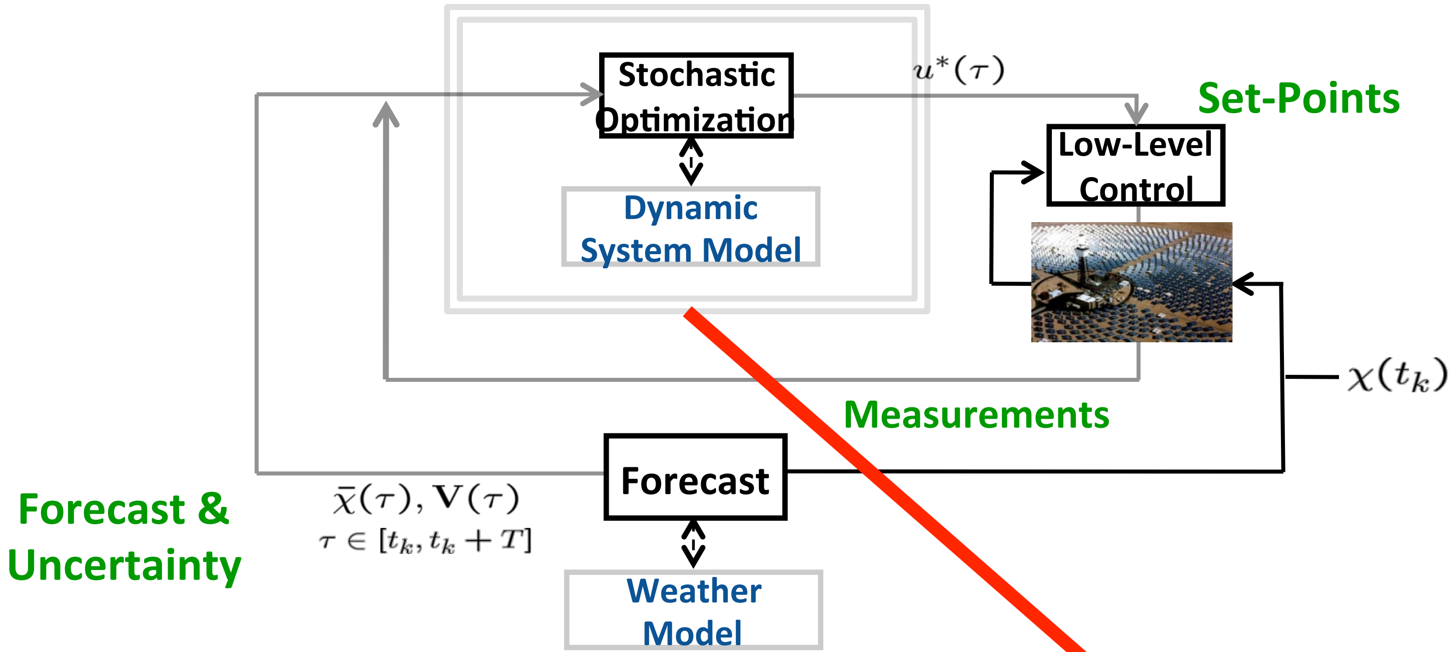
- Increased reliance on renewables must account for variability of ambient conditions, which cannot be done deterministically …
- We must optimize operational and planning decisions accounting for the uncertainty in ambient conditions (and others, e.g. demand)
- **Optimization Under Uncertainty.**



Wind Power Profiles

2. Impact: Stochastic Unit Commitment – Management of Energy Systems

Stochastic Predictive Control



Two-stage Stoch Prog

$$\underset{x_0}{Min}\left\{ f_0(x_0) + \mathbb{E}\left[\underset{x}{Min}\, f(x,\omega) \right] \right\}$$

subj. to.

$$\begin{aligned} g_0(x_0) &= b_0 \\ g_i(x_0, x_i) &= b_i \quad i = 1,2\ldots S \\ x_0 \geq 0, &\quad x_i \geq 0 \end{aligned}$$

Stochastic NLMPC

$$\min_{u(t)} \; \mathbf{E}_{\chi(t)\in\Omega} \left[\int_{t_\ell}^{t_{\ell+N}} \varphi(z(t), y(t), u(t), \chi(t))\,dt \right]$$

$$\left.\begin{aligned} \frac{dz}{dt} &= \mathbf{f}(z(t), y(t), u(t), \chi(t)) \\ 0 &= \mathbf{g}(z(t), y(t), u(t), \chi(t)) \\ 0 &\geq \mathbf{h}(z(t), y(t), u(t), \chi(t)) \end{aligned}\right\}$$

$$z(0) = x_\ell$$

Stochastic Unit Commitment with Wind Power (SAA)

$$\min \quad \text{COST} = \frac{1}{N_s} \sum_{s \in S} \left(\sum_{j \in N} \sum_{k \in T} c^p_{sjk} + c^u_{jk} + c^d_{jk} \right)$$

$$\text{s.t.} \quad \sum_{j \in N} p_{sjk} + \sum_{j \in N_{wind}} p^{wind}_{sjk} = D_k, s \in S, k \in T$$

$$\sum_{j \in N} \overline{p}_{sjk} + \sum_{j \in N_{wind}} p^{wind}_{sjk} \geq D_k + R_k, s \in S, k \in T$$

ramping constr., min. up/down constr.

