

Range Computation of Polynomial Problems using the Bernstein Form

by

Prof. P.S.V. Nataraj
IDP in Systems and Control Engineering
IIT Bombay

Outline

- Introduction
- Bernstein form
- Degree elevation of the Bernstein form
- Vertex property
- Subdivision of the Bernstein form
- Numerical example for practice

Introduction

- ★ Optimization: is study of how to find the best (optimum) solution to a problem.
- ★ Objective Function: mathematical representation of a problem.

e.g. minimum function problem, which is expressed as

$$\min_x f(x)$$

where f is the objective function

Cont...

★ Global and Local Optima

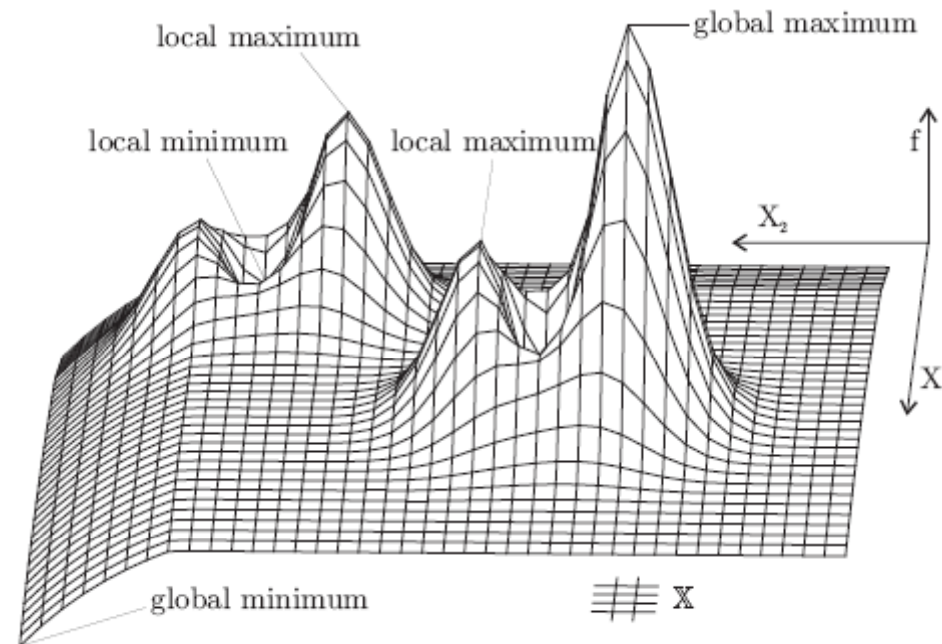
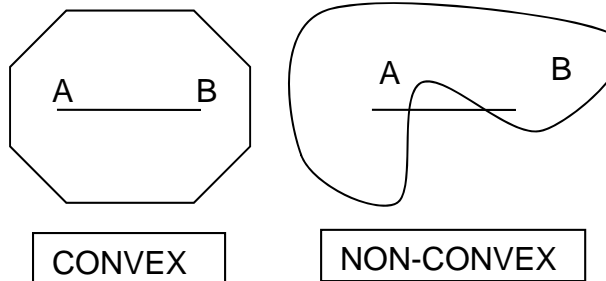


Fig.: Global and local optima of a two-dimensional function

Cont...

- ★ Convex – Non Convex Function:

Convex if, every point on the straight line segment that joins them is within the set.

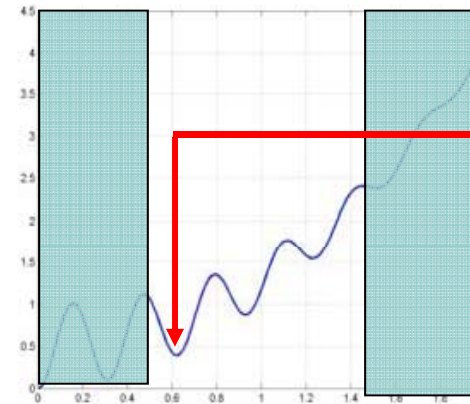


Cont...

☆ Unconstrained - Constrained Problems



Minimum is zero



Minimum is here

☆ Univariate/Bivariate/Multivariate Problems

Single Variable Objective Function = Univariate

Two Variable Objective Function = Bivariate

More than one Variable Objective Function = Multivariate

Introduction

- Various qualitative decision issues (*min. cost*, *max. profit*, etc), from science and engineering can be perceived as optimization problems.
- General optimization problem formulation is

$$\min_x f(x)$$

$$\text{s.t. } h_i(x) = 0, \quad i = 1, 2, \dots, m$$

$$g_j(x) \leq 0, \quad j = 1, 2, \dots, n$$

- Minimize above problem *globally*

Cont...

- Many optimization problem can be reduced to the problem of computing the *sharp range* of polynomials in several variables on box-like domains.
- We solve the problem of finding the sharp range which *encloses* global *minimum* using the Bernstein form of polynomials.
- The Bernstein coefficients of the expansion provide the lower and upper bounds for the range of the polynomial.
- We can perform subdivision of the original box for faster convergence of the range.

