Semidefinite Programming Algorithm for Distributed Stability Analysis

Anders Hansson

In collaboration with :

Sina Khoshfetrat Pakazad

Linköpings universitet

Martin S. Andersson

UCLA

Anders Rantzer

Lunds universitet

Layout

- Convex Feasibility Problem.
- Nonlinear Cimmino Algorithm.
- Modified Nonlinear Cimmino Algorithm.
- ADMM Based Algorithm.
- Robustness Analysis of Dynamical Systems.
- Numerical Results.
- Conclusions.

Problem Statement

Convex Feasibility problem

$$\begin{aligned} &\text{Find} && x \\ &\text{subj. to} && x \in \bigcap_{i=1}^{N} C_i . \end{aligned}$$

Problem Statement

Convex Feasibility problem

$$\begin{aligned} &\text{Find} && x \\ &\text{subj. to} && x \in \bigcap_{i=1}^{N} C_i. \end{aligned}$$

- $x \in R^n$.
- C_is are defined via convex inequalities.

Nonlinear Cimmino Algorithm

- Define

$$P_{C_i}\left(\bar{x}\right) = \underset{x \in C_i}{\operatorname{argmin}} \left\| x - \bar{x} \right\|_2^2 .$$

Nonlinear Cimmino Algorithm

- Define

$$P_{C_i}\left(\bar{x}\right) = \underset{x \in C_i}{\operatorname{argmin}} \left\| x - \bar{x} \right\|_2^2 .$$

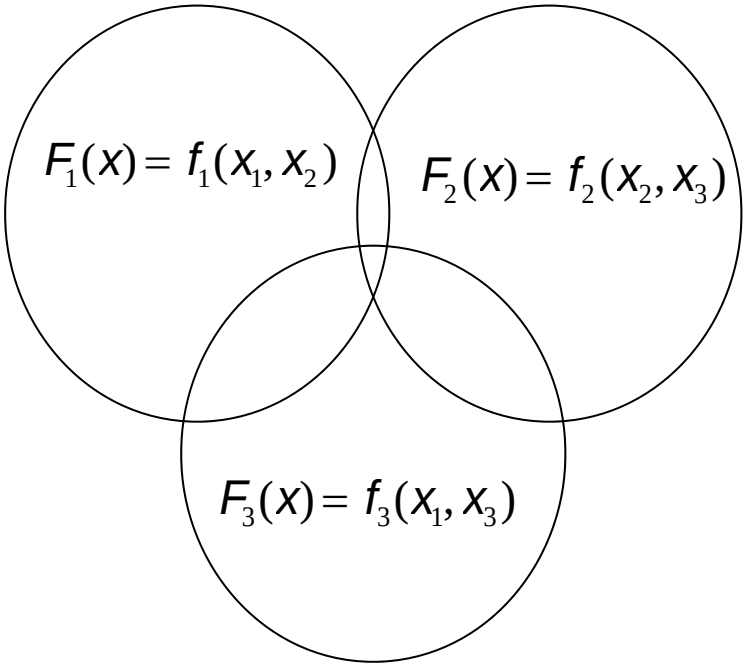
- Nonlinear Cimmino Algorithm

$$x^{k+1} = P\left(x^k\right)$$

where $P\left(x^k\right) = \sum_{i=1}^{N} \alpha_i P_{C_i}\left(x^k\right)$ with $\sum_{i=1}^{N} \alpha_i = 1$ and $\alpha_i > 0.$

An Example

$$C_i = \left\{ x \middle| F_i(x) \leq 0 \right\}$$



Indices

- Let $\mathbf{J}^i$ and J^i denote the vector and the set of indices of the variables in the optimization vector that are affected by the i^{th} constraint, respectively.

Indices

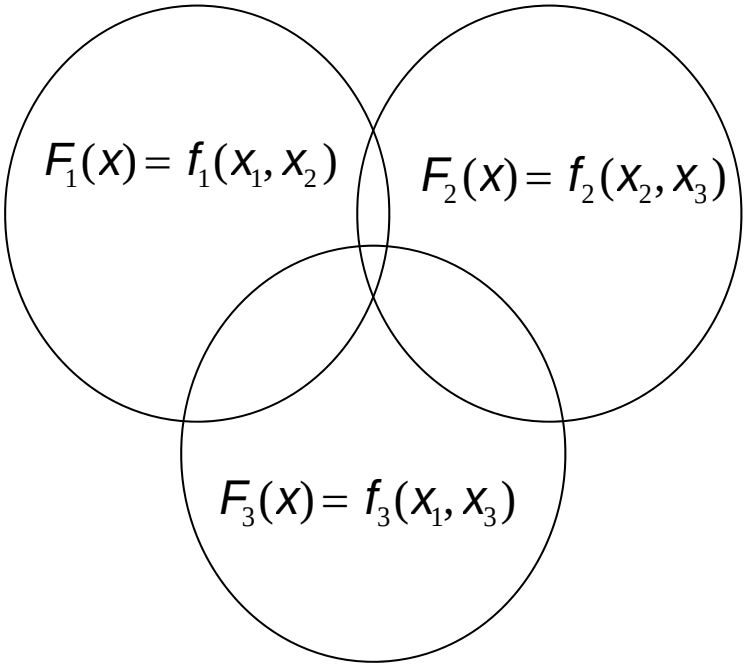
- Let $\mathbf{J}^i$ and J^i denote the vector and the set of indices of the variables in the optimization vector that are affected by the i^{th} constraint, respectively.
- Let $\mathbf{I}^i$ and I^i denote the vector and set of indices of the constraints that depend on the i^{th} variable, respectively.