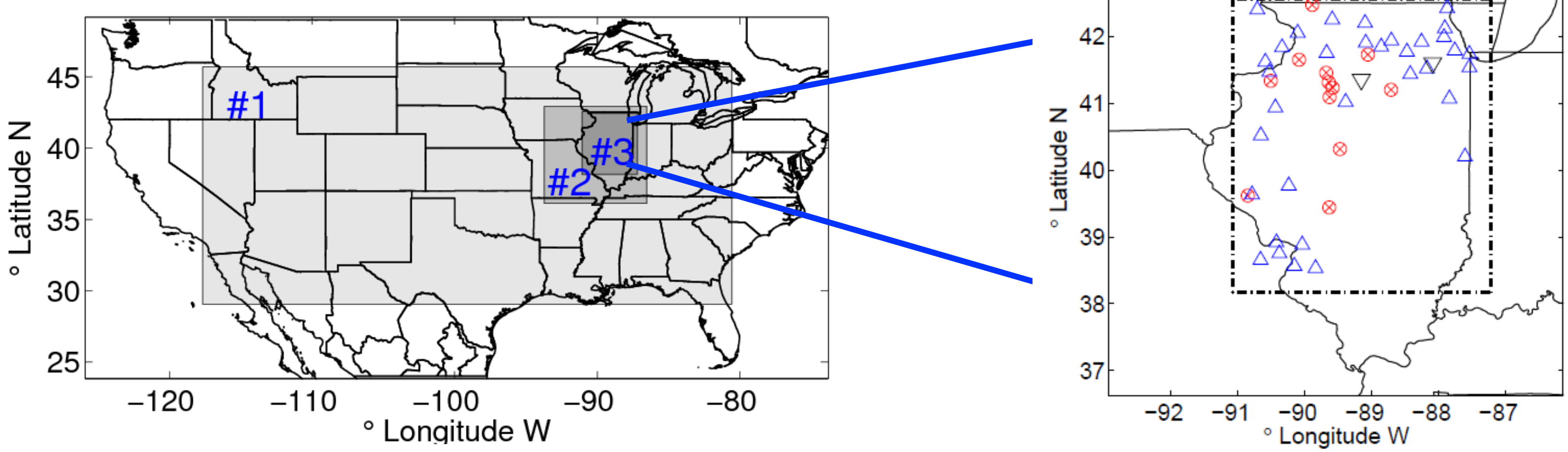
Thermal Units Schedule? Minimize Cost

Satisfy Demand

Have a Reserve

Dispatch through network

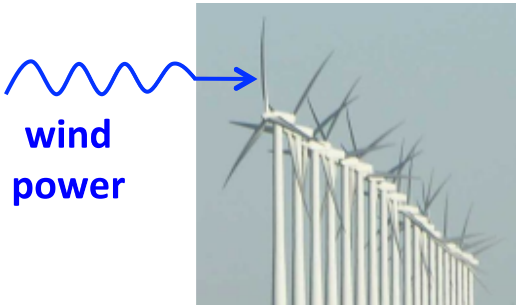
- Wind Forecast – WRF(Weather Research and Forecasting) Model
  - Real-time grid-nested 24h simulation
  - 30 samples require 1h on 500 CPUs (Jazz@Argonne)



Zavala & al 2010.

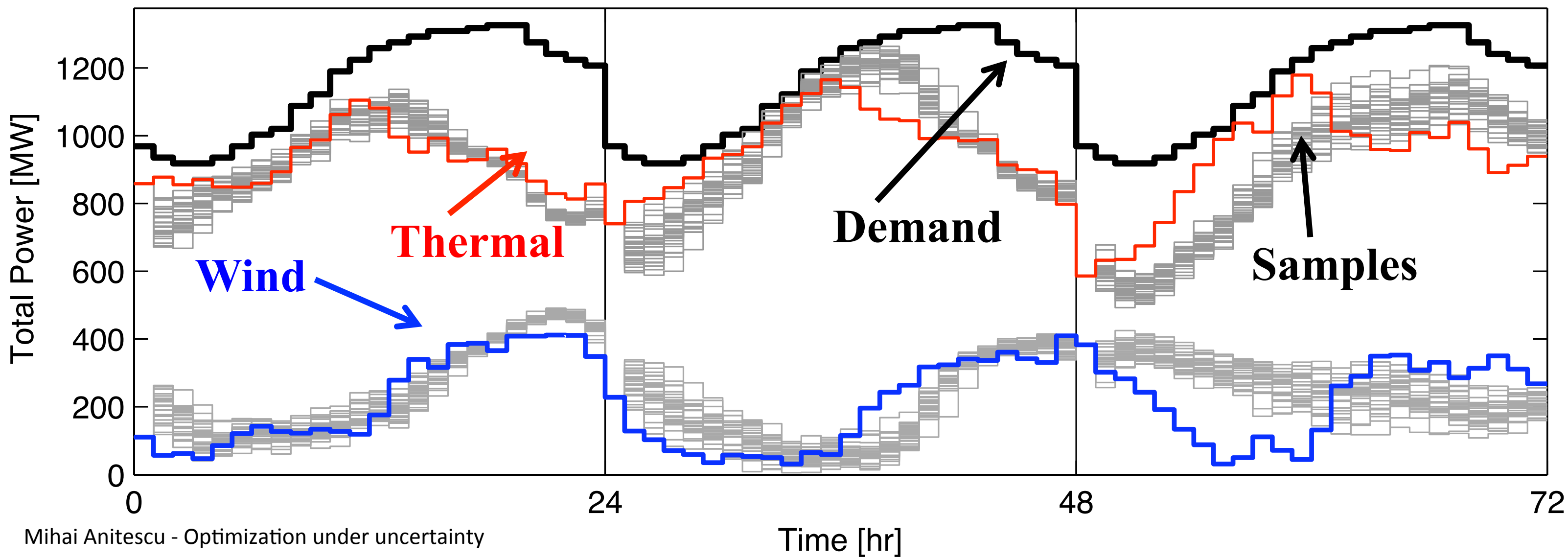
Wind power forecast and stochastic programming

- Unit commitment & energy dispatch with uncertain wind power generation for the State of Illinois, assuming 20% wind power penetration, using the same windfarm sites as the one existing today.



- Full integration with 10 thermal units to meet demands. Consider dynamics of start-up, shutdown, set-point changes

- The solution is only 1% more expensive then the one with exact information. Solution on average infeasible at 10%.



Some Considerations in Using Supercomputing for Power Grid

- Is it really worth using a supercomputer for this task? (We need the answer every 1hr with 24 hour time horizons.)
- Let's look at the most pressing item of Supercomputing usage: power.
  - BG/P (and exascale) needs <~ 20MW of power.
  - The Midwest US has 140GW of power installed, and the peak demands runs up to 110GW.
  - We will never reduce power consumption, but we will make it more reliable, less dependent on fossil, and cheaper by better managing the peak
- If we accept this will lead to 10% more renewable penetration (our SUC study), then this is worth on the order of 10-15GW, far above what BG/P costs in power consumption.
- In addition operational constraints makes supercomputing (if uncertainty needed to account for) necessary and not just useful or convenient.
- But, even if approximations will work, this tool will be helpful as the "gold standard" for validating other algorithms to be deployed on defined computational resources.

3. Low-Hanging Fruit Scalable Software: PIPS (Parallel Interior Point Stochastic Programming) – Petra, Lubin, Anitescu

PIPS – Our Scalable Stochastic Programming Solver Using Direct Schur Complement Method

- The arrow shape of H

$$\begin{bmatrix} H_1 & & & & G_1^T \\ & H_2 & & & G_2^T \\ & & \ddots & & \vdots \\ & & & H_S & G_S^T \\ G_1 & G_2 & \dots & G_S & H_0 \end{bmatrix} = \begin{bmatrix} L_1 & & & & \\ & L_2 & & & \\ & & \ddots & & \\ & & & L_S & \\ L_{10} & L_{20} & \dots & L_{S0} & L_c \end{bmatrix} \begin{bmatrix} D_1 & & & & \\ & D_2 & & & \\ & & \ddots & & \\ & & & D_N & \\ & & & & D_c \end{bmatrix} \begin{bmatrix} L_1^T & & & & L_{10}^T \\ & L_2^T & & & L_{20}^T \\ & & \ddots & & \vdots \\ & & & L_S^T & L_{S0}^T \\ & & & & L_c^T \end{bmatrix}$$

- Solving Hz=r

$$L_i D_i L_i^T = H_i,\; L_{i0} = G_i L_i^{-T} D_i^{-1}, \quad i = 1,\ldots S,$$

$$C = H_0 - \sum_{i=1}^{S} G_i H_i^{-1} G_i^T,\; L_c D_c L_c^T = C.$$

Implicit factorization, C is **dense**, H's are sparse.

