# System Level Synthesis via Dynamic Programming


Shih-Hao Tseng, (pronounced as "She-How Zen")

joint work with Carmen Amo Alonso and SooJean Han


December 15, 2020


Department of Computing and Mathematical Sciences,
California Institute of Technology

- Controller synthesis for a state-feedback system:

$$x[t+1] = Ax[t] + Bu[t] + w[t],$$

where $x$ is the state, $u$ the control, and $w$ the noise. In the frequency domain, the system dynamics writes as

$$z\mathbf{x} = A\mathbf{x} + B\mathbf{u} + \mathbf{w}.$$

- Controller synthesis for a state-feedback system:

$$x[t+1] = Ax[t] + Bu[t] + w[t],$$

where $x$ is the state, $u$ the control, and $w$ the noise. In the frequency domain, the system dynamics writes as

$$z\mathbf{x} = A\mathbf{x} + B\mathbf{u} + \mathbf{w}.$$

- Goal: Finding a linear controller $\mathbf{u} = \mathbf{K}\mathbf{x}$.

# Controller Synthesis for Large-Scale Systems

- Controller synthesis for a state-feedback system:

$$x[t+1] = Ax[t] + Bu[t] + w[t],$$

where $x$ is the state, $u$ the control, and $w$ the noise. In the frequency domain, the system dynamics writes as

$$z\mathbf{x} = A\mathbf{x} + B\mathbf{u} + \mathbf{w}.$$

- Goal: Finding a linear controller $\mathbf{u} = \mathbf{K}\mathbf{x}$.
- Issue: How do we incorporate system level objective and constraints?

# System Level Synthesis (SLS)

- Considering the system response $\{\mathbf{\Phi_x}, \mathbf{\Phi_u}\}$, SLS allows easy incorporation of system level objective $g$ and constraints $\mathcal{S}$.

$$
\begin{aligned}
\min \ & \boxed{g(\mathbf{\Phi_x}, \mathbf{\Phi_u})} \\
\text{s.t.} \ & \begin{bmatrix} zI - A & -B \end{bmatrix} \begin{bmatrix} \mathbf{\Phi_x} \\ \mathbf{\Phi_u} \end{bmatrix} = I \\
& \mathbf{\Phi_x}, \mathbf{\Phi_u} \in z^{-1}\mathcal{RH}_\infty \\
& \boxed{\begin{bmatrix} \mathbf{\Phi_x} \\ \mathbf{\Phi_u} \end{bmatrix} \in \mathcal{S}}
\end{aligned}
$$

# System Level Synthesis (SLS)

- Considering the system response $\{\mathbf{\Phi_x}, \mathbf{\Phi_u}\}$, SLS allows easy incorporation of system level objective $g$ and constraints $\mathcal{S}$.
- Once the optimization problem is solved, the optimal controller is given by $\mathbf{K} = \mathbf{\Phi_u}\mathbf{\Phi_x}^{-1}$.

$$
\begin{aligned}
\min \quad & g(\mathbf{\Phi_x}, \mathbf{\Phi_u}) \\
\text{s.t.} \quad & \begin{bmatrix} zI - A & -B \end{bmatrix} \begin{bmatrix} \mathbf{\Phi_x} \\ \mathbf{\Phi_u} \end{bmatrix} = I \\
& \mathbf{\Phi_x}, \mathbf{\Phi_u} \in z^{-1}\mathcal{RH}_\infty \\
& \begin{bmatrix} \mathbf{\Phi_x} \\ \mathbf{\Phi_u} \end{bmatrix} \in \mathcal{S}
\end{aligned}
$$

# Scalability of SLS

- Solving for $\{\mathbf{\Phi_x}, \mathbf{\Phi_u}\}$ is not trivial, especially for large-scale systems.

$$
\begin{aligned}
\min \quad & g(\mathbf{\Phi_x}, \mathbf{\Phi_u}) \\
\text{s.t.} \quad & \begin{bmatrix} zI - A & -B \end{bmatrix} \begin{bmatrix} \mathbf{\Phi_x} \\ \mathbf{\Phi_u} \end{bmatrix} = I \\
& \mathbf{\Phi_x}, \mathbf{\Phi_u} \in z^{-1}\mathcal{RH}_\infty \\
& \begin{bmatrix} \mathbf{\Phi_x} \\ \mathbf{\Phi_u} \end{bmatrix} \in \mathcal{S}
\end{aligned}
$$

# Scalability of SLS

- Existing methods propose to exploit special system/controller structures to speed up the computation in parallel.