An Example

- Constraints

$$C_i = \{x | F_i(x) \leq 0\}$$

- Then for instance
  - $J^1 = \{1,2\}, J^3 = \{1,3\}.$
  - $I^1 = \{1,3\}, I^3 = \{2,3\}.$

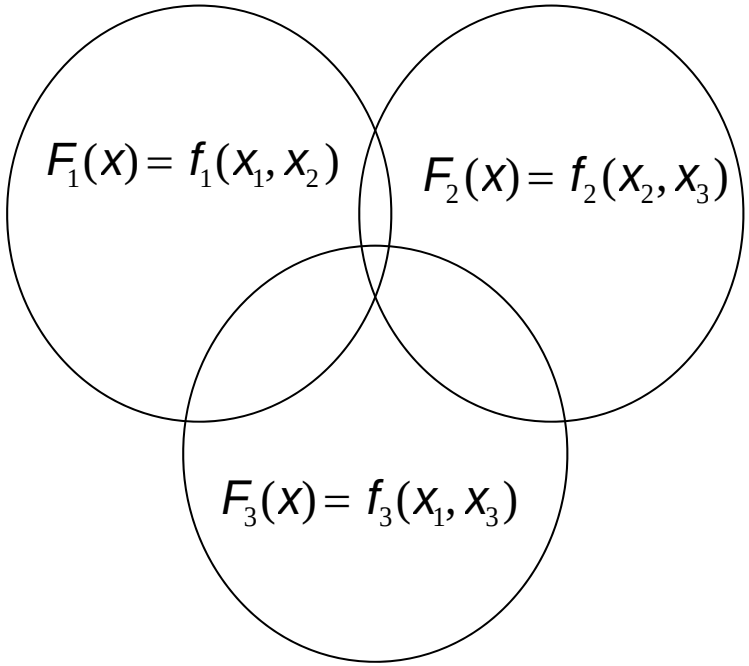


An Example

- Constraints

$$C_i = \{x | F_i(x) \leq 0\}$$

- Then for instance
  - $J^1 = \{1,2\}, J^3 = \{1,3\}.$
  - $I^1 = \{1,3\}, I^3 = \{2,3\}.$
- $\left(P_{C_2}(x)\right)_1 = x_1$



Closer Look

- Nonlinear Cimmino algorithm

$$x^{k+1} = P\left(x^k\right) = \sum_{j=1}^{N} \alpha_j P_{C_j}\left(x^k\right).$$

Closer Look

- Nonlinear Cimmino algorithm

$$x^{k+1} = P\left(x^k\right) = \sum_{j=1}^{N} \alpha_j P_{C_j}\left(x^k\right).$$

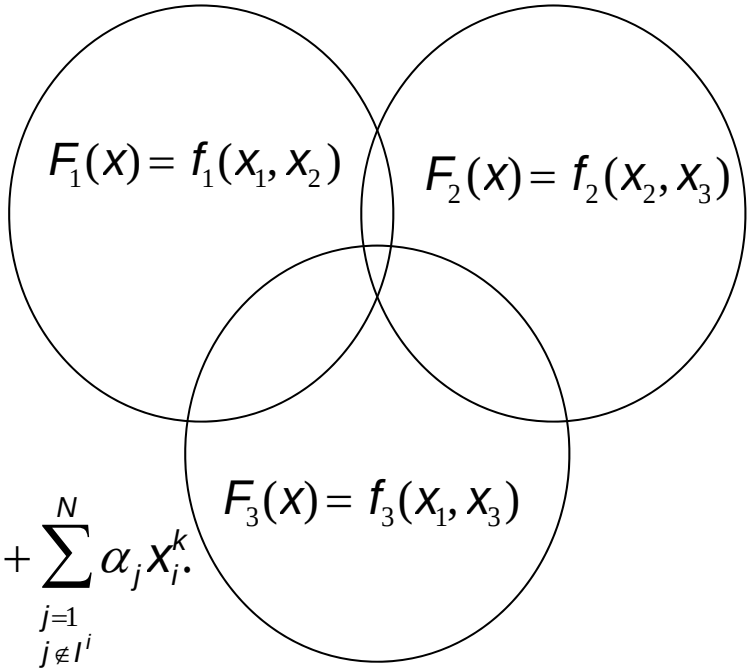
- Component-wise

$$x_i^{k+1} = \sum_{j=1}^{N} \alpha_j \left(P_{C_j}\left(x^k\right)\right)_i = \sum_{j \in I^i} \alpha_j \left(P_{C_j}\left(x^k\right)\right)_i + \sum_{\substack{j=1 \\ j \notin I^i}}^{N} \alpha_j x_i^k.$$

Closer Look

- Nonlinear Cimmino algorithm

$$x^{k+1} = P\left(x^k\right) = \sum_{j=1}^{N} \alpha_j P_{C_j}\left(x^k\right).$$



- Component-wise

$$x_i^{k+1} = \sum_{j=1}^{N} \alpha_j \left(P_{C_j}\left(x^k\right)\right)_i = \sum_{j \in I^i} \alpha_j \left(P_{C_j}\left(x^k\right)\right)_i + \sum_{\substack{j=1 \\ j \notin I^i}}^{N} \alpha_j x_i^k.$$