Back substitution:

$$w_i = L_i^{-1} r_i, \quad i = 1,\ldots S,$$

$$w_0 = L_c^{-1}\left(r_0 - \sum_{i=1}^{S} L_{i0} w_i\right)$$

Diagonal solve:

$$v_i = D_i^{-1} w_i,\; i = 0,\ldots,S$$

Forward substitution:

$$z_0 = L_c^{-1} v_0$$

$$z_i = L_i^{-T}\left(v_i - L_{i0}^T z_0\right), \quad i = 1,\ldots S.$$

Parallelizing the 1st stage linear algebra

- We distribute the 1st stage Schur complement system.

$$C = \begin{bmatrix} \tilde{Q} & A_0^T \\ A_0 & 0 \end{bmatrix}, \tilde{Q} \text{ dense symm. pos. def., } A_0 \text{ sparse full rank.}$$

- C is treated as dense.
- Alternative to PSC for problems with large number of 1st stage variables.
- Removes the memory bottleneck of PSC and DSC.
- We investigated ScaLapack, Elemental (successor of PLAPACK)
  - None have a solver for symmetric indefinite matrices (Bunch-Kaufman);
  - LU or Cholesky only.
  - So we had to think of modifying either.

Cholesky-based LDL^T-like factorization

- $$\begin{bmatrix} \tilde{Q} & A^T \\ A & 0 \end{bmatrix} = \begin{bmatrix} L & 0 \\ AL^{-T} & \bar{L} \end{bmatrix} \begin{bmatrix} I & \\ & -I \end{bmatrix} \begin{bmatrix} L^T & L^{-1}A^T \\ 0 & \bar{L}^T \end{bmatrix}, \text{ where } LL^T = \tilde{Q},\ \bar{L}\bar{L}^T = A\tilde{Q}^{-1}A^T$$

- Can be viewed as an "implicit" normal equations approach.
- In-place implementation inside Elemental: no extra memory needed.
- Idea: modify the Cholesky factorization, by changing the sign after processing p columns.
- It is much easier to do in Elemental, since this distributes elements, not blocks.
- Twice as fast as LU
- Works for more general saddle-point linear systems, *i.e.,* pos. semi-def. (2,2) block.

Distributing the 1st stage Schur complement matrix

- **All** processors contribute to **all** of the elements of the (1,1) dense block

$$\tilde{Q} = \widetilde{Q}_0 + \frac{1}{S}\sum_{i=1}^{S}\left[A_i^T \left(B_i \widetilde{Q}_i^{-1} B_i^T \right)^{-1} A_i \right]$$

- A large amount of inter-process communication occurs.

- Possibly more costly than the factorization itself.

- Solution: use buffer to reduce the number of messages when doing a *Reduce_scatter.*

- LDL^T approach also reduces the communication by half – only need to send lower triangle.

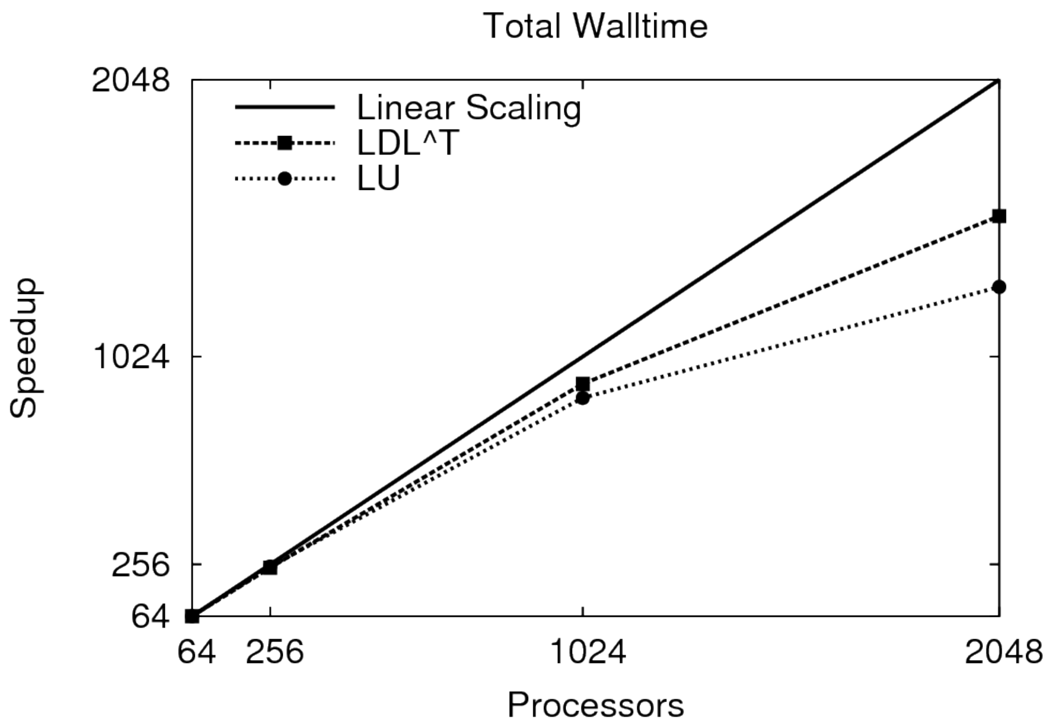
Large-scale performance

- Comparison of ScaLapack (LU), Elemental(LU), and LDL^T (1024 cores)

Units	1st Stage Size $(Q+A)$	Factor (Sec.) LU(S)	Factor (Sec.) LU(E)	Factor (Sec.) LDL^T	Reduce (Sec.) LU	Reduce (Sec.) LDL^T
300	23436+1224	16.59	20.04	**6.71**	54.32	**26.35**
640	49956+2584	60.67	83.24	**36.77**	256.95	**128.59**
1000	78030+4024	173.67	263.53	**90.82**	565.36	**248.22**

SAA problem:
189 million variables

- Strong scaling
  - **90.1%** from 64 to 1024 cores;
  - 75.4% from 64 to 2048 cores.
  - > 4,000 scenarios.



PIPS – Parallel solver for stochastic optimization

- Interior-point method implementation (Mehrotra's algorithm)
- Scenario-based decomposition of the linear algebra.
- PIPS reuses OOQP (Object-oriented quadratic programming solver) class hierarchy.
- New parallel linear algebra layer for block-angular IPM linear systems.
- Hybrid MPI+SMP parallelization

- First-stage Schur complement: **dense** linear algebra.
  - Distributed factorization and backsolves (by using Elemental) if needed.
  - Shared-memory parallelization(SMP) is obtained via Elemental.
  - Distributed assembling of the SC matrix is done by a streamlined **Reduced_scatter** that is also in-node SMP-accelerated.

