A Parallel Computing Framework for Analysis and Control of Large-scale Systems

Reza Kamyar
Advisor: Matthew Peet
Cybernetic Systems and Controls Laboratory
School for Engineering of Matter, Transport and Energy

PhD Dissertation Defense
Arizona State University, Tempe, USA

Committee members:
Matthew Peet, chair
Panagiotis Artemiadis, Spring Berman, Georgios Fainekos, Daniel Rivera
We Focus on Two Distinct Topics
Computational focus, Energy focus

**Topic 1: Application of parallel computing in controls**

- Discussing *intractable* problems in control and their real-world applications
- Formulating these problems as optimization problems with a *special structure*
- Designing parallel algorithms capable of exploiting the structure

**Topic 2: Optimal thermostat programming in a smart-grid environment**

- Determining optimal interior temperature given electricity prices & building parameters
- Benefit to residential customers: minimizing *electricity bills*
- Potential benefit to utility companies: reducing *cost of generation*
Research Goal:

Computational focus, Energy focus

Finding ways to solve *fundamentally difficult* and *large-scale* problems in control. Problems involving stability and/or control of
Research Goal:
Computational focus, Energy focus

Finding ways to solve fundamentally difficult and large-scale problems in control. Problems involving stability and/or control of

1. System of $n$ linear ODEs with $m$ uncertain parameters ($n > 100$, $m > 10$)

$$\dot{x}(t) = A(\alpha)x(t), \ \alpha \in Q \subset \mathbb{R}^m$$

Application in aerospace:
Linearized equations of symmetric flight:

$$\begin{bmatrix}
\dot{u} \\
\dot{v} \\
\dot{w} \\
\dot{p} \\
\dot{q} \\
\dot{r}
\end{bmatrix} =
\begin{bmatrix}
\frac{X_u}{m} & 0 & \frac{X_w}{m} & 0 & \frac{X_q}{m} & 0 \\
0 & \frac{Y_v}{m} & 0 & \frac{Y_p}{m} & \frac{Z_q}{m} & 0 \\
\frac{Z_u}{m} & 0 & \frac{Z_w}{m} & 0 & 0 & \frac{Y_r}{m} - U_0 \\
0 & \frac{I_{zz} L_v + I_{xz} N_v}{I_{xx} I_{zz}} & 0 & -I_{zz} L_p + I_{xz} N_p & 0 & 0 \\
0 & \frac{I_{zz} L_v + I_{xz} N_v}{I_{xx} I_{zz}} & 0 & -I_{zz} L_p + I_{xz} N_p & 0 & 0 \\
0 & \frac{M_u}{I_{yy}} & 0 & \frac{M_w}{I_{yy}} & 0 & \frac{M_q}{I_{yy}} \\
0 & \frac{I_{xz} L_v - I_{xx} N_v}{I_{xx} I_{zz}} & 0 & \frac{I_{xz} L_v - I_{xx} N_v}{I_{xx} I_{zz}} & 0 & 0 \\
0 & \frac{I_{xz} L_v - I_{xx} N_v}{I_{xx} I_{zz}} & 0 & \frac{I_{xz} L_v - I_{xx} N_v}{I_{xx} I_{zz}} & 0 & 0 \\
0 & \frac{I_{xz} L_v - I_{xx} N_v}{I_{xx} I_{zz}} & 0 & \frac{I_{xz} L_v - I_{xx} N_v}{I_{xx} I_{zz}} & 0 & 0
\end{bmatrix}
\begin{bmatrix}
u \\
w \\
p \\
q \\
r
\end{bmatrix}
\begin{array}{c}
A(X_u, X_w, X_q, Y_v, Y_p, Y_r, Z_u, Z_w, Z_q, L_v, L_p, L_r, M_u, M_w, M_q, N_v, N_p, N_r)
\end{array}

Problem: Find the uncertainty set $Q$ such that for all the aerodynamic coefficients $X_u, X_w, \cdots \in Q$, the aircraft is stable.
Research Goal:
Computational focus, Energy focus

Finding ways to solve fundamentally difficult and large-scale problems in control. Problems involving stability and/or control of

2. Systems of $n$ nonlinear ODEs ($n > 10$)

$$\dot{x}(t) = f(x(t))$$

Application in power systems:
Three-machine, nine-bus power generating system:

\[
\begin{align*}
\dot{\delta}_1(t) &= \omega_1(t) \\
\dot{\delta}_2(t) &= \omega_2(t) \\
\dot{\delta}_3(t) &= \omega_3(t) \\
\dot{\omega}_1(t) &= \frac{1}{m}(d_1\omega_1(t) + P_{m_1} - P_{e_1}(\delta_1(t), \delta_2(t), \delta_3(t))) \\
\dot{\omega}_2(t) &= \frac{1}{m}(d_2\omega_2(t) + P_{m_2} - P_{e_2}(\delta_1(t), \delta_2(t), \delta_3(t))) \\
\dot{\omega}_3(t) &= \frac{1}{m}(d_3\omega_3(t) + P_{m_3} - P_{e_3}(\delta_1(t), \delta_2(t), \delta_3(t))) \\
\end{align*}
\]