- For instance

$$x_1^{k+1} = \sum_{j=1}^{3} \alpha_j \left(P_{C_j}\left(x^k\right)\right)_1 = \alpha_1 \left(P_{C_1}\left(x^k\right)\right)_1 + \alpha_3 \left(P_{C_3}\left(x^k\right)\right)_1 + \alpha_2 x_1^k.$$

Modification

- Nonlinear Cimmino algorithm

$$x^{k+1} = P\left(x^k\right) = \sum_{j=1}^{N} \alpha_j P_{C_j}\left(x^k\right).$$

- Component-wise

$$x_i^{k+1} = \sum_{j=1}^{N} \alpha_j \left(P_{C_j}\left(x^k\right)\right)_i = \sum_{j \in I^i} \alpha_j \left(P_{C_j}\left(x^k\right)\right)_i + \sum_{\substack{j=1 \\ j \notin I^i}}^{N} \alpha_j x_i^k.$$

Modification

- Nonlinear Cimmino algorithm

$$x^{k+1} = P\left(x^k\right) = \sum_{j=1}^{N} \alpha_j P_{C_j}\left(x^k\right).$$

- Component-wise

$$x_i^{k+1} = \sum_{j=1}^{N} \alpha_j \left(P_{C_j}\left(x^k\right)\right)_i = \sum_{j \in I^i} \alpha_j \left(P_{C_j}\left(x^k\right)\right)_i + \sum_{\substack{j=1 \\ j \notin I^i}}^{N} \alpha_j x_i^k.$$

Modification

- Nonlinear Cimmino algorithm

$$x^{k+1} = P\left(x^k\right) = \sum_{j=1}^{N} \alpha_j P_{C_j}\left(x^k\right).$$

- Component-wise

$$x_i^{k+1} = \sum_{j=1}^{N} \alpha_j \left(P_{C_j}\left(x^k\right)\right)_i = \sum_{j \in I^i} \alpha_j \left(P_{C_j}\left(x^k\right)\right)_i + \sum_{\substack{j=1 \\ j \notin I^i}}^{N} \alpha_j x_i^k.$$

- Modification

$$x_i^{k+1} = \sum_{j \in I^i} \bar{\alpha}_j \left(P_{C_j}\left(x^k\right)\right)_i.$$

such that $\bar{\alpha}_j > 0$ and $\sum_{j \in I^i} \bar{\alpha}_j = 1$. From now on $\bar{\alpha}_j = \frac{1}{I_i}$, where I_i is the number of elements in I^i.

Structure Exploitation

- Possible to define $\bar{C}_j$ such that

$$\left(P_{C_j}\left(x^k\right)\right)_i = \left(P_{\bar{C}_j}\left(x^k_{\mathbf{J}^j}\right)\right)_{t_j^i}, \text{for some } t_j^i.$$

Structure Exploitation

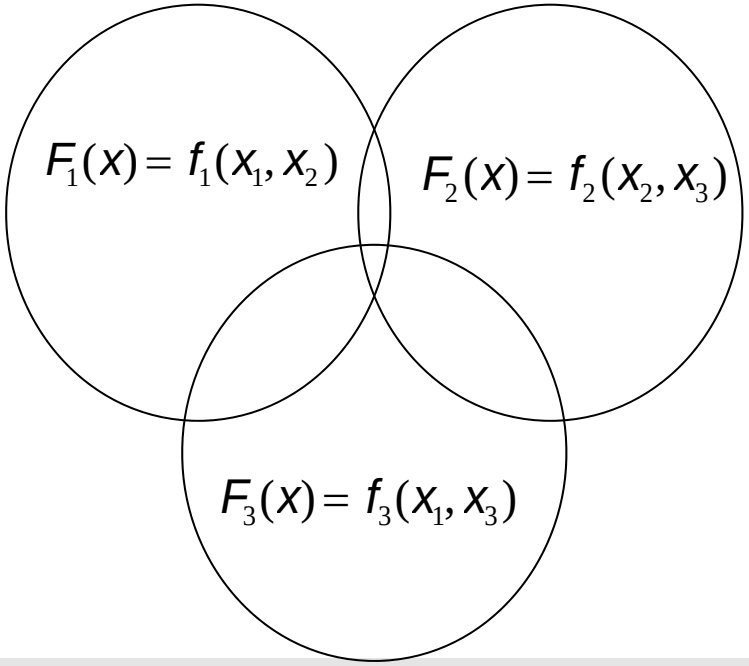
- Possible to define $\bar{C}_j$ such that

$$\left(P_{C_j}\left(x^k\right)\right)_i = \left(P_{\bar{C}_j}\left(x^k_{\mathbf{J}^j}\right)\right)_{t^i_j}, \text{for some } t^i_j.$$

- For example, $\bar{C}_1 = \left\{ \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \middle\| f_1(x_1, x_2) \le 0 \right\}$.