- Second-stage linear systems are **sparse**.
  - Supports various sparse solvers: MA57 (HSL UK), WSMP(IBM).
  - SMP is obtained with WSMP

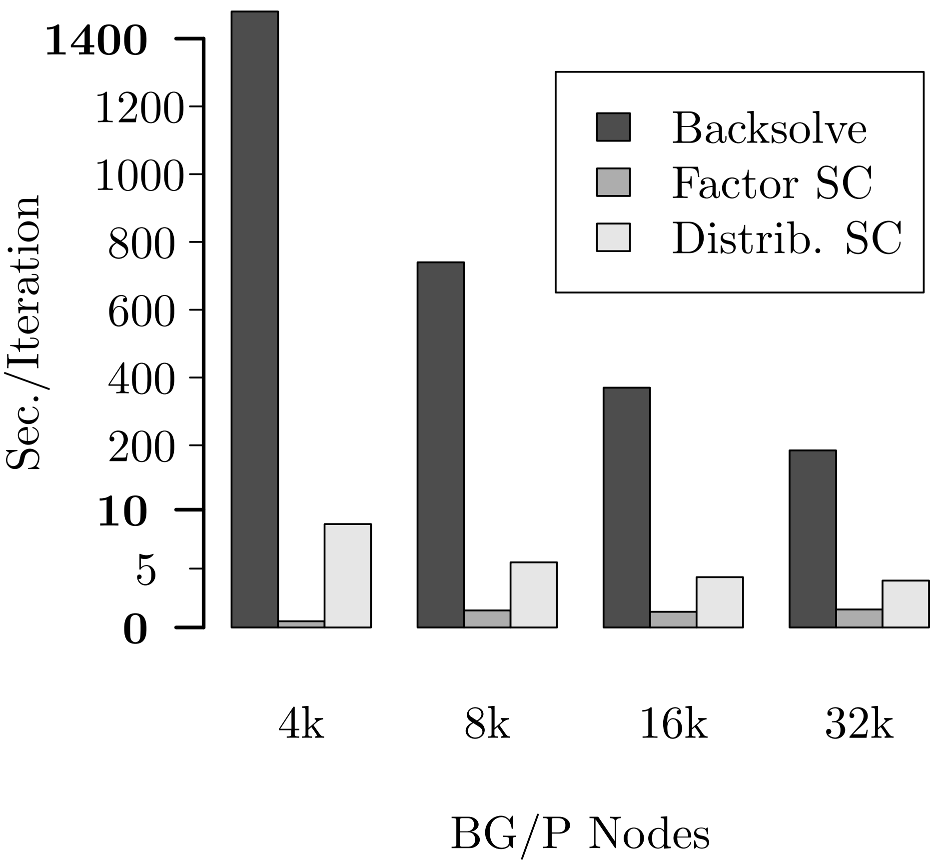
PIPS Solver Capabilities

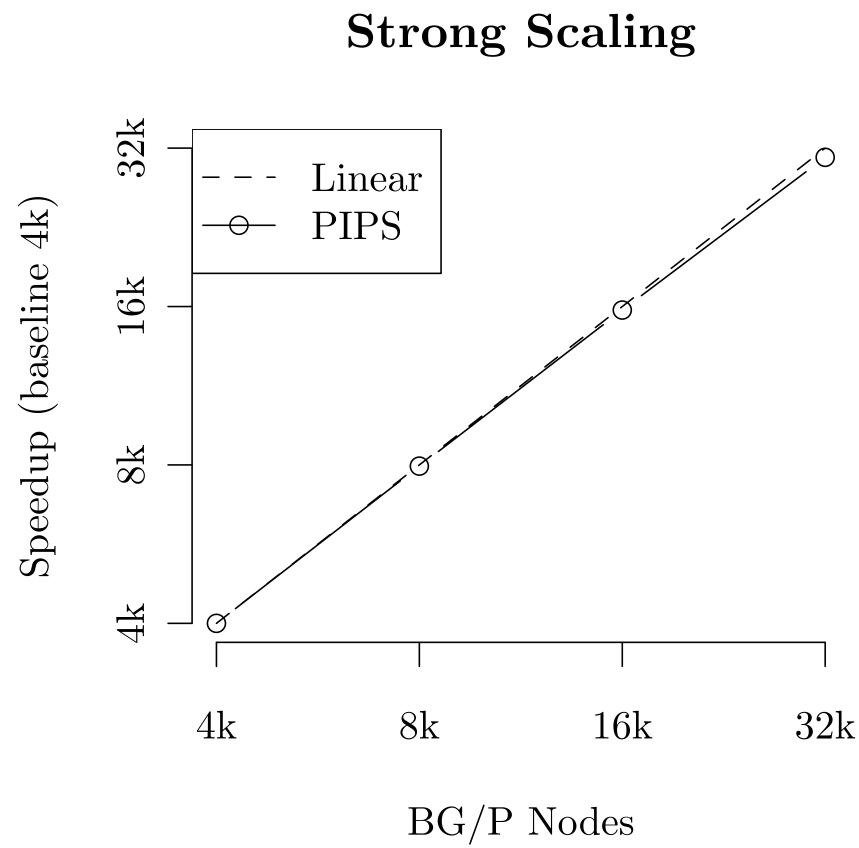
- Hybrid MPI/SMP running on Blue Gene/P
  - Successfully (though incompletely due to allocation limit) run on up to **32,768** nodes (**96**% strong scaling) for Illinois problem with grid constraints. 3B variables, maybe largest ever solved?

- Handles up to 100,000 first-stage variables. Previous results dealt with O(20-50).

- Close to real-time solutions (24 hr horizon in 1 hr wallclock)
  - Further development needed, since users aim for
    - More uncertainty, more detail (x 10)
    - Faster Dynamics → Shorter Decision Window (x 10)
    - Longer Horizons (California == 72 hours) (x 3)

Components of Execution Time and Strong Scaling





- 32K nodes=130K cores (80% BG/P)
- "Backsolve" phase embarrassingly parallel, but not Schur Complement (SC)
- Communication for "Distrib. SC" not yet a bottleneck, but we will get there.

4. The harder problems need some mathematics

4.1 Q1: How do I deal with the impending first-stage bottleneck?

The Stochastic Preconditioner

- The limiting factor in the scalability of Schur method is the expensive solve with **dense** Schur complement matrix

$$C = \widetilde{Q}_0 + \frac{1}{N}\sum_{i=1}^{N}\left[A_i^T \left(B_i \widetilde{Q}_i^{-1} B_i^T \right)^{-1} A_i \right]$$

  - A computational bottleneck: workers sit idle waiting for the master to factorize C.

- Remedy – the preconditioned Schur complement (PSC)
  - 1. factorize incomplete matrix P in the same time C is computed.
  - 2. use the factorization of P to solve with C very fast.

- In linear algebra terms
  - P is a preconditioner for C. Our choice of P is

$$S_n = \widetilde{Q}_0 + \frac{1}{n}\sum_{i=1}^{n}\left[A_{k_i}^T \left(B_{k_i} \widetilde{Q}_{k_i}^{-1} B_{k_i}^T \right)^{-1} A_{k_i} \right],$$

  where $K = \{k_1, k_2, K, k_n\}$ is an IID subset of n scenarios.
  - Krylov iteratives solves (PCG or BiCGStab) replaces the direct solves

Preconditioned Schur Complement (PSC)

