An efficient non-condensed approach for model predictive control

Nilay Saraf

Advisor: Prof. Alberto Bemporad

oCPS Fall School, Eindhoven
October 30, 2019

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

ODYS Advanced Controls & Optimization

OCPS Optimizing Cyber Physical Systems

This research has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under grant agreement no. 674875.

Outline

- Optimization problem formulation
- **Equality constraint elimination** and proposed **nonlinear least-squares** approach with **bounded variables**
- Bounded-variable least-squares (**BVLS**) solver
- Problem **sparsity** and **matrix abstraction** (**build-free MPC**)
- Numerically-stable **sparse recursive QR factorization**
- Numerical results

Discrete-time nonlinear model

General multivariable nonlinear (NL) prediction model

$$\mathcal{M}(Y_k, U_k, S_k) = \mathbf{0}$$

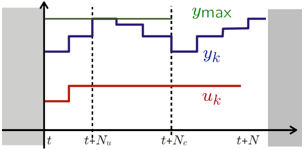
- Past inputs $U_k = (u_{k-n_\mathrm{b}}, \ldots, u_{k-1})$, $u_k \in \mathbb{R}^{n_u}$
- Past outputs $Y_k = (y_{k-n_\mathrm{a}}, \ldots, y_k)$, $y_k \in \mathbb{R}^{n_y}$
- Measured exogenous signals $S_k = (s_{k-n_\mathrm{c}}, \ldots, s_{k-1})$, $s_k \in \mathbb{R}^{n_s}$
- $n_\mathrm{a}, n_\mathrm{b}$ and n_c define the model order
- Special case (state-space model): $U_k = u_k$, $Y_k = x_k$

- Assumption: $\mathcal{M}$ is differentiable
- Examples: NL state-space models, deterministic **parameter-varying** NL-ARX models (black-box), I/O difference equations from first principles, **neural networks** with smooth activation function...
- On linearization about arbitrary $\hat{U}, \hat{Y}$:

$$-\ A\,(S_k)_0\,\Delta y_k \ = \ \sum_{j=1}^{n_\mathrm{a}} A\,(S_k)_j\,\Delta y_{k-j} \ + \ \sum_{j=1}^{n_\mathrm{b}} B\,(S_k)_j\,\Delta u_{k-j} \ + \ \mathcal{M}(\hat{Y},\ \hat{U},\ S_k),$$

A, B represent required Jacobians

MPC problem setup



- Prediction horizon N, control horizon N_u
- $z_k = \{u_k, \ldots, u_{k+N_u-1}, y_{k+1}, \ldots, y_{k+N}\}$
  =vector of decision variables
- Performance index

$$\min_z \|f_k(z)\|_2^2$$

  Examples:
  - f_k is linear, $f_k(z) = W_k(z - z_{\text{ref},k})$ (standard tracking problem)
  - f_k is arbitrary nonlinear differentiable function
- Constraints
  - (nonlinear) equality constraints due to the prediction model $\mathcal{M}$
  - upper and lower bounds on inputs and outputs $p_k \leq z \leq q_k$
  - general inequality constraints $g(u_{k+j}, y_{k+j}) \leq 0$ can be softened and treated as equalities $g(u_{k+j}, y_{k+j}) + \sigma_j = 0$, with $\sigma_j \geq 0$

Constrained nonlinear programming (NLP) problem

- Consider tracking problem with quadratic costs for simplicity (everything immediately extends to arbitrary nonlinear costs $\|f_k(z)\|_2^2$)

Resulting NLP formulation at each sample step k

$$\begin{aligned} \min_{z} \ & \frac{1}{2}\|W_k(z - z_{\text{ref},k})\|_2^2 \\ \text{s.t.} \ & h_k(z, \phi_k) = \mathbf{0}, \\ & p_k \leq z \leq q_k. \end{aligned}$$