$$
\begin{aligned}
\min \quad & g(\mathbf{\Phi_x}, \mathbf{\Phi_u}) \\
\text{s.t.} \quad & \begin{bmatrix} zI - A & -B \end{bmatrix} \begin{bmatrix} \mathbf{\Phi_x} \\ \mathbf{\Phi_u} \end{bmatrix} = I \\
& \mathbf{\Phi_x}, \mathbf{\Phi_u} \in z^{-1}\mathcal{RH}_\infty \\
& \boxed{\begin{bmatrix} \mathbf{\Phi_x} \\ \mathbf{\Phi_u} \end{bmatrix} \in \mathcal{S}}
\end{aligned}
$$

# Scalability of SLS

- Existing methods propose to exploit special system/controller structures to speed up the computation in parallel.
  ⇒ The strategy does not work for general systems/controllers without special structures.

$$
\begin{aligned}
\min \quad & g(\mathbf{\Phi_x}, \mathbf{\Phi_u}) \\
\text{s.t.} \quad & \begin{bmatrix} zI - A & -B \end{bmatrix} \begin{bmatrix} \mathbf{\Phi_x} \\ \mathbf{\Phi_u} \end{bmatrix} = I \\
& \mathbf{\Phi_x}, \mathbf{\Phi_u} \in z^{-1}\mathcal{RH}_\infty \\
& \boxed{\begin{bmatrix} \mathbf{\Phi_x} \\ \mathbf{\Phi_u} \end{bmatrix} \in \mathcal{S}}
\end{aligned}
$$

# Scalability of SLS

- Is it possible to expedite the solving process without special structural assumptions?

$$
\begin{aligned}
\min \quad & g(\mathbf{\Phi_x}, \mathbf{\Phi_u}) \\
\text{s.t.} \quad & \begin{bmatrix} zI - A & -B \end{bmatrix} \begin{bmatrix} \mathbf{\Phi_x} \\ \mathbf{\Phi_u} \end{bmatrix} = I \\
& \mathbf{\Phi_x}, \mathbf{\Phi_u} \in z^{-1}\mathcal{RH}_\infty \\
& \begin{bmatrix} \mathbf{\Phi_x} \\ \mathbf{\Phi_u} \end{bmatrix} \in \mathcal{S}
\end{aligned}
$$

# Scalability of SLS

- Is it possible to expedite the solving process without special structural assumptions?
- Idea: Leveraging the structure of SLS feasibility constraints.

$$
\begin{aligned}
\min \quad & g(\mathbf{\Phi_x}, \mathbf{\Phi_u}) \\
\text{s.t.} \quad & \boxed{\begin{bmatrix} zI - A & -B \end{bmatrix} \begin{bmatrix} \mathbf{\Phi_x} \\ \mathbf{\Phi_u} \end{bmatrix} = I} \\
& \boxed{\mathbf{\Phi_x}, \mathbf{\Phi_u} \in z^{-1}\mathcal{RH}_\infty} \\
& \begin{bmatrix} \mathbf{\Phi_x} \\ \mathbf{\Phi_u} \end{bmatrix} \in \mathcal{S}
\end{aligned}
$$

# SLS via Dynamic Programming (DP)

- We can expand the SLS feasibility constraints

$$\begin{bmatrix} zI - A & -B \end{bmatrix} \begin{bmatrix} \mathbf{\Phi_x} \\ \mathbf{\Phi_u} \end{bmatrix} = I \quad \text{and} \quad \mathbf{\Phi_x}, \mathbf{\Phi_u} \in z^{-1}\mathcal{RH}_\infty$$

  for finite impulse response (FIR) system with horizon $T$ using spectral components $\Phi_x[\tau]$ and $\Phi_u[\tau]$:

$$\begin{aligned} &\Phi_x[\tau+1] = A\Phi_x[\tau] + B\Phi_u[\tau], \quad \forall \tau = 1, \ldots, T-1, \\ &\Phi_x[1] = I, \\ &A\Phi_x[T] + B\Phi_u[T] = 0. \end{aligned}$$

## SLS via Dynamic Programming (DP)

- When the objective $g$ is decomposable into a sum in the form

$$g(\mathbf{\Phi_x}, \mathbf{\Phi_u}) = \sum_{\tau=1}^{T} g_\tau(\Phi_x[\tau], \Phi_u[\tau]),$$

we can deem $g_\tau$ as the cost function at "time" $\tau$ and solve SLS by dynamic programming (DP).

$$\begin{aligned}
&\Phi_x[\tau+1] = A\Phi_x[\tau] + B\Phi_u[\tau], \quad \forall \tau = 1, \ldots, T-1, \\
&\Phi_x[1] = I, \\
&A\Phi_x[T] + B\Phi_u[T] = 0.
\end{aligned}$$

# SLS via Dynamic Programming (DP)

- To perform DP, we deem $\Phi_x[\tau]$ as the "state" and $\Phi_u[\tau]$ the "control" at time $\tau$, the SLS feasibility constraints describe a system dynamics with constrained states at the boundaries.