P is a C built from a subset of scenarios

$$L_i D_i L_i^T = H_i,\ \ L_{i0} = G_i L_i^{-T} D_i^{-1}, \quad i = 1,\ldots,N,$$

$$C = H_0 - \sum_{i=1}^{N} G_i H_i^{-1} G_i^T$$

$$L_M D_M L_M^T = P$$

(separate process)

$$w_i = L_i^{-1} r_i$$

$$\tilde{r}_0 = \left(r_0 - \sum_{i=1}^{N} L_{Ni} w_i \right)$$

$$v_i = D_i^{-1} w_i,\ \ i = 0,\ldots,N$$

$$z_0 = \mathrm{Krylov}(C, P, \tilde{r}_0)$$

$$z_i = L_i^{-T}\left(v_i - L_{i0}^T z_0 \right)$$

Quality of the Stochastic Preconditioner

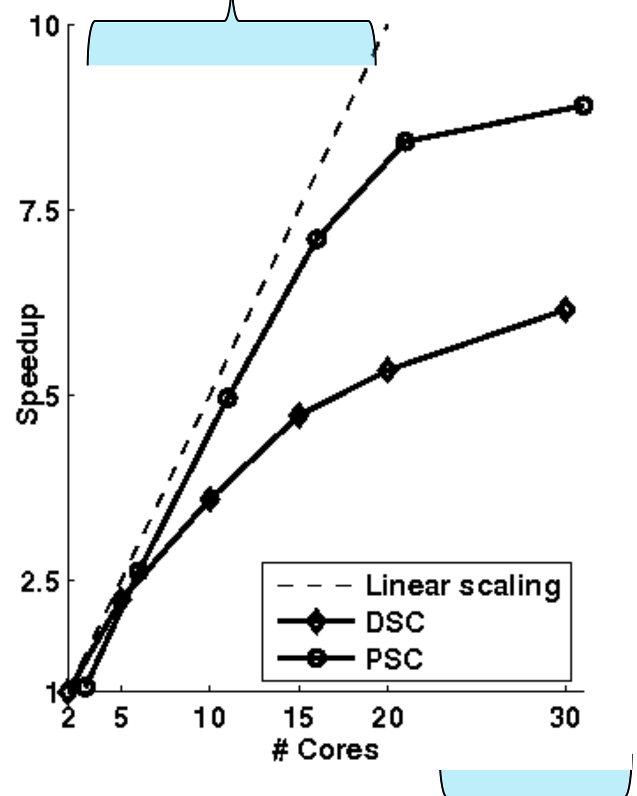
- **"Exponentially" better preconditioning** (Petra & Anitescu 2010)

$$\Pr(|\lambda(S_n^{-1}S_N)-1|\geq\varepsilon)\leq 2p^4\exp\left(-\frac{n\varepsilon^2}{2p^4L^2\|S_N\|_{max}^2}\right)$$

- A typical scaling behavior of our approach. Better scaling than the direct Schur complement method (DSC) is exhibited by PSC.

- DSC uses p processes, PSC uses p+1.

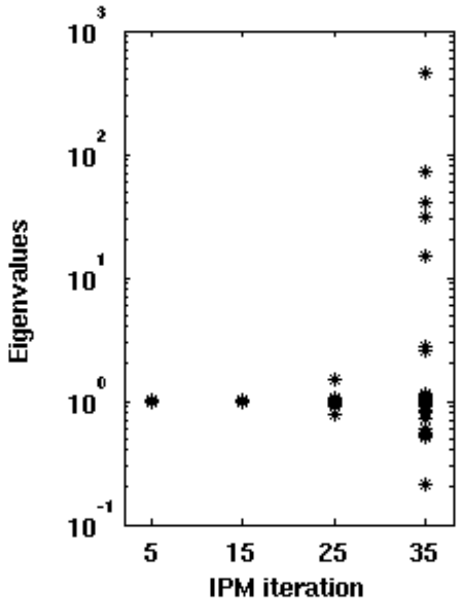
Optimal use of PSC – **linear scaling**

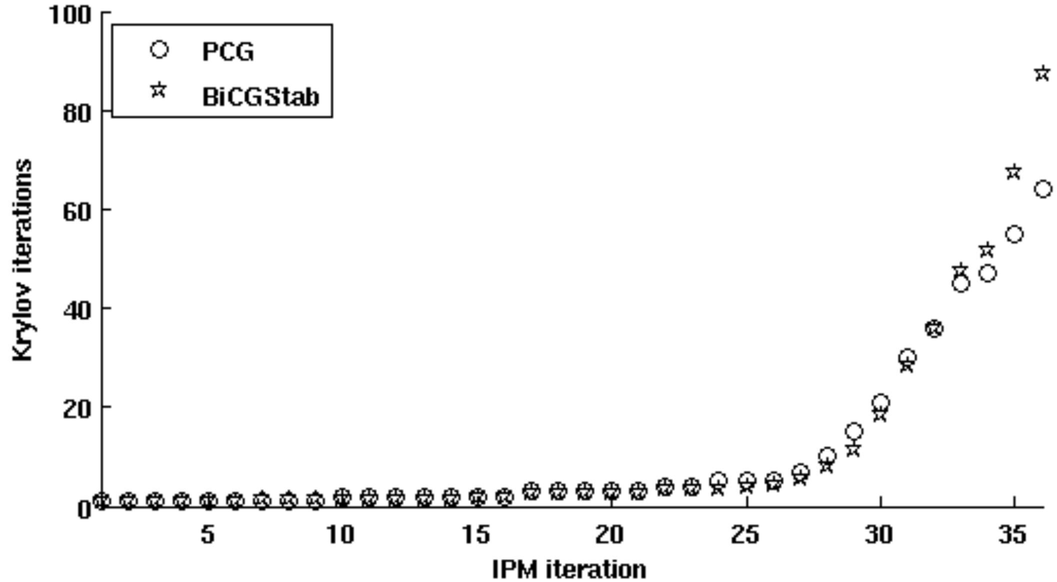


Factorization of the preconditioner can not be hidden anymore by the computation of C.

Performance of the preconditioner

- Eigenvalues clustering & Krylov iterations





- Affected by the well-known ill-conditioning of IPMs.

$$S_n \approx S_N \text{ and } \mathrm{S}_N \approx \mathbb{E}\left[S(\omega)\right], \text{ where}$$

$$S(\omega) = \left(Q_0 + D_0\right) + \left[A^T(\omega)\left(B(\omega)\left(Q(\omega) + D(\omega)\right)^{-1} B^T(\omega)\right)^{-1} A(\omega)\right]$$

4.2 Q2: How do I use very expensive samples?

Bootstrap for stochastic optimization of energy systems

- Sampling the uncertainty present in complex energy systems may be a computationally intensive task
  - Example: weather forecasting
  - May need 400K CPUs for 30 samples at the resolution we need.
- Obtaining uncertainty estimates (confidence intervals) on the optimal value is important in policy-making process.
- Only a small number of samples(scenarios) can be afforded. Therefore an operational constraint makes me start to care about the low-sample size regime and its asymptotics.
- But how good is the current state of the theory in that regime?