- Matrix W is often diagonal and the Jacobian of $h(z)$ is sparse and structured
- Initial condition vector ϕ consists of past I/O values

Proposed NMPC formulation

Key Idea

Soften equality constraints via quadratic penalties

$$\min_{p_k \leq z \leq q_k} \frac{1}{2}\|W_k(z - z_{\mathrm{ref},k})\|_2^2 + \frac{\rho}{2}\|h_k(z)\|_2^2$$

- Penalty parameter $\rho > 0$ is a large weight
- **Motivation**: model is uncertain anyway, so why impose $h_k(z, \phi_k) = \mathbf{0}$ exactly?
- The problem can be rewritten as

$$\min_{p_k \leq z \leq q_k} \frac{1}{2}\|r_k(z)\|_2^2, \; r_k(z) = \begin{bmatrix} \frac{1}{\sqrt{\rho}} W_k(z - z_{\mathrm{ref},k}) \\ h_k(z, \phi_k) \end{bmatrix}$$

- Box-constrained nonlinear least squares problem is always feasible
- Fast solution using bounded-variable nonlinear least squares (**BVNLLS**)
- Same control performance as with conventional NMPC/NLP

Bounded-variable nonlinear least squares (BVNLLS)

Problem: Sum-of-squares cost function with box constraints

$$\min_{p_k \leq z \leq q_k} \frac{1}{2}\|r_k(z)\|_2^2$$

- Gauss-Newton method: efficiently solves unconstrained nonlinear LS
  - Sequence of linear least-squares problems
  - Hessian (H) is approximated as $H \approx J^\top J$, $J = \nabla_z r(z)^\top$
  - Rapid convergence with good initial guess
  - Only first-order information (J) is needed
- Proposed solver: Gauss-Newton method with box constraints
- Line-search problem: Linear least-squares with box constraints (BVLS)
- Guaranteed convergence using sufficient decrease condition

Bounded-variable nonlinear least squares (BVNLLS)

Problem: Sum-of-squares cost function with box constraints

$$\min_{p_k \le z \le q_k} \frac{1}{2}\|r_k(z)\|_2^2$$

- BVNLLS: Gauss-Newton method with box constraints

Bounded-variable nonlinear least squares

Initialize $z^{(0)} \in \{z|p \le z \le q\}$, $j \leftarrow 0$

1: Update Jacobian $J \leftarrow \nabla_z r\left(z^{(j)}\right)^\top$; (Linearization)

2: Compute gradient $J^\top r$ of Lagrangian function;

3: If first-order optimality conditions are satisfied then **stop**;

4: Solve $\Delta \leftarrow \arg \min_{p-z^{(j)} \le \Delta \le q-z^{(j)}} \|J\Delta + r\left(z^{(j)}\right)\|_2^2$ via **BVLS solver**; (Line search)

5: Compute step-size $0 < \alpha \le 1$ for backtracking;

6: $z^{(j+1)} \leftarrow z^{(j)} + \alpha\Delta$; (Update iterate)

7: $j \leftarrow j+1$; **go to** Step 1;

- **Linear MPC** case exactly recovered by **single BVNLLS iteration**!

Bounded-variable least squares (BVLS)

Problem: Least-squares with box constraints

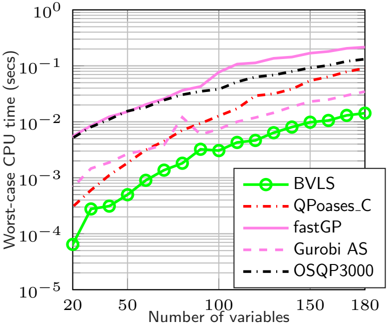
$$\min_{p \le x \le q} \frac{1}{2} \|Jx - b\|_2^2$$

- BVLS [2] is a primal active-set algorithm
- Finds solution x^* by iterating until the optimal active set ($\mathcal{A}^*$) is found
- Active set? $\mathcal{A} = \{\{i|x(i) = p(i)\} \cup \{i|x(i) = q(i)\}\}$
- Main computations:
  - Solve unconstrained LS: $J(:, \neg\mathcal{A})^\dagger \left(b - J(:, \mathcal{A})x(\mathcal{A})\right)$ (every iteration)
  - Gradient entries: $J(:, \mathcal{A})^\top (b - Jx)$ (in some iterations)
- $\mathcal{A}$ is updated by one index (inserted or removed)

  $\implies$ Subsequent LS problems are related by insertion or deletion of **1** column in J!