System dynamics:

$$
\begin{aligned}
&\Phi_x[\tau+1] = A\Phi_x[\tau] + B\Phi_u[\tau], && \forall \tau = 1, \ldots, T-1, \\
&\Phi_x[1] = I, && \text{Initial condition.} \\
&A\Phi_x[T] + B\Phi_u[T] = 0. && \text{Boundary condition.}
\end{aligned}
$$

# SLS via Dynamic Programming (DP)

- To perform DP, we deem $\Phi_x[\tau]$ as the "state" and $\Phi_u[\tau]$ the "control" at time $\tau$, the SLS feasibility constraints describe a system dynamics with constrained states at the boundaries.
- We then maintain a cost-to-go function and solve for the optimal control $\Phi_u[\tau]$ given state $\Phi_x[\tau]$ at each time $\tau$.

System dynamics:

$$\begin{aligned}
&\Phi_x[\tau+1] = A\Phi_x[\tau] + B\Phi_u[\tau], && \forall \tau = 1, \ldots, T-1, \\
&\Phi_x[1] = I, && \text{Initial condition.} \\
&A\Phi_x[T] + B\Phi_u[T] = 0. && \text{Boundary condition.}
\end{aligned}$$

# Challenge of SLS via DP

- Starting with $\Phi_x[1] = I$, the control $\Phi_u[\tau]$ must allow $A\Phi_x[T] + B\Phi_u[T] = 0$ to be satisfied by some $\Phi_u[T]$ at time $T$.

System dynamics:

$$\begin{aligned} &\Phi_x[\tau+1] = A\Phi_x[\tau] + B\Phi_u[\tau], && \forall \tau = 1, \dots, T-1, \\ &\Phi_x[1] = I, && \text{Initial condition.} \\ &A\Phi_x[T] + B\Phi_u[T] = 0. && \text{Boundary condition.} \end{aligned}$$

# Challenge of SLS via DP

- Starting with $\Phi_x[1] = I$, the control $\Phi_u[\tau]$ must allow $A\Phi_x[T] + B\Phi_u[T] = 0$ to be satisfied by some $\Phi_u[T]$ at time $T$.
  $\Rightarrow$ The admissible control $\Phi_u[\tau]$ is not a free variable but dependent on the time $\tau$ and state $\Phi_x[\tau]$.

  System dynamics:

$$
\begin{aligned}
&\Phi_x[\tau+1] = A\Phi_x[\tau] + B\Phi_u[\tau], && \forall \tau = 1, \dots, T-1, \\
&\Phi_x[1] = I, && \text{Initial condition.} \\
&A\Phi_x[T] + B\Phi_u[T] = 0. && \text{Boundary condition.}
\end{aligned}
$$

# Our Contribution: Enabling DP for SLS

- We show that the admissible control $\Phi_u[\tau]$ lies in a linear subspace, offset by a linearly mapped $\Phi_x[\tau]$.

# Our Contribution: Enabling DP for SLS

- We show that the admissible control $\Phi_u[\tau]$ lies in a linear subspace, offset by a linearly mapped $\Phi_x[\tau]$.
- We can then incorporate the admissible control set into the backward recursion in the DP procedure.

# Our Contribution: Enabling DP for SLS

- We show that the admissible control $\Phi_u[\tau]$ lies in a linear subspace, offset by a linearly mapped $\Phi_x[\tau]$.
- We can then incorporate the admissible control set into the backward recursion in the DP procedure.
- We derive DP that
    - solves plain SLS without system level constraints;
    - approximates infinite horizon SLS;
    - incorporates entry-wise linear (system level) constraints.