Theory situation for Stochastic Programming

- Most estimates for SAA are based on results of the following type:

$$N^{0.5}\left[\frac{\theta-\hat{\theta}}{\hat{\sigma}}\right]\xrightarrow{D} N(0,1)$$

- This allows, in principle, for the convergence of the confidence intervals to be arbitrarily slow.
- Current state of the area is built around application of the Delta Theorem , which provides the results of the type:

$$X_N(\omega)-X(\omega)=o_P(N^{-a}) \Leftrightarrow P\left(N^a\left|X_N(\omega)-X(\omega)\right|\right)\to 0$$

- But this is not sufficient for similar results for the confidence intervals !!!

$$X(\omega)=\omega,\quad X_N(\omega)=\begin{cases}-1, & 0\le\omega<\dfrac{1}{\log(N+1)}\\ \omega, & \dfrac{1}{\log(N+1)}\le\omega\le 1.\end{cases} \Rightarrow \begin{cases} P(\lim_{N\to\infty} N^a\left|X_N(\omega)-X(\omega)\right|=0)=1 \\ P(X_N(\omega)\le 0)-P(X(\omega)\le 0)=\dfrac{1}{\log(N+1)}\gg N^{-b}\end{cases}$$

- Intuition: Convergence in probability tells me how well I behave on a “good” set increasing to probability 1, but tells me nothing about the bad set.

Large Deviation + Bootstrap

- Bootstrap is a resampling method that builds high-order confidence estimates.
- The idea of bootstrapping is to squeeze out information from a small number of samples by resampling (with replacement).
- But it applies only to finite-dimensional functions of means, and the optimal value of stochastic optimization is not one (due to nonlinearity).

- Idea: Use large deviations, to produce exponentially convergent probability sets

$$\mathbb{P}(| N^b(\hat{\theta}-\theta) | > \epsilon) = f_1(\epsilon) N^{f_2(\epsilon)} \exp(-f_3(\epsilon) N^c)$$

- Here, $\hat{\theta}$ depends on the expected value of the objective function and its higher derivatives at the solution of the SAA approximation problem. We can thus use bootstrap theory to produce confidence intervals for $\hat{\theta}$ and exponential convergence ensures order stays the same as for bootstrap !!

The estimator

- We proposed a corrected statistic, computable at the SAA approximation

$$\Phi = \mathbb{E}\left[f(x^N) \right] - \frac{1}{2} \begin{pmatrix} \mathbb{E}\nabla L(x^N, \lambda^N) \\ 0 \end{pmatrix}^T \begin{pmatrix} \mathbb{E}\nabla^2 L(x^N, \lambda^N) & J(x^N)^T \\ J(x^N) & 0 \end{pmatrix}^{-1} \begin{pmatrix} \mathbb{E}\nabla L(x^N, \lambda^N) \\ 0 \end{pmatrix}$$

- L is the Lagrangian of the problem and J is the Jacobian of the constraints.
- x^N is the solution of the SAA problem obtained from a sample of size N.

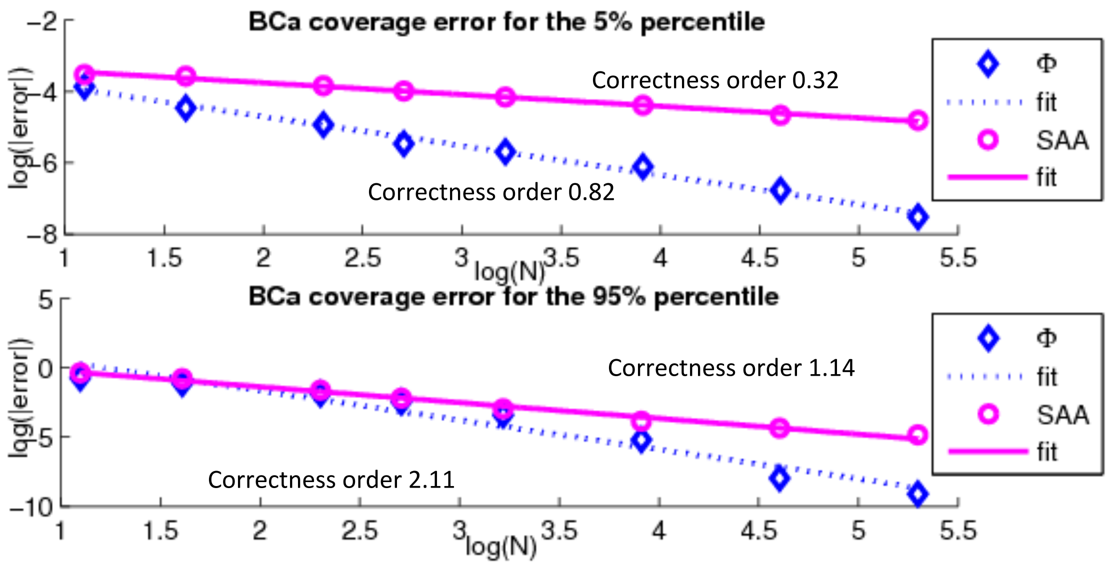
- Bootstrapping is performed based on a second sample of size M, and it works due to the fact that it is now applied at a set point x^N so finite dimensional results do apply.

Accuracy of the estimator's confidence levels

- We proved that bootstrap confidence intervals build using Φ are close to second order correct for the true optimal value $\theta = f(x^*)$:

$$P(\theta \in J_\alpha^b) = \alpha + O(N^{-1+a}), a > 0.$$

- We observed the predicted or better correctness in the numerical simulations



- bootstrapping Φ outperforms classical normal approximation method.
- We now have analytical techniques for asymptotics of confidence intervals in SP!

4.3: Q3: How do I roll the horizon in real time?

Fundamental Limitations of Off-The-Shelf Optimization

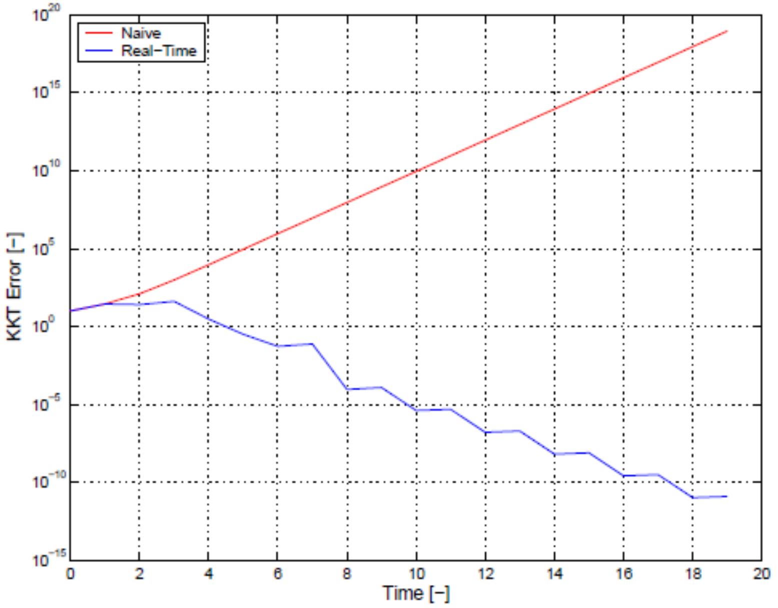
Example DO:

$$\min_{x(t)} \frac{1}{2}(x(t)-\eta(t))^2 + \frac{1}{2}x(t)^2 \cdot \eta(t)$$

Off-the-Shelf : Solve to Given Accuracy (Neglect Dynamics)

$$\epsilon^j(t) = \|\nabla_x f(x^j(t), \eta(t))\| \leq \delta_\epsilon$$

Real-Time (Z & A) : <u>One</u> SQP Iteration per step



MPC as Dynamic Generalized Equation (Z & A)