BVLS solver

- BVLS solves a sequence of related LS problems
- Efficient implementation with numerically stable recursive QR updates [3]
- Library free, simple arithmetic operations
- Stable also in single precision
- Suitable for embedded hardware platforms



Double precision, random **poorly-conditioned**, box-constrained LS problems (2.6GHz Intel Core i5 Mac)

- Implemented and tested on a real industrial PLC (paper under preparation)

BVLS for MPC: Problem Sparsity

Problem: Least-squares with box constraints

$$\min_{p \leq x \leq q} \frac{1}{2} \|Jx - b\|_2^2$$

- $J = \begin{bmatrix} W_k \\ \nabla_z h_k(z_k, \phi_k)^\top \end{bmatrix}$
- Problem can be constructed using sequence of affine models obtained from linearization over previously computed or guess trajectory

$$-A\left(S_k\right)_0^{(i)} \Delta y_k = \sum_{j=1}^{n_\mathrm{a}} A\left(S_k\right)_j^{(i)} \Delta y_{k-j} + \sum_{j=1}^{n_\mathrm{b}} B\left(S_k\right)_j^{(i)} \Delta u_{k-j} + \mathcal{M}(\hat{Y}^{(i)},\ \hat{U}^{(i)},\ S_k)$$

(i =prediction step)

$x = \Delta z_k = \{\Delta u_k, \Delta y_{k+1}, \ldots, \Delta u_{k+N_u-1}, \Delta y_{k+N_u}, \Delta y_{k+N_u+1}, \ldots, \Delta y_{k+N}\}$

- Outputs are kept as decision variables (non-condensed approach) for a larger but sparse problem formulation which is cheap to construct