Problem: Find the set of initial phase angles $\delta_i(0)$ and frequencies $\omega_i(0)$ such that $\delta_i(t)$ and $\omega_i(t)$ converge to a stable equilibrium.
Research Goal:
Computational focus, Energy focus

Finding ways to solve **fundamentally difficult** and **large-scale** problems in control. Problems involving stability and/or control of

3. PDEs with uncertain parameters

\[
\frac{\partial}{\partial t} u(x, t) = \alpha_0 u(x, t) + \sum_{i=1}^{m} \alpha_i \frac{\partial^i}{\partial x^i} u(x, t), \quad \alpha \in Q
\]

**Application in ecology:**

Modelling of population density in a 2D landscape:

\[
u_t(x, y, t) = \alpha (u_{xx}(x, y, t) + u_{yy}(x, y, t)) + \beta (u_x(x, y, t) + u_y(x, y, t)) + \gamma u(x, y, t)
\]

- **population diffusion**
- **population drift**
- **population growth**
Research Goal:
Computational focus, Energy focus

Finding ways to solve fundamentally difficult and large-scale problems in control. Problems involving stability and/or control of

4. Systems of linear ODEs with time-delay

\[
\dot{x}(t) = \sum_{i=1}^{m} A_i x(t - \tau_i)
\]

Application in immunology:
A linearized model for immune system response:

\[
\begin{bmatrix}
\dot{T}(t) \\
\dot{T}^*(t) \\
\dot{V}(t) \\
\dot{E}(t)
\end{bmatrix} =
\begin{bmatrix}
-d - kV_{ss} & 0 & -kT_{ss} & 0 \\
-kV_{ss} & -\delta - d_x E_{ss} & kT_{ss} - d_x T^*_{ss} & 0 \\
0 & N\delta & -c & 0 \\
0 & 0 & -d_E & 0
\end{bmatrix}
\begin{bmatrix}
T(t) \\
T^*(t) \\
V(t) \\
E(t)
\end{bmatrix} +
\begin{bmatrix}
0 \\
0 \\
pT^*(t - \tau)
\end{bmatrix}
\]

\(T^*(t - \tau)\) allows for a time delay between the moment of infection and the recognition of the infected cells.
What Are The Computational Challenges?

- **NP-hardness**: Most likely there exists no algorithm which can find exact solutions to these problems in polynomial-time.
  
  e.g., Stability analysis of $\dot{x}(t) = A(\alpha)x(t)$ using the converse Lyapunov theory:
  
  \[ P(\alpha) > 0, \quad A^T(\alpha)P(\alpha) + P(\alpha)A(\alpha) < 0 \]
  
  The question of feasibility of parameter-dependent Lyapunov inequalities is NP-hard.

- **Dimension**: The required memory for the existing algorithms scales exponentially with the dimension of the problem and accuracy of the solutions.

  Even a “rough” discretization of a 2D PDE can create hundreds of states!

  e.g., current algebraic geometry techniques (SOS) require 1 TB of memory to verify stability of a nonlinear system with 10 states.
The SOS method defines a sequence of convex optimization problems (SOS programs) whose solutions converge to a solution of the intractable problem.

SOS programs admit polynomial-time solutions - complexity $\sim n^{O(d)}$.

$n$ : state-space dimension, $d$ : degree of the Lyapunov function

Example: Robust stability

System $\dot{x}(t) = A(\alpha)x(t)$, $\alpha \in [0, 1]$ is stable if and only if $\exists P(\alpha)$:

$P(\alpha) > 0 \text{ and } -A(\alpha)^T P(\alpha) - P(\alpha)A(\alpha) > 0 \text{ for all } \alpha \in [0, 1]$

Instead one can solve

$P(\alpha) = S_0(\alpha) + \alpha(1 - \alpha)S_1(\alpha)$
$-A(\alpha)^T P(\alpha) - P(\alpha)A(\alpha) = S_2(\alpha) + \alpha(1 - \alpha)S_3(\alpha)$

$S_0, S_1, S_2, S_3$ are SOS polynomials, i.e., $S_i(\alpha) = \sum_i G_i(\alpha)^2$. 
Is Polynomial-Time Good Enough?

- Polynomial-time algorithms have been perceived as the gold standard for what the solution to a control problem should look like.

- However, polynomial-time algorithms are **NOT** always practical!
  
  e.g., computing a Lyapunov function for a 10-state nonlinear system by the SOS algorithm requires 116 DAYS!