Context: Parametric NLP $\min_{x \in X} f(x,t), \text{ s.t. } c(x,t) = 0$

Time linearization of Optimality Conditions: Find $\bar{w}_t = [\bar{x}_t \ \bar{\lambda}_t]$

$$0 \in F(w^*_{t_0}, t) + \nabla_w F(w^*_{t_0}, t_0)(w - w^*_{t_0}) + \mathcal{N}_W(w)$$

KKT system for QP

$$\begin{aligned} \min \quad & \nabla_x f(x^*_{t_0}, t)^T \Delta x + \tfrac{1}{2} \Delta x^T \nabla_{xx} \mathcal{L}(w^*_{t_0}, t_0) \Delta x \\ \text{s.t.} \quad & c(x^*_{t_0}, t) + \nabla_x c(x^*_{t_0}, t_0)^T \Delta x = 0 \\ & \Delta x \geq -x^*_{t_0} \end{aligned}$$

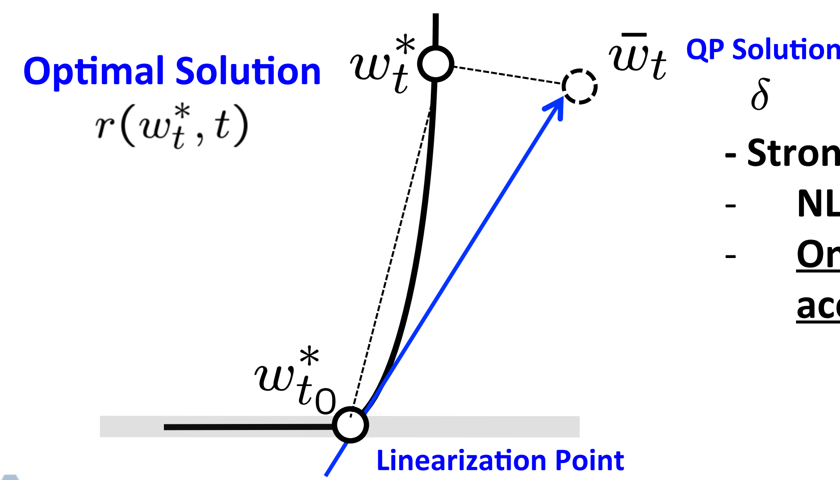
$w^*_{t_0}$

Note: Canonical Form Identical to Time-Steping for DVI

Exact Solution Satisfies:

$$\delta \in F(w^*_{t_0}, t_0) + \nabla_w F(w^*_{t_0}, t_0)(w - w^*_{t_0}) + \mathcal{N}_W(w) \qquad \delta = F(w^*_{t_0}, t_0) - F(w^*_{t_0}, t)$$

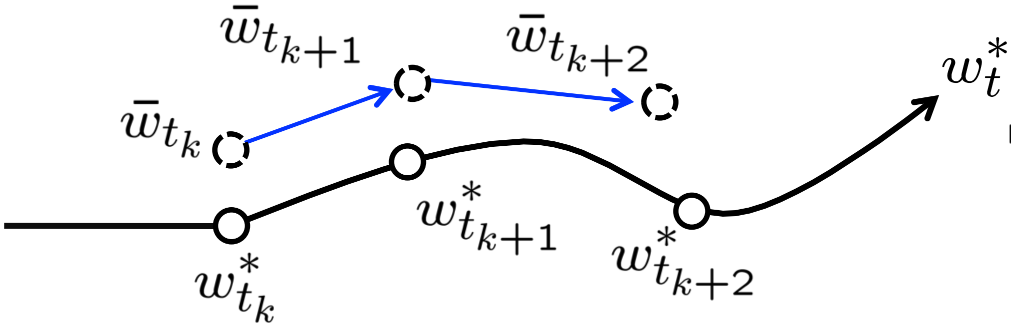
From Lipschitz Continuity of strongly regular GE: $\left\| w^*_t - \bar{w}_t \right\| \leq L \Delta t^2$



- **Strong Regularity Requires SSOC and LICQ**
  - **NLP Error is Bounded by LGE Perturbation**
  - **One QP solution from exact manifold is second-order accurate**

One-QP per step stabilizes

But for linearized DO I am never EXACTLY on the manifold: What then?



Solve off-manifold time-dependent QP

$$\begin{aligned} \min \quad & \nabla_x f(\bar{x}_{t_k}, t_{k+1})^T \Delta x + \tfrac{1}{2}\Delta x^T \nabla_{xx}\mathcal{L}(\bar{w}_{t_k}, t_k)\Delta x \\ \text{s.t.} \quad & c(\bar{x}_{t_k}, t_{k+1}) + \nabla_x c(\bar{x}_{t_k}, t_k)^T \Delta x = 0 \\ & \Delta x \geq -\bar{x}_{t_k} \end{aligned}$$

Theorem (elucidating an issue posed by Diehl et al.)

- **A: LGE is Strongly Regular at ALL $w^*_{t_k}$ e.g. NLP satisfies LICQ and SOSC everywhere**

Then: For sufficiently small Δt, we can track the manifold stably, solving 1 QP per step

$$\|\bar{w}_{t_k} - w^*_{t_k}\| \leq L_\psi \delta_r \Rightarrow \|\bar{w}_{t_{k+1}} - w^*_{t_{k+1}}\| \leq L_\psi \delta_r$$

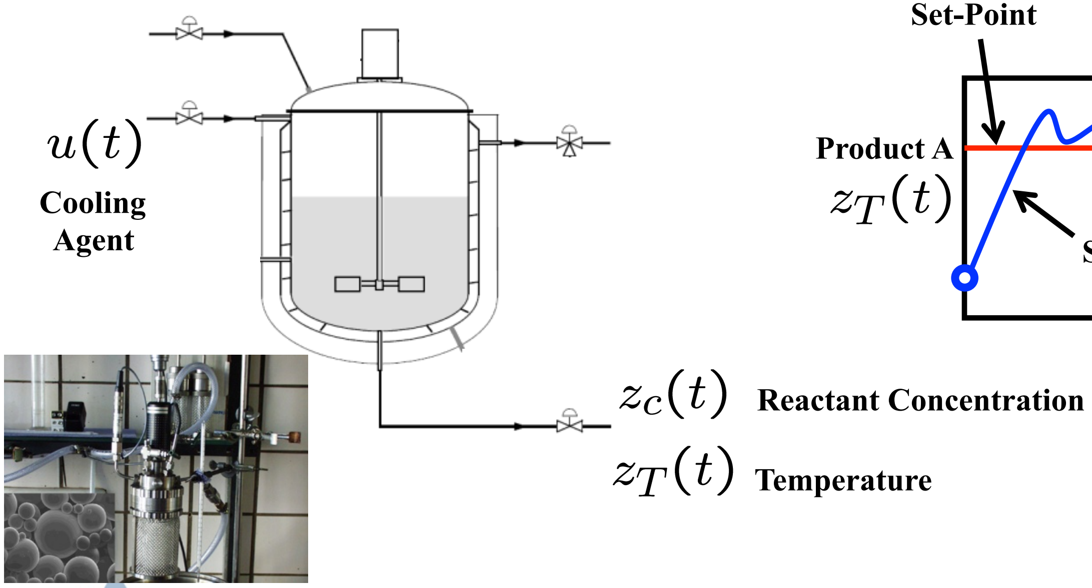
Moreover: Stability Holds Even if QP Solved to $O(\Delta t^2)$. Can use iterative methods.