BVLS for MPC: Problem Sparsity

Problem: Least-squares with box constraints

$$\min_{p \leq x \leq q} \frac{1}{2} \| \boldsymbol{J} x - b \|_2^2$$

$$\nabla_z h_k(z_k, \phi_k)^\top =$$

$$\left[\begin{array}{ccccccccc|ccccc}
B_1^{(1)} & A_0^{(1)} & \mathbf{0} & \mathbf{0} & \cdots & & & & & & \cdots & & & \mathbf{0} \\
B_2^{(2)} & A_1^{(2)} & B_1^{(2)} & A_0^{(2)} & \mathbf{0} & \cdots & & & & & & \cdots & & \mathbf{0} \\
 & \vdots & & \ddots & & & & \ddots & & & & & & \vdots \\
B_{N_\mathrm{u}}^{(N_\mathrm{u})} & A_{N_\mathrm{u}-1}^{(N_\mathrm{u})} & & & & \cdots & & B_1^{(N_\mathrm{u})} & A_0^{(N_\mathrm{u})} & \mathbf{0} & \cdots & & & \mathbf{0} \\
\hline
B_{N_\mathrm{u}+1}^{(N_\mathrm{u}+1)} & A_{N_\mathrm{u}}^{(N_\mathrm{u}+1)} & & \cdots & & B_3^{(N_\mathrm{u}+1)} & A_2^{(N_\mathrm{u}+1)} & \sum_{i=1}^{2} B_i^{(N_\mathrm{u}+1)} & A_1^{(N_\mathrm{u}+1)} & A_0^{(N_\mathrm{u}+1)} & \mathbf{0} & \cdots & & \mathbf{0} \\
B_{N_\mathrm{u}+2}^{(N_\mathrm{u}+2)} & A_{N_\mathrm{u}+1}^{(N_\mathrm{u}+2)} & \ddots & & \cdots & B_4^{(N_\mathrm{u}+2)} & A_3^{(N_\mathrm{u}+2)} & \sum_{i=1}^{3} B_i^{(N_\mathrm{u}+2)} & A_2^{(N_\mathrm{u}+2)} & A_1^{(N_\mathrm{u}+2)} & A_0^{(N_\mathrm{u}+2)} & \mathbf{0} & \cdots & \mathbf{0} \\
 & \vdots & & \ddots & & \ddots & & \vdots & & \vdots & & \ddots & & \mathbf{0} \\
B_{N_\mathrm{p}}^{(N_\mathrm{p})} & A_{N_\mathrm{p}-1}^{(N_\mathrm{p})} & & \cdots & & B_{N_\mathrm{p}-N_\mathrm{u}+2}^{(N_\mathrm{p})} & A_{N_\mathrm{p}-N_\mathrm{u}+1}^{(N_\mathrm{p})} & \sum_{i=1}^{N_\mathrm{p}-N_\mathrm{u}+1} B_i^{(N_\mathrm{p})} & A_{N_\mathrm{p}-N_\mathrm{u}}^{(N_\mathrm{p})} & A_{N_\mathrm{p}-N_\mathrm{u}-1}^{(N_\mathrm{p})} & \cdots & & A_1^{(N_\mathrm{p})} & A_0^{(N_\mathrm{p})}
\end{array}\right]$$

$$z = \{u_k, y_{k+1}, \ldots, u_{k+N_u-1}, y_{k+N_u}, y_{k+N_u+1}, \ldots, y_{k+N}\}$$

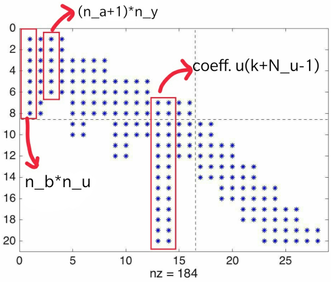
Structure depends on the ordering of decision variables, model (n_a, n_b, n_u, n_y) and tuning parameters (N, N_u)

BVLS for MPC: Problem Sparsity

Problem: Least-squares with box constraints

$$\min_{p \le x \le q} \frac{1}{2}\|\boldsymbol{J}x - b\|_2^2$$

Structure depends on the ordering of decision variables, model and tuning parameters



Sparsity pattern of $\nabla_z h_k(z)$ for a random model with $N_\mathrm{p} = 10$, $N_\mathrm{u} = 4$, $n_\mathrm{a} = 2$, $n_\mathrm{b} = 4$, $n_u = 2$ and $n_y = 2$.

Build-free MPC

- Typical MPC setup:
  - Step 1: Construct an optimization problem based on the prediction model and tuning parameters (e.g., N, N_u)
  - Step 2: Pass it in *standard* form to an optimization solver
- Constructing optimization problem matrices can be time consuming, especially in approaches with *condensed* formulation
- A change in the model coefficients, horizons, tuning weights, model size, needs re-construction of the optimization problem
- We propose methods that systematically eliminate the problem construction phase, resulting in:
  - reduced memory requirement
  - faster execution
  - ability to adapt to changes in model/tuning parameters at runtime at no computational cost