- A polynomial-time algorithm is “good” when the ratio of its complexity to the computing power of current computers is reasonably low (technology-dependent).

- The per-core speed of commercial CPUs **has saturated**, while majority of controls algorithms and software can use only a single core.

Moor’s law is manifesting in the form of multi-core CPUs.

Single-core processor speed has saturated
The real problem with computation in control is not the availability of resources, but rather the lack of algorithms capable of efficiently utilizing those resources.

We look for algorithms capable of using those computational resources which have the fastest growth in speed: *cluster-computing, supercomputing*.

Surprisingly there has been little study on the use of parallel computation for control!

No surprise! The mathematical machinery for analysis and control is based on two inherently sequential algorithms: Linear Programming (LP) & Semi-Definite Programming (SDP).

How then parallel computing can help? Is it sufficient to focus on parallelizing SDPs with “special structure”? 

- 2009: Cray XT5-HE goes live
- 2009: First world-class GPU-powered supercomputer
- 2008: Petaflop barrier broken
- 2003: Human genome mapped
- 2005: Millennium run simulation
- 2009: First world-class GPU-powered supercomputer
- 2006: Petaflop barrier broken
- 1999: ASCI blue pacific live
- 1993: CM-5/1024 supercomputer
- 1984: M-13 supercomputer
- 1976: Cray 1 goes live
- 1960: Univac LARC goes live
- 2014: Tianhe 2 (Milky-way) goes live
- 2012: Titan - Cray XK7 goes live
- 2009: Cray XT5-HE goes live
- 2008: Petaflop barrier broken
Semi-Definite Programming:

Optimization over the cone of positive semi-definite matrices

\[
\begin{align*}
\min_{y,Z} & \quad a^T y \\
\text{subject to} & \quad \sum_{i=1}^{K} y_i B_i - C = Z \\
& \quad Z \geq 0
\end{align*}
\]

- Decision variables: \( y \in \mathbb{R}^K, \ Z \in \mathbb{S}^n \) (symmetric matrix)
- SDP elements (given): \( B_i, C \in \mathbb{S}^n \) and \( a \in \mathbb{R}^K \)
- SDPs can be solved efficiently using **interior-point algorithms**.
Interior-point algorithms solve SDPs in TWO steps:

1. Reducing the SDP to a sequence of optimization programs with only equality constraints

\[
\begin{align*}
\text{Dual SDP} & \\
\min_{y,Z} & a^T y \\
\text{subject to} & \sum_{i=1}^{K} B_i y_i - C = Z \\
& Z \geq 0
\end{align*}
\]

\[
\begin{align*}
\text{Approximation using barrier function} & \\
\min_{y,Z} & a^T y - \mu \log(\det(Z)) \\
\text{subject to} & \sum_{i=1}^{K} B_i y_i - C = Z
\end{align*}
\]

2. Applying a descent algorithm, e.g., Newton’s algorithm, to solve the equality constrained problems

\[
\begin{align*}
y^{k+1} &= y^k + t \Delta y^k \\
Z^{k+1} &= Z^k + t \Delta Z^k \\
X^{k+1} &= X^k + t \Delta X^k
\end{align*}
\]

- $\Delta y^k, \Delta Z^k, \Delta X^k$ are the step directions.
- Calculating the step directions is the most computationally expensive part.
What If The SDP Elements Have a Special Structure?

**Block-diagonality** is preserved through iterations

**Assumption:** The SDP elements $B_i$ and $C$ are **block-diagonal** matrices.

Primal step:  
\[
\Delta X^k = -X^k + Z^{k-1} \left( - \sum_{i=1}^{K} B_i y_i^k + Z^k + C \right) \sum_{i=1}^{K} B_i \Delta y_i^k X^k
\]

**Observations:**

- If $X^0$ and $Z^0$ are **block-diagonal**, then $\Delta X^k$ and $\Delta Z^k$ are **block-diagonal** $\forall k$.

- Then $X^k$ and $Z^k$ are also **block-diagonal** for all $k$ because
\[
X^{k+1} = X^k + t \Delta X^k \quad Z^{k+1} = Z^k + t \Delta Z^k.
\]

- We decentralize the computation of step directions $\Delta X, \Delta Z$ by assigning each block to a processor.
Can Stability/Control Problems Reduce To Block-diagonal SDPs?

Alternatives to SOS algorithm

- Unfortunately SOS algorithm does NOT yield block-diagonal SDPs.