  Then $C_1 = \bar{C}_1 \times R$ and

$$\left(P_{C_1}\left(\begin{bmatrix} x^k_1 \\ x^k_2 \\ x^k_3 \end{bmatrix} \right) \right)_{1\text{or } 2} = \left(P_{\bar{C}_2}\left(\begin{bmatrix} x^k_1 \\ x^k_2 \end{bmatrix} \right) \right)_{1\text{or } 2}.$$



Restatement

- Modified Nonlinear Cimmino Algorithm

$$x_i^{k+1} = \frac{1}{l_i} \sum_{j \in I^i} \left(P_{C_j}\left(x^k\right) \right)_i = \frac{1}{l_i} \sum_{j \in I^i} \left(P_{\bar{C}_j}\left(x^k_{\mathbf{J}^j}\right) \right)_{t_j^i}.$$

- Can be rewritten as below

$$s^{j,k+1} = P_{\bar{C}_i}\left(x^k_{\mathbf{J}^i}\right),$$

$$x_i^{k+1} = \frac{1}{l_i} \sum_{j \in I^i} s^{j,k+1}_{t_j^i}.$$

This can also be achieved by implementing dual ascent on the Lagrangian for an equivalent problem with constraints $s^j = x_{\mathbf{J}^j}$ for dual decomposition.

Alternating Direction Method of Multipliers (ADMM)

- Similarly by applying the ADMM to the Augmented Lagrangian of the equivalent problem

$$s^{j,k+1} = P_{\overline{C}_i}\left(x^k_{\mathbf{J}^j} - \overline{\lambda}^{i,k}\right),$$

$$x_i^{k+1} = \frac{1}{I_i}\sum_{j\in I^i} s^{i,k+1}_{t^i_j} = \frac{1}{I_i}\sum_{j\in I^i}\left(P_{\overline{C}_j}\left(x^k_{\mathbf{J}^j} - \overline{\lambda}^{j,k}\right)\right)_{t^i_j}.$$

$$\overline{\lambda}^{i,k+1} = \overline{\lambda}^{i,k} + \left(s^{j,k+1} - x^{k+1}_{\mathbf{J}^i}\right).$$

Summary

- Modified nonlinear Cimmino algorithm

$$s^{i,k+1} = P_{\bar{C}_i}\left(x^k_{\mathbf{J}^i}\right),$$

$$x_i^{k+1} = \frac{1}{l_i}\sum_{j \in I^i}\left(P_{\bar{C}_i}\left(x^k_{\mathbf{J}^i}\right)\right)_{t^i_j}.$$

Summary

- Modified nonlinear Cimmino algorithm

$$s^{j,k+1} = P_{\bar{C}_i}\left(x^k_{\mathbf{J}^j}\right),$$

$$x_i^{k+1} = \frac{1}{l_i}\sum_{j\in I^i}\left(P_{\bar{C}_i}\left(x^k_{\mathbf{J}^j}\right)\right)_{t^i_j}.$$

- ADMM based algorithm

$$s^{j,k+1} = P_{\bar{C}_i}\left(x^k_{\mathbf{J}^j} - \bar{\lambda}^{i,k}\right),$$

$$x_i^{k+1} = \frac{1}{l_i}\sum_{j\in I^i}\left(P_{\bar{C}j}\left(x^k_{\mathbf{J}^j} - \bar{\lambda}^{j,k}\right)\right)_{t^i_j}.$$

$$\bar{\lambda}^{i,k+1} = \bar{\lambda}^{i,k} + \left(s^{j,k+1} - x^{k+1}_{\mathbf{J}^j}\right).$$

Some remarks

- Both presented algorithms are Simultaneous projection methods and have similar update rules.

Some remarks

- Both presented algorithms are Simultaneous projection methods and have similar update rules.
- The ADMM based method utilizes a Dykstra like correction term in its update rule.

Some remarks

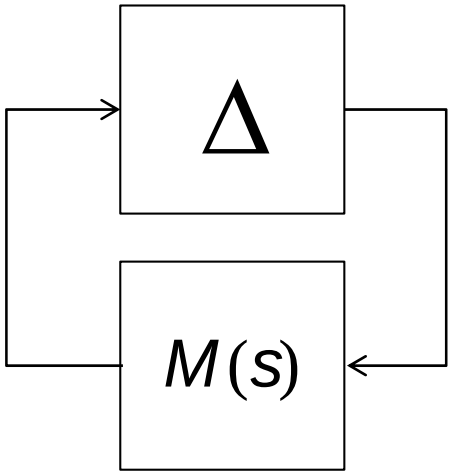
- Both presented algorithms are Simultaneous projection methods and have similar update rules.
- The ADMM based method utilizes a Dykstra like correction term in its update rule.
- Proposed simultaneous projection methods can also be viewed as counter parts to successive projection methods, Von Neumann's and Dykstra's alternating projection methods.