# Example: $\mathcal{H}_2$ Objective

- We demonstrate the performance of DP using the $\mathcal{H}_2$ objective

$$g(\mathbf{\Phi_x}, \mathbf{\Phi_u}) = \|C\mathbf{\Phi_x} + D\mathbf{\Phi_u}\|_{\mathcal{H}_2}^2 = \sum_{\tau=1}^{T} g_\tau(\Phi_x[\tau], \Phi_u[\tau])$$

where $C$ and $D$ are some matrices and

$$g_\tau(\Phi_x[\tau], \Phi_u[\tau]) = \|C\Phi_x[\tau] + D\Phi_u[\tau]\|_F^2 \,.$$

# Scalability with System Size

- Chain-like system (banded $A$ and $B$) of size $N_x$ with fixed $C$ and $D$ in the objective.
- Comparing methods:
  - DP,
  - CVX,
  - naive Lagrange multiplier solution.
- Scenarios:
  - plain SLS,
  - SLS with locality constraints.

# Scalability with System Size – DP Scales the Best

**Table 1:** Average synthesis time of plain SLS for random chain-like systems.

| $N_x$ | Synthesis Time (ms) | | |
|---|---|---|---|
| | DP | CVX | Lagrange Multiplier |
| 5 | 5.11 | 72.44 | 54.86 |
| 10 | 6.60 | 90.55 | 1176.51 |
| 15 | 8.42 | 136.76 | 9586.75 |
| 20 | 11.25 | 244.37 | 45782.15 |

# Scalability with System Size – DP Scales the Best

**Table 2:** Average synthesis time of SLS with locality constraints for random chain-like systems.

| $N_x$ | Synthesis Time (ms) | | |
|---|---|---|---|
| | DP | CVX | Lagrange Multiplier |
| 5 | 12.35 | 217.81 | 479.60 |
| 10 | 129.48 | 1411.00 | 78405.76 |
| 15 | 685.03 | 4890.28 | 1013700.11 |
| 20 | 1968.85 | 8549.75 | not feasible |

# Random Objectives – DP Scales the Best

**Table 3:** Average synthesis time of plain SLS for random chain-like systems under random objectives.

| $N_x$ | Synthesis Time (ms) / Solvable Rate | |
|---|---|---|
| | DP | CVX |
| 5 | 3.35 / 100% | 44.20 / 100% |
| 10 | 4.18 / 100% | 92.33 / 100% |
| 15 | 5.09 / 100% | 240.10 / 100% |
| 20 | 6.81 / 100% | 535.77 / 100% |

# Random Objectives – DP Scales the Best

**Table 4:** Average synthesis time of SLS with locality constraints for random chain-like systems under random objectives.

| $N_x$ | Synthesis Time (ms) / Solvable Rate | |
|---|---|---|
| | DP | CVX |
| 5 | 7.07 / 100% | 104.21 / 100% |
| 10 | 120.89 / 100% | 664.09 / 91% |
| 15 | 590.84 / 100% | 1989.17 / 85% |
| 20 | 1498.32 / 100% | 4051.55 / 82% |

# Cost for FIR System

- Enforcing the boundary constraint

$$A\Phi_x[T] + B\Phi_u[T] = 0.$$

is the main challenge of DP. If we omit the constraint, DP becomes much easier as the control $\Phi_u[\tau]$ is no longer constrained.

# Cost for FIR System

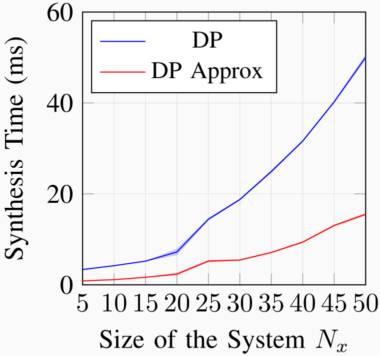
- Enforcing the boundary constraint

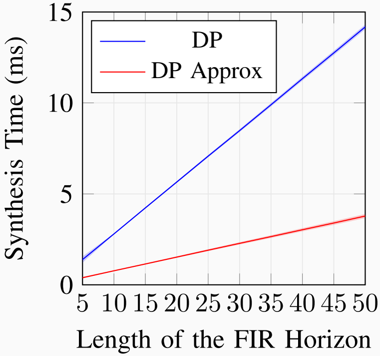
$$A\Phi_x[T] + B\Phi_u[T] = 0.$$

  is the main challenge of DP. If we omit the constraint, DP becomes much easier as the control $\Phi_u[\tau]$ is no longer constrained.