**Example:** Is \( f(x) = 4x^4 + 4x^3y - 7x^2y^2 - 2xy^3 + 10y^4 \geq 0? \)

If \( \exists M := \begin{bmatrix} M_1 & M_2 & M_3 \\ M_2 & M_3 & M_4 \\ M_3 & M_4 & M_5 \end{bmatrix} \geq 0 \) such that \( f = \begin{bmatrix} x^2 \\ xy \\ y^2 \end{bmatrix}^T M \begin{bmatrix} x^2 \\ xy \\ y^2 \end{bmatrix} \Rightarrow f \) is SOS

- We identified alternatives to SOS - Theorems which reformulate polynomial positivity (e.g., \( V > 0, \dot{V} < 0 \)) as feasibility of block-diagonal SDPs and LPs:
  - **Polya’s Theorem** (positivity over the standard unit simplex)
  - **Bernstein’s Theorem** (positivity over simplex)
  - **Handelman’s Theorem** (positivity over polytopes)
Polya’s Theorem
A test for non-negativity over the standard simplex

Example: Is \( p(x, y) = 2x^4 - 0.11x + y^3 \geq 0? \)

Step 1) **Homogenizing** \( p(x, y) \):

\[
\tilde{p}(x, y) = 2x^4 - 0.11x(x + y)^3 + y^3(x + y)
\]

Step 2) **Polya’s iterations on** \( \tilde{p}(x) \):

Multiply \( \tilde{p}(x, y) \) by \( (x + y) \) until all the coefficients are positive.

**Iteration #1:**

\[
(x + y)\tilde{p}(x, y) = 1.89x^5 + 1.56x^4y - 0.66x^3y^2 + 0.56x^2y^3 + 1.89xy^4 + y^5
\]

**Iteration #2:**

\[
(x + y)^2\tilde{p}(x, y) = 1.89x^6 + 3.45x^5y + 0.9x^4y^2 - 0.1x^3y^3 + 2.45x^2y^4 + 2.89xy^5 + y^6
\]

**Iteration #3:**

\[
(x + y)^3\tilde{p}(x, y) = 1.89x^7 + 5.34x^6y + 4.35x^5y^2 + 0.8x^4y^3 + 2.35x^3y^4 + 5.34x^2y^5 + 3.89xy^6 + y^7
\]
Applying Polya’s Theorem To Robust Stability Problem $\dot{x}(t) = A(\alpha)x(t)$

Enforcing $P(\alpha) > 0$ over the unit simplex

- Recall that $\dot{x}(t) = A(\alpha)x(t)$, $\alpha \in \Delta$ is stable if and only if $\exists P(\alpha)$:

\[P(\alpha) > 0 \text{ and } -A(\alpha)^T P(\alpha) - P(\alpha)A(\alpha) > 0 \text{ for all } \alpha \in \Delta\]

- Let $P(\alpha)$ be of the form

\[P(\alpha) = P_1 \alpha_1^2 + P_2 \alpha_1 \alpha_2 + P_3 \alpha_2^2 \quad (P_i \in \mathbb{S}^m \text{ are unknown})\]

Then by calculating the coefficients of $(\alpha_1 + \alpha_2)P(\alpha)$ as

\[(\alpha_1 + \alpha_2)P(\alpha) = P_1 \alpha_1^3 + (P_1 + P_2)\alpha_1^2 \alpha_2 + (P_2 + P_3)\alpha_1 \alpha_2^2 + P_3 \alpha_2^3,\]

positive definiteness of $P(\alpha)$ is guaranteed if $\exists P_1, P_2, P_3$ such that

\[
\begin{bmatrix}
    P_1 & 0 & 0 & 0 \\
    0 & P_1 + P_2 & 0 & 0 \\
    0 & 0 & P_2 + P_3 & 0 \\
    0 & 0 & 0 & P_3
\end{bmatrix} > 0.
\]

- Similarly, we can apply Polya’s theorem to $A(\alpha)^T P(\alpha) + P(\alpha)A(\alpha) \leq 0$. 
The Resulting SDPs Are Large!
The required memory for setup and solving the SDPs is beyond desktop/shared-memory computers.

### Memory required for storing the SDP

<table>
<thead>
<tr>
<th>d₁, d₂, n</th>
<th>Memory (Gbytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 1, 1</td>
<td>10³</td>
</tr>
<tr>
<td>1, 10, 1</td>
<td>10⁴</td>
</tr>
<tr>
<td>1, 10, 10</td>
<td>10⁵</td>
</tr>
<tr>
<td>10, 10, 1</td>
<td>10⁶</td>
</tr>
<tr>
<td>10, 10, 10</td>
<td>10⁷</td>
</tr>
</tbody>
</table>