Key Idea

Parameterize the optimization algorithm in terms of model and tuning parameters

Matrix abstraction

BVLS problem: Least-squares with box constraints

$$\min_{p\leq x\leq q} \frac{1}{2}\|\boldsymbol{J}x - b\|_2^2$$

- The sparse Jacobian matrix $J = \begin{bmatrix} W_k \\ \nabla_z h_k(z_k, \phi_k)^\top \end{bmatrix}$ contains tuning weights and coefficients from the sequence of affine models with indexing completely defined by model and tuning parameters
- Role of J in BVLS:
  - Solve unconstrained LS: $J(:, \neg\mathcal{A})^\dagger\,(b - J(:, \mathcal{A})x(\mathcal{A}))$
  - Gradient entries: $J(:, \mathcal{A})^\top(b - Jx)$
- **Key observation**: all operations with J can be replaced by 2 abstract operators
  1) `Jix`: return ith column of J times a given scalar `x`
  2) `JtiX`: return ith column of J times a given vector `X`
  (ith column of J = ith row of $J^\top$)

Two operators to replace all J instances

- To code `Jix` and `JtiX` we need

  1) Model coefficients from the sequence of affine models: store all in a single vector M (faster execution) **or** generate online via linearization routines (lower memory)

  2) Model parameters n_a, n_b, n_u, n_y

  3) Tuning parameters N, N_u and weights
- For J times a vector v, use `Jix` over each element of v and accumulate the result
- For $J^\top$ times a vector v, use `JtiX` over each element of v and store result in the corresponding element of output vector
- Recall: location of non-zeros is already known in terms of model and tuning parameters!

  $\implies$ Only non-zero entries in J are operated

  $\implies$ Matrix operations as fast as sparse linear algebra while using significantly lesser amount of memory!

Build-free MPC algorithm

BVNLLS without problem construction

Initialize $z^{(0)} \in \{z|p \leq z \leq q\}$, $j \leftarrow 0$

1: ~~Update Jacobian~~ $\cancel{J \leftarrow \nabla_z r\left(z^{(j)}\right)^\top}$ Update M; (Linearization)

2: Compute gradient $J^\top r$ of Lagrangian function; Use `JtiX`

3: If first-order optimality conditions are satisfied then **stop**;

4: Solve $\Delta \leftarrow \arg \min_{p-z^{(j)} \leq \Delta \leq q-z^{(j)}} \|\cancel{J}\Delta + r\left(z^{(j)}\right)\|_2^2$ via **BVLS solver**; Uses `Jix`, `JtiX`

5: Compute step-size $0 < \alpha \leq 1$ for backtracking;

6: $z^{(j+1)} \leftarrow z^{(j)} + \alpha\Delta$; (Update iterate)

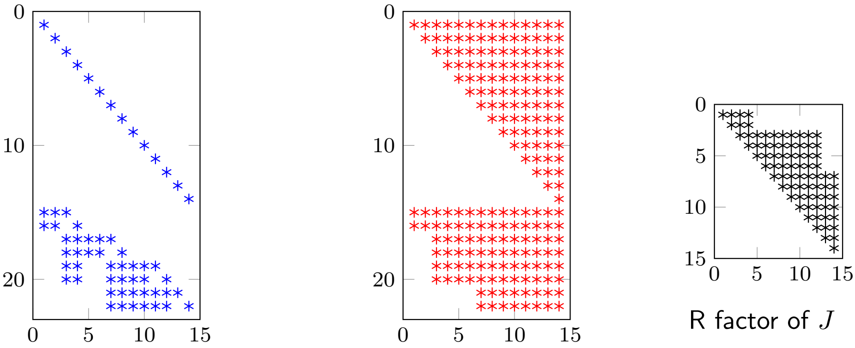
7: $j \leftarrow j+1$; **go to** Step 1;

Code of `Jix` and `JtiX` does not change with any change in model or tuning parameters or problem size

$\implies$ entire MPC code is stand-alone for a given problem formulation

Sparse matrix factors

- Recall: we solve $J(:,\neg\mathcal{A})^\dagger\,(b - J(:,\mathcal{A})x(\mathcal{A}))$ in each BVLS iteration
- Best way: **recursive thin QR factorization** (using Gram-Schmidt orthogonalization)
- How to exploit sparsity? How to know the location of non-zeros in QR?