μ-Analysis

- Consider the system description

$$Y(s) = M(s)U(s),$$
$$U(s) = \Delta Y(s).$$

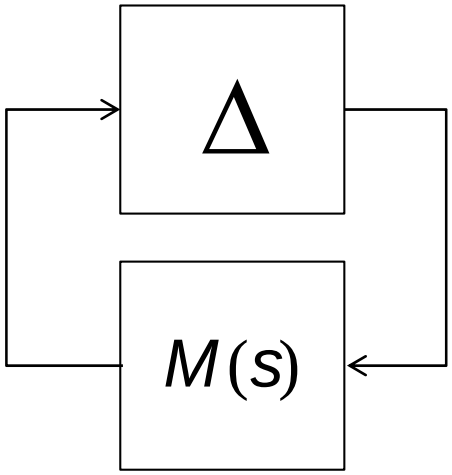
where $\Delta = \mathrm{diag}(\delta_i)$, with $\delta_i \in R, |\delta_i| \leq 1.$



μ-Analysis

- Consider the system description

$$Y(s) = M(s)U(s),$$
$$U(s) = \Delta Y(s).$$



where $\Delta = \mathrm{diag}(\delta_i)$, with $\delta_i \in R, |\delta_i| \le 1.$

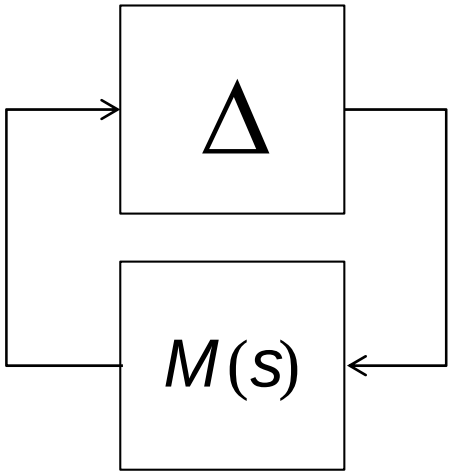
- This system is said to be robustly stable if

$$M(j\omega)^* X(\omega) M(j\omega) - \mu^2 X(\omega) \le -\varepsilon I$$

for some $X(\omega) \ge \varepsilon I$ and $0 < \mu < 1,$ for all $\omega.$

μ-Analysis

- Consider the system description



$$Y(s) = M(s)U(s),$$
$$U(s) = \Delta Y(s).$$

where $\Delta = \mathrm{diag}(\delta_i)$, with $\delta_i \in R, |\delta_i| \le 1.$

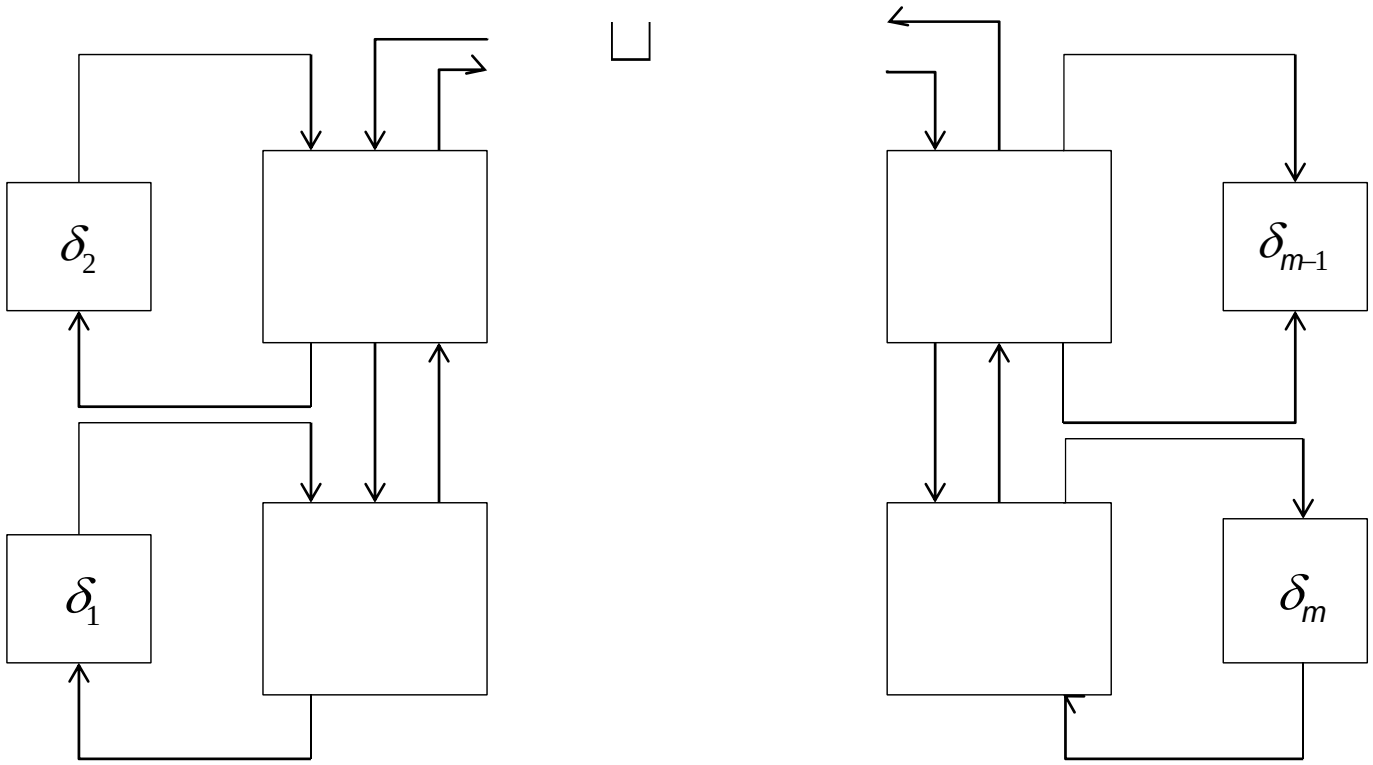
- This system is said to be robustly stable if

$$M(j\omega)^* X(\omega) M(j\omega) - \mu^2 X(\omega) \le -\varepsilon I$$

for some $X(\omega) \ge \varepsilon I$ and $0 < \mu < 1,$ for all $\omega.$

- Consider only real-valued M and one $\omega.$

Application



Systems Matrix and the LMI

- Systems Matrix

$$M = \begin{bmatrix} g_1 & h_1 & 0 & 0 & 0 \\ f_1 & g_2 & h_2 & 0 & 0 \\ 0 & f_2 & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & g_{m-1} & h_{m-1} \\ 0 & 0 & 0 & f_{m-1} & g_m \end{bmatrix}$$