### Number of monomials in \((\sum \alpha_i)^d A^T P + PA\)

<table>
<thead>
<tr>
<th>d₁, d₂</th>
<th>Number of H coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10⁰</td>
</tr>
<tr>
<td>2</td>
<td>10¹</td>
</tr>
<tr>
<td>4</td>
<td>10²</td>
</tr>
<tr>
<td>6</td>
<td>10³</td>
</tr>
<tr>
<td>8</td>
<td>10⁴</td>
</tr>
<tr>
<td>10</td>
<td>10⁵</td>
</tr>
</tbody>
</table>

- **Number of SDP variables:** \(\sim n^2 l^{d_p}\)
- **Number of SDP constraints:** \(\sim nl^{d_p + d_a + d}\)

Recall:
- \(n\) : state-space dimension
- \(l\) : number of uncertain parameters
- \(d\) : number of Polya’s iterations
- \(d_p, d_a\) : degrees of \(P(\alpha)\) and \(A(\alpha)\)
We Designed And Implemented TWO Parallel Algorithms: Setup & Solver

- **Parallel setup algorithm:**
  1. Distributes monomials of $P(\alpha)$ and $A(\alpha)$ among processors, evenly.
  2. Each processor applies Polya's iteration to its monomials:
     
     $$Q_1 = \left( \sum_i \alpha_i \right) P(\alpha) \quad \text{and} \quad Q_2 = \left( \sum_i \alpha_i \right) A^T(\alpha) P(\alpha) + P(\alpha) A(\alpha)$$
  3. Redistributes the monomials of $Q_1$ and $Q_2$ among processors, evenly (Communication)

- **Parallel SDP solver:**
  - Recall that the step directions $\Delta X$ and $\Delta Z$ are **block-diagonal**.
    
    $$\Delta X = \text{diag}\{\Delta X_1, \cdots, \Delta X_M\} \quad \Delta Z = \text{diag}\{\Delta Z_1, \cdots, \Delta Z_M\}$$
  - Having $N$ Processors, each processor computes at least $\text{floor} \left( \frac{M}{N} \right)$ blocks and updates
    
    $$X_i = X_i + t \Delta X_i, \quad Z_i = Z_i + t \Delta Z_i \quad \text{for } i = 1, \cdots, \text{floor} \left( \frac{M}{N} \right)$$

\[ \text{Setup} \quad \text{Blockdiagonal SDP} \quad \text{Solver} \]

\[ \begin{bmatrix} [1] & [1] & [1] & [1] \end{bmatrix} \geq 0 \]

V(x) = $x^T(\Sigma P_i \alpha^h)x$
Per-Core Complexity of The Algorithms Is $O(n^7)$

**Assumptions:**

1. Having sufficiently large number of processors ($\geq$ number of blocks)
2. Number of states, $n \gg$ number of uncertain parameters, $l$

Then

1. Our algorithms solve robust stability problem

$$A^T(\alpha)P(\alpha) + P(\alpha)A(\alpha) < 0 \quad \alpha \in \Delta^l$$

with the **same per-core cost** $O(n^7)$ as required for solving the stability problem

$$A^TP + PA < 0.$$ 

2. Increasing accuracy (performing Polya’s iterations) does **NOT** add any per-core computation and communication.
Theoretically Our Algorithms Achieve Linear Speed-up

**Speed-up:** The ratio of the execution time using one core to the execution time using $N \geq 1$ cores.

- Potential speed-up is calculated as

$$SP = \frac{N}{D + NC}$$

$D$: decentralized computation  
$C$: centralized computation

- For sufficiently large number of processors, we have shown

$$\lim_{n \to \infty} D(n) = 1 \text{ and } \lim_{n \to \infty} C(n) = 0.$$ 

$$\Rightarrow \lim_{n \to \infty} SP(n) = \lim_{n \to \infty} \frac{N}{D(n) + NC(n)} = N \quad (\text{Linear speed-up})$$
Linear Experimental Speed-up of Our Parallel SDP Solver

Our parallel SDP solver outperforms the general purpose parallel SDP solver, SDPARA, in terms of speed-up.
Linear Experimental Speed-up of Our Parallel Set-up Algorithm

Computation time of the set-up algorithm scales log-linearly with number of cores.

Executed on IBM’s Blue-Gene supercomputer at Argonne National Laboratory.
How Big A Problem Can The Algorithms Solve?