R factor of J

Sparse matrix $J = QR$ Q factor of J

Sparsity pattern of Jacobian J and its thin QR factors for a random NARX model with diagonal weights, and parameters $n_y = n_u = 2$, $n_\mathrm{a} = 2$, $n_\mathrm{b} = 1$, $N = 4$, $N_u = 3$.

Gram-Schmidt orthogonalization

- If $J = QR$,

$$
\begin{aligned}
Q'(:,i) &= J(:,i) - \sum_{j=1}^{i-1} Q(:,j)Q(:,j)^\top J(:,i), \\
Q(:,i) &= Q'(:,i) / \left\| Q(:,i)' \right\|_2 . \\
R(j,i) &= Q(:,j)^\top J(:,i), \forall j \in [1, i-1], \\
R(i,i) &= \left\| Q(:,i)' \right\|_2
\end{aligned}
$$

- Above formulae refer to **classical** Gram-Schmidt process: catastrophic numerical cancellation possible
- We use the theoretically equivalent **modified** Gram-Schmidt method with **automatic reorthogonalization** for numerical stability

Sparsity analysis

- Define the non-zero structure of a vector x to be the set of indices $\mathcal{S}(x)$ such that $x(i) \neq 0$, $\forall i \in \mathcal{S}(x)$, and $x(j) = 0$, $\forall j \notin \mathcal{S}(x)$.

Theorem: Non-zero structure of columns of Q factor

Consider an arbitrary sparse matrix $J \in \mathbb{R}^{n_1 \times n_2}$ of full rank such that $n_1 \geq n_2$ and let Q denote the Q-factor from its thin QR factorization i.e., $J = QR$. The non-zero structure of each column $Q(:,i)$ of Q satisfies

$$\mathcal{S}\left(Q(:,i)\right) \subseteq \bigcup_{j=1}^{i} \mathcal{S}\left(J(:,j)\right), \forall i \in [1, n_2],$$

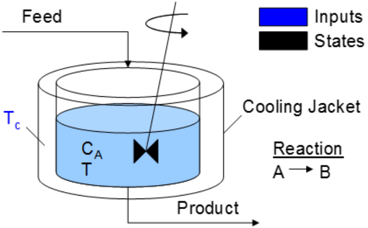
and $\mathcal{S}\left(Q(:,1)\right) = \mathcal{S}\left(J(:,1)\right)$.

- Using the above theorem, which is based on MGS, and its corollaries [4], we exploit sparsity without even storing J!
(paper [4] = Saraf, Bemporad 2019 available on arXiv)

Recursive QR updates

- With diagonal weights, sparsity pattern info of Q factor can be stored in just 2 integer vectors
- Recursive update in sparsity pattern $\implies$ update entries in 2 vectors of dimension $=$ no. of columns of J
- Main principle: thin QR factorization of a matrix is unique
  $\implies$ column indices of J may be added to or removed from the active set in an arbitrary order!
- R factor's sparsity exploited using orthogonality: $R = Q^\top J$

Numerical example: NMPC of CSTR



Continuous Stirred Tank Reactor (CSTR) [1]

- **CSTR** model

$$T^{(k+1)} = T^{(k)} + t_s(T_f^{(k)} - 1.3T^{(k)} + \kappa_1 C_A^{(k)} e^{\frac{-5963.6}{T^{(k)}}} + 0.3{T_C}^{(k)})$$

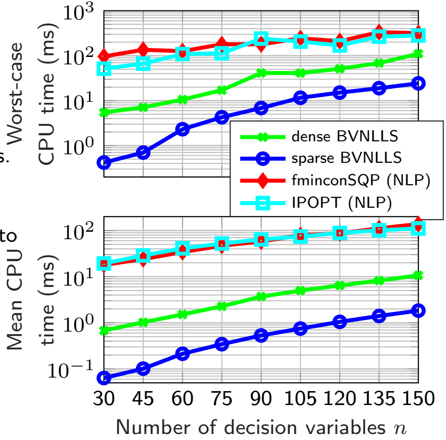
$$C_A^{(k+1)} = C_A^{(k)} + t_s(C_{Af}^{(k)} - \kappa_2 C_A^{(k)} e^{\frac{-5963.6}{T^{(k)}}} - C_A^{(k)})$$