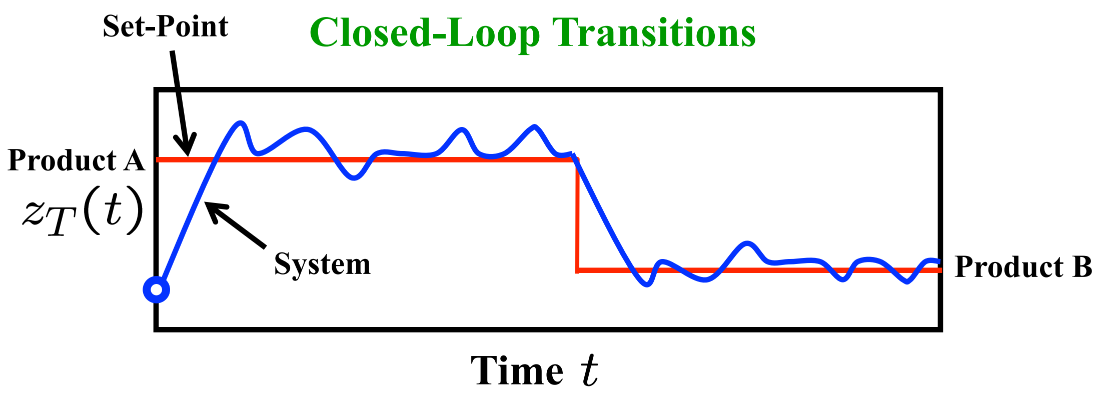
Much less effort per step and better chances for real-time performance !

Need for more features of DO solvers

- One QP per step may still be too much
- Moreover I may need also good global and fast local convergence properties as well, it is not all about asymptotics!
- Sometimes one switch regimes, the optimal point moves far away, and you still want to be able to track well. – MPC algorithm must exhibit global convergence and fast local convergence (i.e. Newton)!
- Also, power grid problems can be huge (US ~ 1 – 100 Billion Variables). Need scalable solvers.

Control of Polymerization Reactor



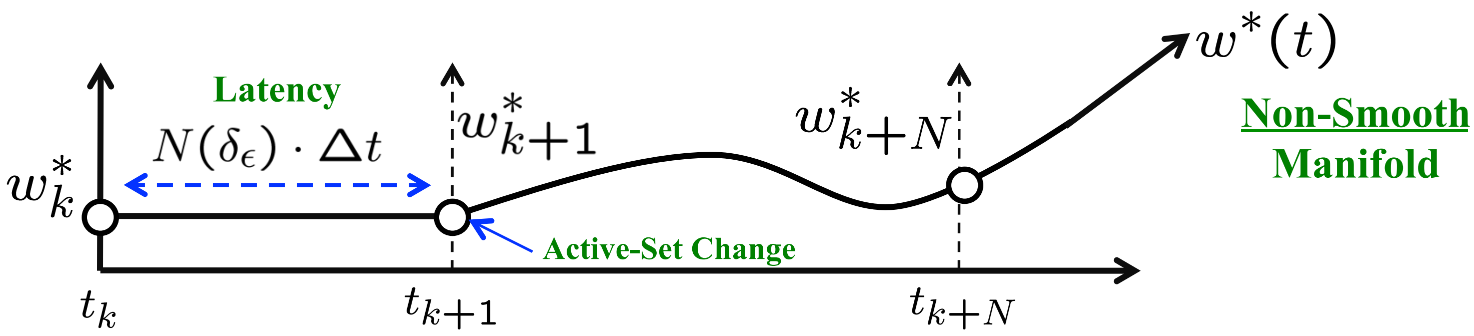


Technical Problem

$$\min_x \; f(x,t)$$
$$\text{s.t. } h(x,t) = 0, \quad (\lambda)$$
$$x \geq 0.$$

$$w^T = [x^T, \lambda^T]$$

Solution forms Time-Moving and Non-Smooth Manifold



- **Challenge is to Track Manifold Accurately (Classical Optimization) AND Stably (Latency Conscious: A good Step, Computer Fast)**

Technical Problem

- Challenge is to Track Manifold Accurately AND Stably (Get Good Step with Minimum Latency)

- This requires NLP Solvers with the Following Features:
  - A) Classical Optimization Oriented :

    1) Superlinear Convergence (Newton-Based)
    2) Scalable Step Computation (Iterative Linear Algebra)

  - B) Latency Conscious:

    3) Asymptotic Monotonicity of Minor Iterations (Makes Progress in O(N))
    4) Active-Set Detection and Warm-Start

- Existing Solvers Tend to Fail at Least One Feature
  - Interior Point: 4, and to some extent, 2,3
  - Augmented Lagrangian: 1
  - SQP: 2

Exact Differentiable Penalty Functions (EDPFs)

Consider Transformation using Squared Slacks

$$\begin{aligned} \min_{x} \ & f(x) \\ \text{s.t. } h(x) &= 0 \\ x &\geq 0 \end{aligned} \qquad\qquad \begin{aligned} \min_{x,z} \ & f(x) \\ \text{s.t. } h(x) &= 0 \\ x &= z^2 \end{aligned}$$

Equivalent To:

$$\begin{aligned} \min_{z} \ & f(z^2) \\ \text{s.t. } h(z^2) &= 0 \end{aligned} \qquad\qquad \mathcal{L}(z^2, \lambda) = f(z^2) + \lambda^T h(z^2)$$

$$\begin{aligned} \nabla_z \mathcal{L}(z^2, \lambda) &= 2 \cdot Z \cdot \left(\nabla f(z^2) + \nabla h(z^2) \lambda \right) \\ &= 2 \cdot X^{1/2} \nabla_x \mathcal{L}(x, \lambda) \end{aligned}$$

Apply DiPillo and Grippo's Penalty Function *DiPillo,Grippo, 1979, Bertsekas, 1982*

$$P(x, \lambda, \alpha, \beta) = \mathcal{L}(x, \lambda) + \frac{1}{2} \alpha c(x)^T c(x) + \boxed{2\beta \nabla_x \mathcal{L}(x, \lambda)^T X \nabla_x \mathcal{L}(x, \lambda)}$$

Solve NLP Indirectly Through EDPF Problem:

$$\min_{x,\lambda} \ P(x, \lambda, \alpha, \beta) \text{ s.t. } x \geq 0$$

Conclusions

- Complex energy systems pose an enormous number of modeling and simulation challenges.
- Computational Power will help, but it will not alone address many of the challenges.
- This research requires sustained, multi-area, integrative thinking in mathematics.
- Increases focus on area of mathematics that were not explored up to this point and creates opportunities for FUNDAMENTAL math advances:
- Examples: resampling in stoch prog for expensive scenarios; stochastic preconditioning, fast nonlinear programming.
- We expect many more such challenges, as embodied in our MACS center.