The proposed decentralized algorithms can solve problems with 100+ state-space dimension

Executed on one and nine nodes of Karlin cluster computer with 24GByte/node RAM
Conservatism Reduces As Degree of $P$ & No. of Polya’s Iterations Increase

Error of algorithm’s approximation for the largest $r$ such that $\dot{x}(t) = A(\alpha)x(t)$ is stable $\forall \alpha \in \Delta_r$

$$\Delta_r := \{ \alpha \in \mathbb{R}^l : \sum_{i=1}^{l} \alpha_i = r, \alpha_i \geq 0 \}$$

- SOS algorithm runs out of memory for $d_p \geq 2$
Summary of Contributions

Computational focus, Energy focus

- Designed a parallel SDP solver for block-diagonal SDPs (ACC 2012)

- Designed a parallel setup algorithm to apply Polya’s theorem to robust stability over the simplex (TAC 2013)

\[ \Delta^n := \{ x \in \mathbb{R}^n : \sum_{i=1}^{n} x_i = 1, x_i \geq 0 \} \]

- Extending Polya’s theorem for robust stability over hypercubes (CDC 2012)

\[ \Phi_{r_i}^n := \{ x \in \mathbb{R}^n : |x_i| \leq r_i, i = 1, \ldots, n \} \]

- Extension to nonlinear local stability/region of attraction estimation inside hypercubes (CDC 2013)

- Extension to stability over arbitrary polytopes using Handelman’s theorem (CDC 2014)

\[ \Gamma^K := \{ x \in \mathbb{R}^n : w_i^T x + u_i \geq 0, i = 1, \ldots, K \} \]

- A survey on alternatives to SOS (Polya, Handelman, Bernstein, Blossoms, \ldots) (DCDS 2015)
Generalizing our parallel set-up algorithm to apply Polya’s theorem to arbitrary parameter-dependent inequalities of the form:

\[
\sum_{i=1}^{N} \left( A_i(\alpha)X(\alpha)B_i(\alpha) + B_i^T(\alpha)X(\alpha)A_i^T(\alpha) + R_i(\alpha) \right) < -\gamma I \quad \text{for all } \alpha \in Q,
\]

Parallel algorithm for Optimal Control:

\[
J^* := \min_{u_k \in U} \sum_{k=0}^{\infty} \beta^k g(x_k, u_k)
\]

subject to \( x_{k+1} = f(x_k, u_k) \) for \( k = 1, 2, 3, \ldots \)

\( x_k \in X, \ x_0 = z \) for \( k = 1, 2, 3, \ldots \)

By searching for polynomial value functions \( V \) which satisfy Bellman’s formula:

\[
V(z) = \inf_{v \in U} \{ g(z, v) + \beta V(f(z, v)) \} \quad \forall z \in X.
\]

Then \( V(z) = J^* \).
Transition To Our Second Topic: Optimal Thermostat Programming

Computational focus, Energy focus

- Computing optimal response of residential customers to electricity prices
  - Quantifying the benefits of using energy storage and solar by the customers
  - Minimizing the electricity bill by designing optimal thermostats for HVAC systems

- Economical implications for power companies
  - Optimal electricity pricing for minimizing cost of generating electricity
  - Optimal unit scheduling

![Diagram](Image)
A simplified model for cost of generating electricity is a combination of

1. **Cost of fuel** required to generate the total energy (kWh) consumed by users

   A common model is:
   \[
   \text{cost of fuel} = a \int q(t) dt
   \]
   
   - \( q(t) \) (kW): power consumed by users
   - \( a \) ($/kWh): cost of fuel required to produce the next kWh

2. **Cost of building & maintaining generators** to accommodate for the maximum total power (kW) consumed by users

   A simple model can be:
   \[
   \text{Cost of building & maintaining generators} = b \sup_{t \in \text{on-peak}} q(t)
   \]
   
   - \( b \) ($/kW): cost of installing the next kW of generating capacity
Most power companies use **flat** or **Time-of-Use (ToU)** pricing.

**Flat pricing:** Charges are independent of when energy is used.

\[
\int q_1(t) dt \times \frac{\text{price}}{\text{kWh}} = \int q_2(t) dt \times \frac{\text{price}}{\text{kWh}}
\]

Electricity bills independent of \(q_{1\text{max}}\) & \(q_{2\text{max}}\)

**ToU pricing:** Does not explicitly charge for max power used.

\[
\text{Elect. Bill} = p_{\text{off}} \int_{\text{off-peak}} q(t) dt + p_{\text{on}} \int_{\text{on-peak}} q(t) dt
\]

Large peak does not necessarily result in a higher monthly bill!
Current Pricing Strategies Are Problematic For Power Companies

- **Fact 1:** The ratio of maximum power used per year to average power used per year is setting records in the US!
  
  - Partially due to increasing integration of renewables, e.g., solar.

  ![Graph showing peak to average demand from 1995 to 2010 for California and New England with real data and trendlines.](image)

- **Fact 2:** Integration of renewables does NOT affect maximum power consumption, but reduces the total power sold by power companies ⇒ revenue decreases

- **Consequence:** Power companies won’t have enough revenue to supply for electricity without raising the prices
Demand Charge: A Solution To The Revenue Problem

- **Demand charge**: A monthly charge proportional to the maximum power consumed by the user during the on-peak hours of a month

- A combination of off-peak, on-peak and demand charges can differentiate between “good” and “bad” user behavior

Electricity Bill = \[ p_{\text{off}} \int_{t \in \text{off-peak}} q(t) \, dt + p_{\text{on}} \int_{t \in \text{on-peak}} q(t) \, dt + p_{d} \sup_{t \in \text{on-peak}} q(t) \]
How Can Power Companies Optimize Their Prices?