Systems Matrix and the LMI

- Systems Matrix

$$M = \begin{bmatrix} g_1 & h_1 & 0 & 0 & 0 \\ f_1 & g_2 & h_2 & 0 & 0 \\ 0 & f_2 & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & g_{m-1} & h_{m-1} \\ 0 & 0 & 0 & f_{m-1} & g_m \end{bmatrix}$$

- The corresponding LMI to solve

$$\begin{bmatrix} * & * & * & & & 0 \\ * & * & * & * & & \\ * & * & * & * & * & \\ & * & * & * & * & * \\ & & * & * & * & * \\ 0 & & & * & * & * \end{bmatrix} \leq 0$$

Systems Matrix and the LMI

- Systems Matrix

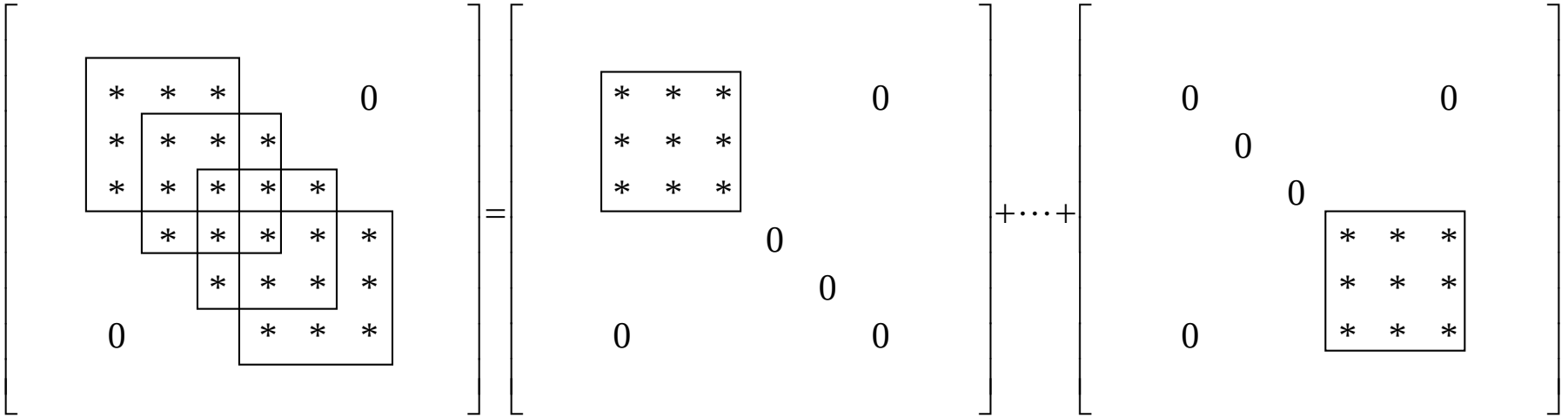
$$M = \begin{bmatrix} g_1 & h_1 & 0 & 0 & 0 \\ f_1 & g_2 & h_2 & 0 & 0 \\ 0 & f_2 & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & g_{m-1} & h_{m-1} \\ 0 & 0 & 0 & f_{m-1} & g_m \end{bmatrix}$$

- The corresponding LMI to solve

$$\begin{bmatrix} * & * & * & & & 0 \\ * & * & * & * & & \\ * & * & * & * & * & \\ & * & * & * & * & * \\ & & * & * & * & * \\ 0 & & & * & * & * \end{bmatrix} \leq 0$$