- Nonlinear system with 2 outputs, 1 input and 2 measured disturbances
- Model coefficients as in MPC Toolbox demo (Mathworks)

[1] retrieved from apmonitor.com

Execution time: Small-sized problems

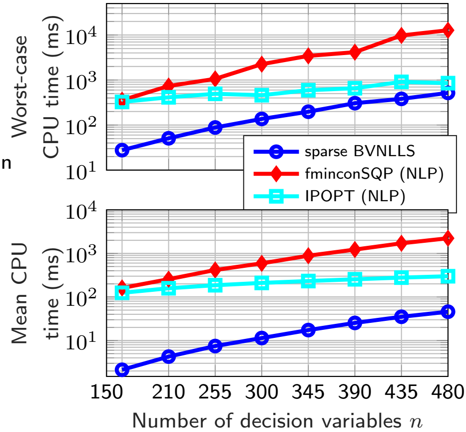
- SQP: subproblems may be infeasible. Warm start exploited (MATLAB `fmincon`)
- IPOPT: uses `MA57` solver, sparse routines. No warm start exploited
- BVNLLS: very few Gauss-Newton steps to converge, exploits warmstarts
- Sparse linear algebra (LA) is typically slower than dense LA for small problems
  - proposed sparsity exploiting methods allow $\approx 10\times$ faster solution than dense variant even for small problem sizes!
  - $\approx 100\times$ speedup on mean CPU time w.r.t. benchmarks



Solver comparison in MATLAB for NMPC of CSTR simulated on a Mac with 2.6GHz Intel Core i5. $N = n/3$, no. of box constraints $= n$ pairs, no. of equality constraints $= 2N$, $\sqrt{\rho} = 10^4$, $N_{\text{sim}} = 1500$ sample steps.

Execution time: Larger problems

- Active-set methods can be faster than interior-point methods if sparsity is exploited [1]
- Faster even for large problems
- sparse BVNLLS tool for NMPC
  - Efficient C implementation
  - easily embeddable



Solver comparison in MATLAB for NMPC of CSTR simulated on a Mac with 2.6GHz Intel Core i5. $N = n/3$, no. of box constraints $= n$ pairs, no. of equality constraints $= 2N$, $\sqrt{\rho} = 10^4$, $N_{\text{sim}} = 1500$ sample steps.

Conclusions

- **Key ideas**:
  - relax equality constraints due to dynamics using penalty functions
  - parameterize optimization solver in terms of MPC parameters
- The proposed optimization solvers for MPC are:
  - simple to code, fast to execute, flexible in real time
  - good for embedded platforms (PLCs, μcontrollers, ...)
  - competitive with state-of-the-art algorithms
- Novel linear algebra methods devised to heavily exploit sparsity
- Unifying MPC framework for LTI/LPV/NLTI/NLPV systems
- Linearization step can be code-generated using symbolic math software, no other code generation required - easy deployability!
- Extensions: Matrix-free MPC, more general problems

Key References

J. Nocedal and S. Wright
Numerical Optimization.
Springer, 2006.

P. Stark and R. Parker
Bounded-Variable Least-Squares: An Algorithm and Applications.
Computational Statistics, 10: 129–141, 1995.

N. Saraf and A. Bemporad
A bounded-variable least-squares solver based on stable QR updates.
IEEE Transactions on Automatic Control, 2019.

N. Saraf and A. Bemporad
An efficient non-condensed approach for linear and nonlinear model predictive control with bounded variables.
Preprint available at arXiv.org, 2019.