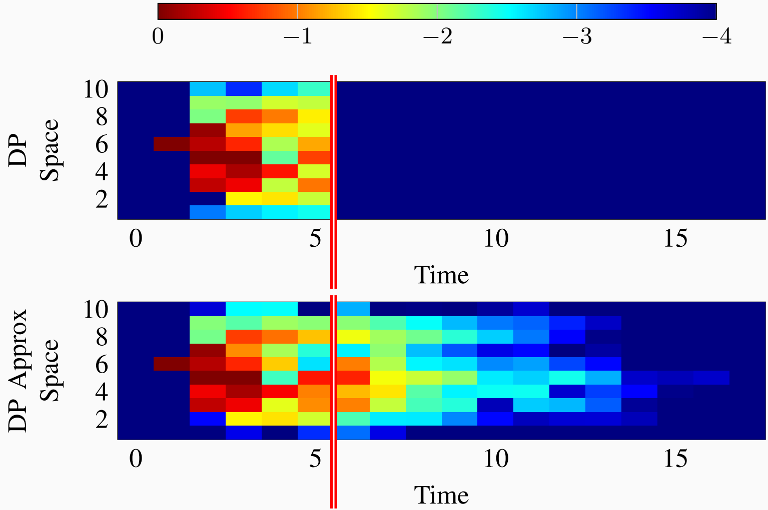
- In exchange, the resulting controlled system will not be FIR within horizon $T$. Rather, the simpler DP (DP Approx) can only approximate the SLS solution, and the approximating solution agrees with the true solution when the horizon is long.

# Cost for FIR System – DP Approx is Faster

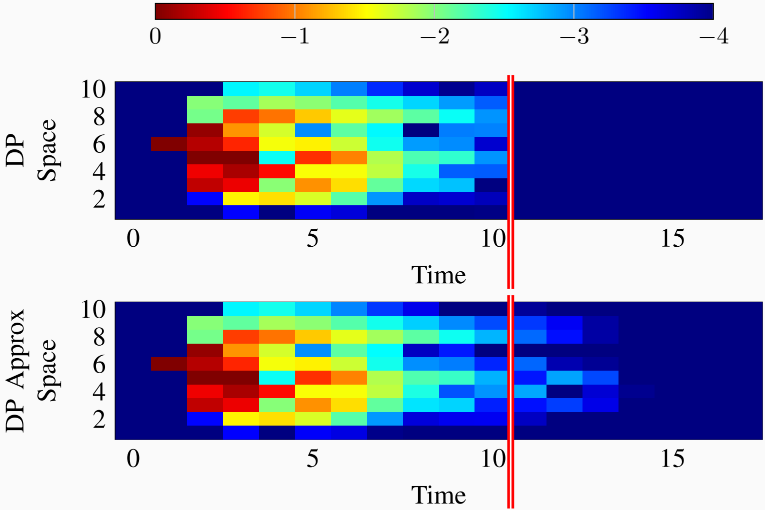


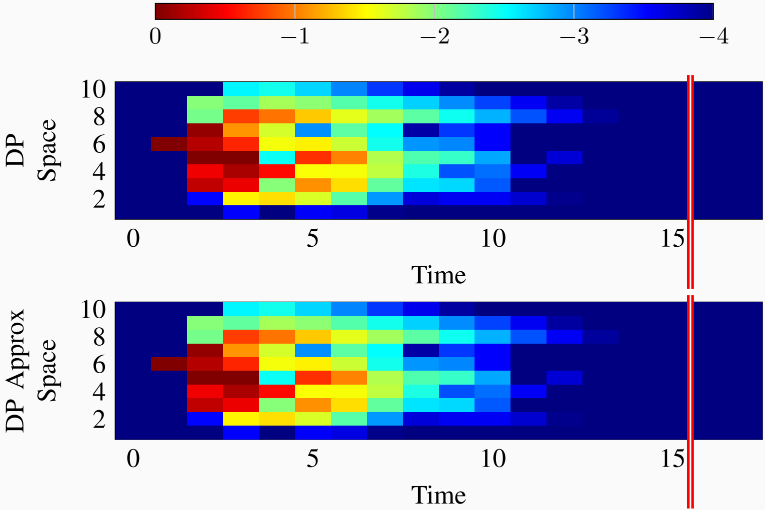
# Cost for FIR System – DP Ensures FIR



# Cost for FIR System – DP Ensures FIR



# Cost for FIR System – DP Ensures FIR



# Conclusion and Future Directions

- We derive DP that
  - solves plain SLS without system level constraints;
  - approximates infinite horizon SLS;
  - incorporates entry-wise linear (system level) constraints.
- Future directions:
  - Output-feedback SLS.
  - More classes of system level constraints.
  - SLS DP for model predictive control.