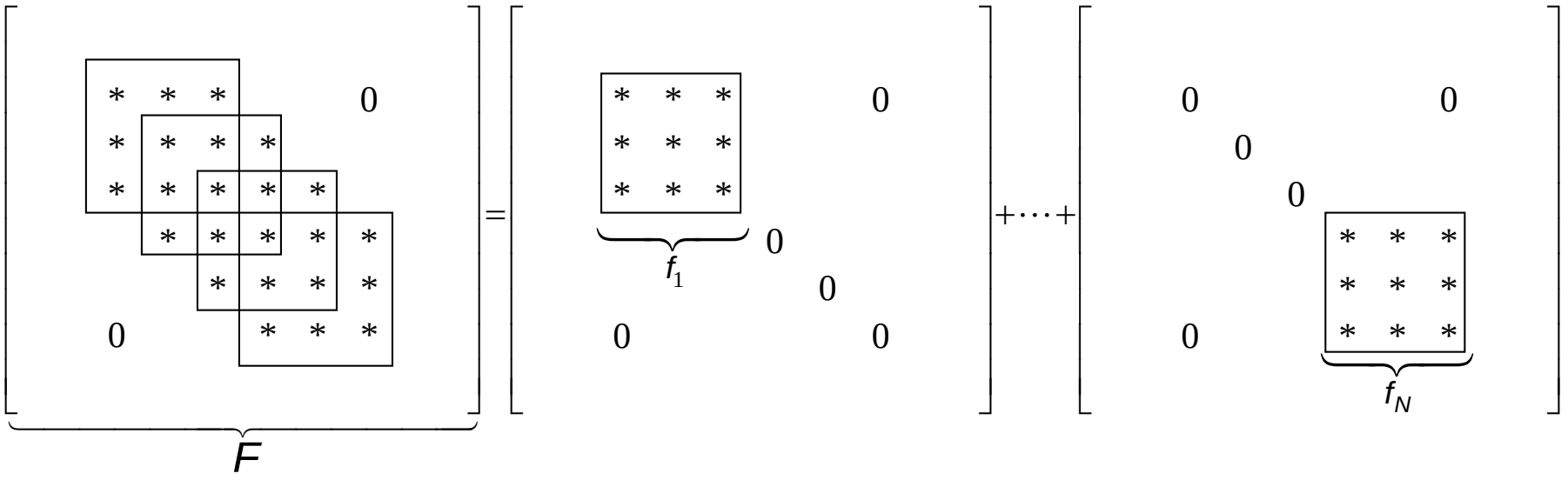
- Special Case of Chordal sparsity pattern.

Range space conversion



AUTOMATIC CONTROL
REGLERTEKNIK
LINKÖPINGS UNIVERSITET

Range space conversion

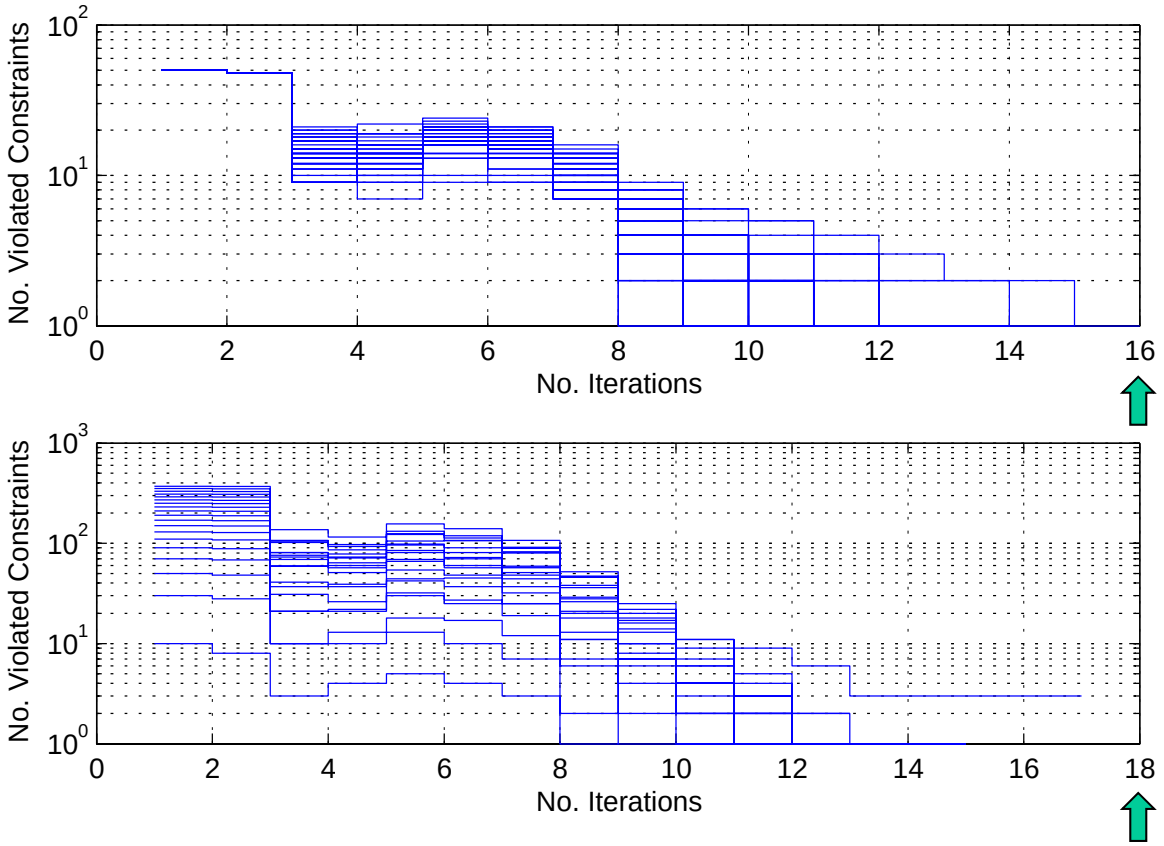


$$F \leq 0 \;\Leftrightarrow\; \exists \text{ a decompostion as above and } f_1 \leq 0, \square \;, f_N \leq 0$$