Power companies can solve the following optimization problem:

- **Objective:** minimize the cost of generating electricity

\[
\min_{p_{on}, p_{off}, p_d} \left( \int_{t=0}^{t=24} \left( a g(t)^2 + b g(t) \right) dt + c \sup_{t \in \text{on-peak period}} g(t) \right)
\]

- **Variables:**
  - \( g(t) \): power (kW) generated at time \( t \)
  - \( a, b \) ($/kWh): fuel cost coefficients
  - \( c \) ($/kW): cost of installing the next kW of production capacity

- **Constraint:**
  - Equality of generation, \( g(t) \), and consumed power, \( q_{user}(t) \):

\[
g(t) = q_{user}(t, p_{off}, p_{on}, p_d) \quad \forall t
\]

- **Variables:** on-peak, off-peak and demand prices: \( p_{on}, p_{off}, p_d \)
To optimize electricity prices, we need a **model** for users' power consumption

Model should predict how much electricity would a **rational** user consume, given the prices

**Question:** How can a rational user reduce his electricity bill?

One way is to reduce HVAC load by using **Energy storage**

1. Energy storage in residential **batteries** allows users to shift peaks from high-demand hours to other hours

2. Using walls/floors as **thermal energy storage**: A free alternative to batteries
Precooling exploits thermal energy storage in walls to shift loads:

- Cool down walls/floors when electricity is cheap

- Cold walls will reduce the load on HVAC during on-peak hours - thus reducing the electricity bill
How Do Thermostat Settings Affect Energy Consumption?

Power consumed by user is a combination of heat **loss to outside** and heat **given to/taken from interior walls**

\[
q_{\text{user}}(t) = q_{\text{loss}}(t) + q_{\text{wall}}(t) \quad \forall k
\]

- Heat loss \(q_{\text{loss}}(t)\) is modeled by a **linear heat sink** and can be controlled by interior temperature \(T_{\text{in}}\):

\[
q_{\text{loss}}(t) = \frac{T_{\text{out}}(t) - T_{\text{in}}(t)}{R_w}
\]

\(T_{\text{out}}\): Outside temperature \quad R_w: \text{thermal resistance}

- Heat thru walls \(q_{\text{wall}}(k)\) is modeled by the **Heat equation** (PDE):

\[
\frac{\partial T_w(t, x)}{\partial t} = \alpha \frac{\partial^2 T_w(t, x)}{\partial x^2}
\]

\[
q_{\text{wall}}(k) = 2C_w \frac{\partial T_w}{\partial x}(t, 0)
\]
User can solve a discrete-time thermostat programming problem with

- **Objective:** minimize the electricity bill

\[
\min_{T_{\text{in}}(k)} \left( 30 p_{\text{off}} \sum_{k \in I_{\text{off}}} q_{\text{user}}(k) + 30 p_{\text{on}} \sum_{k \in I_{\text{on}}} q_{\text{user}}(k) + p_d \sup_{k \in I_{\text{on}}} q_{\text{user}}(k) \right)
\]

- **Constraints:**
  1. Interior temperature with a certain bound:
     \[
     T_{\text{min}} \leq T_{\text{in}}(k) \leq T_{\text{max}} \quad \forall k
     \]
  2. Energy conservation:
     \[
     q_{\text{user}}(k) = q_{\text{loss}}(T_{\text{in}}(k), T_e(k)) + q_{\text{wall}}(T_w(x, k)) \quad \forall k
     \]
  3. Discretized heat dynamics:
     \[
     T_w(k + 1) = A T_w(k) + B T_{\text{in}}(k)
     \]

- **Variables:** Interior temperature \(T_{\text{in}}(k)\) over time
A Reformulation of User’s Problem Can Be Solved By Dynamic Programming

- We first reformulate the user’s problem

\[
\begin{align*}
\min_{T_{in}(k)} & \quad 30 p_{off} \sum_{k \in I_{off}} q(k) + 30 p_{on} \sum_{k \in I_{on}} q(k) + p_d \sup_{k \in I_{on}} q(k) \\
\text{subject to} & \quad q(k) = q_{loss}(T_{in}, T_{out}) + q_w(T_w) \quad \forall k \\
& \quad T_w(k + 1) = f(T_w(k), T_{in}) \quad \forall k \\
& \quad T_{\min} \leq T_{in}(k) \leq T_{\max} \quad \forall k
\end{align*}
\]

as

\[
\begin{align*}
\min_{T_{in}(k), \gamma \in \mathbb{R}} & \quad 30 p_{off} \sum_{k \in I_{off}} q(k) + 30 p_{on} \sum_{k \in I_{on}} q(k) + p_d \gamma \\
\text{subject to} & \quad q(k) \leq \gamma \quad \forall k \in I_{on} \\
& \quad q(k) = q_{loss}(T_{in}, T_{out}) + q_w(T_w) \quad \forall k \\
& \quad T_w(k + 1) = f(T_w(k), T_{in}) \quad \forall k \\
& \quad T_{\min} \leq T_{in}(k) \leq T_{\max} \quad \forall k
\end{align*}
\]