Modified Nonlinear Cimmino Algorithm Results

$N = 50$

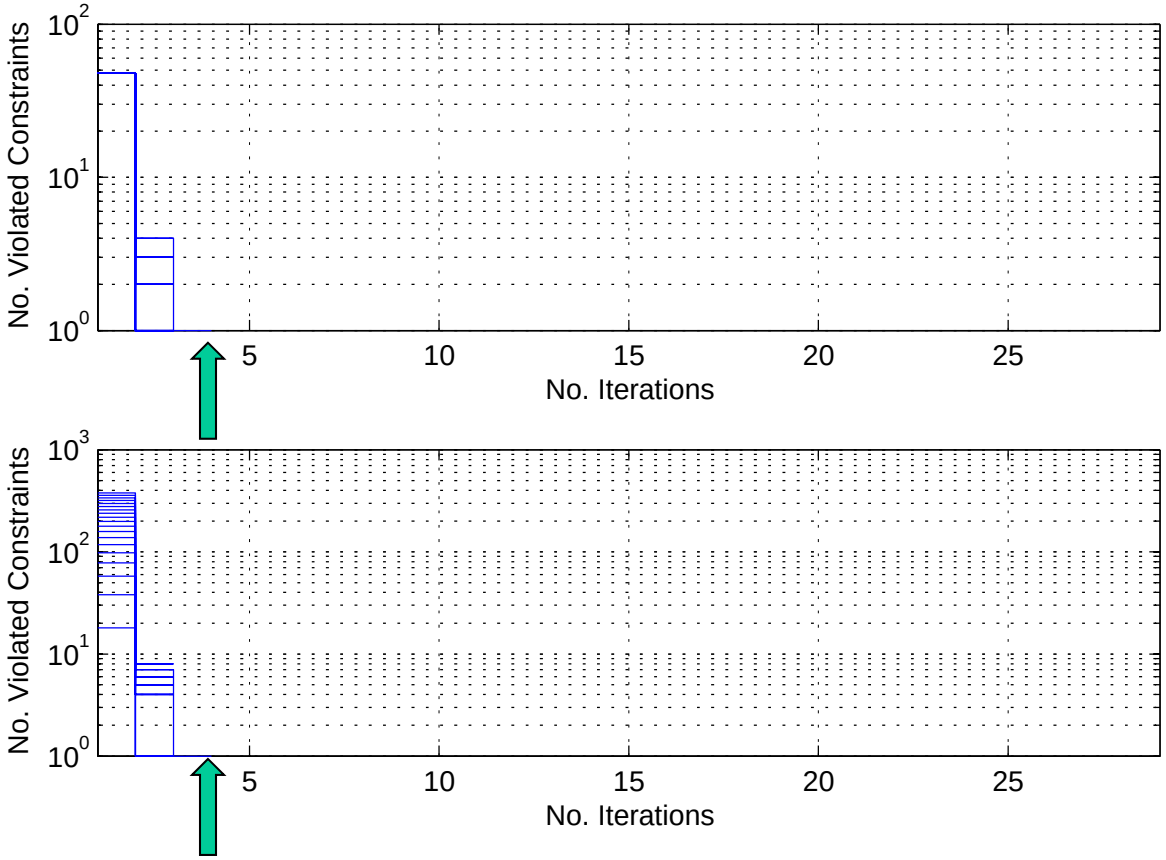
$N = 10 - 490$



ADMM Based Results

$N = 50$

$N = 10 - 490$



Convergence

- Global feasibility check

$$F(x) \leq 0$$

Convergence

- Global feasibility check

$$F(x) \leq 0$$

- Local feasibility check

  - Modified nonlinear Cimmino algorithm

$$s^j = x_{\mathbf{J}^j}$$

Convergence

- Global feasibility check

$$F(x) \leq 0$$

- Local feasibility check

  - Modified nonlinear Cimmino algorithm

$$s^j = x_{\mathbf{J}^j}$$

  - ADMM based algorithm

$$s^j = x_{\mathbf{J}^j} \;\; \& \;\; \bar{\lambda}_i = 0$$

Convergence

- Global feasibility check

$$F(x) \leq 0$$

- Local feasibility check

  - Modified nonlinear Cimmino algorithm

$$s^j = x_{\mathbf{J}^j}$$

  - ADMM based algorithm

$$s^j = x_{\mathbf{J}^j} \;\; \& \;\; \bar{\lambda}_i = 0$$

- Modified nonlinear Cimmino algorithm

$$\text{Global feasibility} \Leftrightarrow \text{Local feasibility}$$

Convergence

- Global feasibility check

$$F(x) \leq 0$$

- Local feasibility check

  - Modified nonlinear Cimmino algorithm

$$s^j = x_{\mathbf{J}^j}$$

  - ADMM based algorithm

$$s^j = x_{\mathbf{J}^j} \;\; \& \;\; \bar{\lambda}_i = 0$$

- Modified nonlinear Cimmino algorithm

$$\text{Global feasibility} \Leftrightarrow \text{Local feasibility}$$

- ADMM based algorithm

$$\text{Global feasibility} \Leftarrow \text{Local feasibility}$$

Results Summary

	Modified NCA	ADMM Based
Global		Faster
Local	Faster	

Conclusions

- Investigated structure exploitation for simultaneous projection methods.

Conclusions

- Investigated structure exploitation for simultaneous projection methods.
- Application to robustness analysis for dynamical uncertain systems.

Conclusions

- Investigated structure exploitation for simultaneous projection methods.
- Application to robustness analysis for dynamical uncertain systems.
- Parallel/Distributed solutions are provided.

Conclusions

- Investigated structure exploitation for simultaneous projection methods.
- Application to robustness analysis for dynamical uncertain systems.
- Parallel/Distributed solutions are provided.
- Speed of convergence has been considerably improved.