- For fixed \( \gamma \), the reformulated problem can be solved by Dynamic Programming.

- \( \gamma \) is a scalar, so we use \textbf{bisection} over \( \gamma \).
Our Algorithm Can Reduce electricity Bills By Up To 25% (average 9.2%)

User’s consumption and interior temperature using prices from Arizona Public Service (APS)

![Graph showing power consumption and interior temperature over time with different algorithms: Our Algorithm, Precooling, Constant, GPOPS. The graph indicates that Our Algorithm results in lower power consumption and more stable interior temperature compared to the other algorithms.]

<table>
<thead>
<tr>
<th>Temperature setting</th>
<th>Our algorithm</th>
<th>GPOPS</th>
<th>Pre-cooling</th>
<th>Constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monthly bill</td>
<td>$365.8$</td>
<td>$370.3$</td>
<td>$392.3$</td>
<td>$394.2$</td>
</tr>
</tbody>
</table>
Increasing $\frac{P_d}{P_{off}}$ Helps Reducing Maximum Consumption during on-peak

- Weight of demand price relative to on-peak & off-peak prices affects maximum consumption during on-peak hours
Summary of Contributions on Thermostat Programming/Electricity Pricing

- Defined a new model for optimal behavior of a customer who minimizes his electricity bill based on given prices (ACC 2015)
  - Including thermal energy storage using the heat equation
  - Including monthly demand charges

- Used our model to define a framework for optimization of electricity prices for rational users (submitted to IEEE Transactions on Power Systems)
  - Minimizing the cost to the power company
  - Considering integration of solar power

- A Multi-objective Approach To Optimal Battery Storage In The Presence of Demand Charges (Under preparation for IBO Conference, 2016)
Our Ongoing Research On Storage: Optimal Programming of Batteries

- Incorporating **batteries**, such as Tesla’s Powerwall & Tesla’s Powerpack in our user’s models and utility model

- Including **stochasticity** due to weather temperature and solar radiation in our customer’s model - minimizing $E_\omega \{ \sup_u g(t, u, \omega) \}$. 
Our Ongoing Research: Benefits of Battery Storage To Power Companies

- Optimal **battery storage & unit scheduling** to minimize generation costs
  
  ▶ **Fuel cost** of various types of generating units
  
  ▶ **Unit commitment**: Cost for bringing each generating unit online
  
  ▶ **Arbitrage**: Selling/buying from electricity spot market
  
  ▶ **Spinning reserve** and **frequency regulation** costs

![Graph showing cost comparison of different energy sources](image.png)
Conclusions & Achievements

Computational focus, Energy focus

**Topic 1: Application of parallel computing in controls**

- Developed a parallel optimization framework using Polya’s & Handelman’s theorems for robust stability analysis over various geometries.

- Our algorithms achieve **near-linear** theoretical and experimental speed-up.

- Our algorithms enable robust stability analysis of systems **3 times** larger than ANY other algorithm (100+ states, tens of parameters).

**Topic 2: Optimal thermostat programming in an smart-grid environment**

- Developed a model for rational customers who **exploit storage** to minimize their monthly bill

- Designed an algorithm for **optimal thermostat programming**, capable of reducing monthly bills by up to 25%

- Proposed **optimal combinations** of on-peak, off-peak, demand prices which reduce both peak consumption and generation costs

- Quantified the effects of **solar integration** on customers behavior and generation costs
Acknowledgments

- **My Adviser**
  Matthew Peet

- **My committee**
  Panagiotis Artemiadis
  Spring Berman
  Georgios Fainekos
  Daniel Rivera

- **National Science Foundation** for funding our research: grant # XAS0430 and CMMI-1100376.

- **Salt River Project** Arizona’s utility company for funding our research and providing data and consultation

- **My parents**, Jafar Kamyar and Ashraf Foroutani.

- **My labmates and friends**
  Aditya Gahlawat
  Chaitanya Murti
  Evgeny Meyer
  Hesameddin Mohammadi
  Shahrzad Talebianmoghaddam
  Karthick Elamvazuthi
  Mohammad Hekmatnejad

- **The audience**