Bernstein Form

- Consider the n^{th} degree polynomial \mathbf{p} in a single variable $x \in U = [0,1]$

$$p(x) = \sum_{i=0}^n a_i x^i$$

- Bernstein form of order \mathbf{k} is

$$p(x) = \sum_{j=0}^k b_j^k B_j^k(x) \ , \quad k \geq n$$

- $B_j^k(x)$ are the Bernstein basis polynomials of degree \mathbf{k}

Cont...

- b_j^k are the Bernstein coefficients

$$b_j^k = \sum_{i=0}^j a_j \frac{\binom{j}{i}}{\binom{k}{i}}$$

- The unit interval is not really a restriction as any finite interval **X** can be linearly transformed to it.

Properties of Bernstein Coefficients

The range enclosure property of the Bernstein Form

- The Bernstein coefficients provide bounds for range **p** of over **$\mathbf{U}=[0,1]$** .
- Lemma 1 (**Range lemma**) (Cargo and Shisha, 1966): The range $\bar{p}([0,1])$ is bounded by the Bernstein coefficients as:

$$\bar{p}([0,1]) \subseteq \left[\min_j b_j^k, \max_j b_j^k \right]$$

- Convex hull property:

$$\text{conv}\{(x, p(x))\} \subseteq \text{conv}\{(I / N, b_I(\mathbf{U})) : I \in S_0\}$$

where $S_0 = \{0, n_1\} \times \{0, n_2\} \times \dots \times \{0, n_l\}$

Cont...

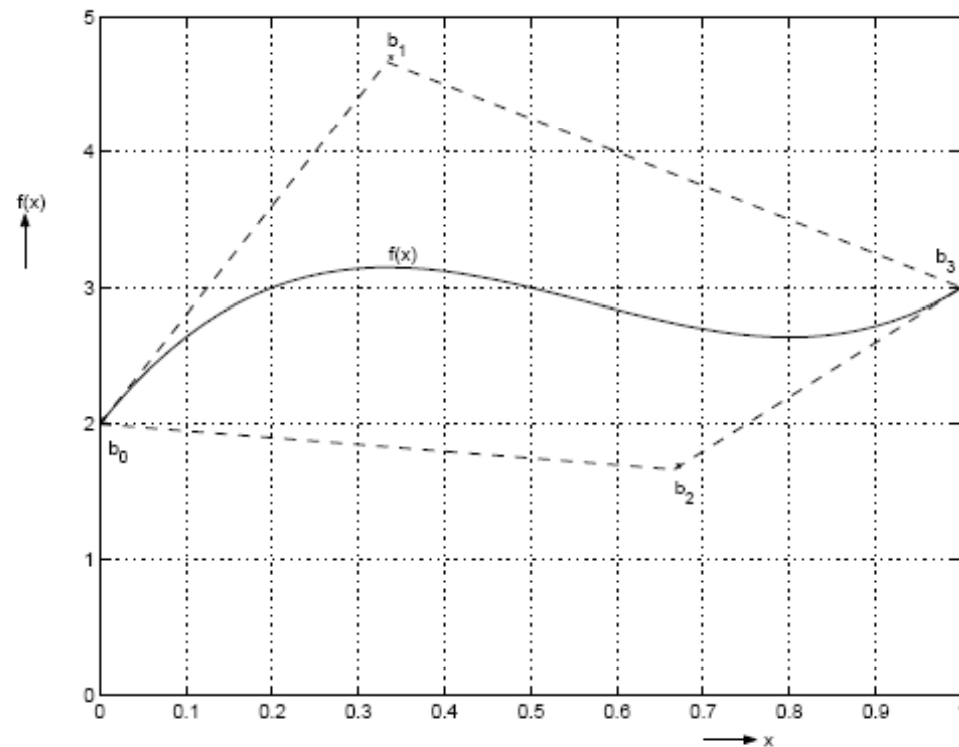


Figure : The polynomial function, its Bernstein coefficients, and the convex hull

Illustration

To illustrate the Bernstein approach for bounding the ranges of polynomials consider the simple polynomial

$$p(x) = x(1 - x)$$

whose range $\bar{p}([0, 1])$ is $[0, \frac{1}{4}]$.

- In the Bernstein approach, put polynomial in standard sums of power form

$$p(x) = \sum_{i=0}^n a_i x_i$$

where

$$n = 2, a_0 = 0, a_1 = 1, a_2 = -1$$

Cont...

- For $k = 2$ this gives

$$b_0^2 = 0, \quad b_1^2 = \frac{1}{2}, \quad b_2^2 = 0$$

so that

$$\min_j b_j^2 = 0, \quad \max_j b_j^2 = \frac{1}{2}$$

- Range lemma implies

$$\bar{p}([0, 1]) \subseteq \left[0, \frac{1}{2}\right]$$

Degree Elevation of Bernstein Form

- Improved Accuracy through Degree Elevation
- Tighter bounds on $\bar{p}([0, 1])$ can be obtained by elevating the degree k of the Bernstein form.
- For $k = 3$, we have

$$b_0^3 = 0, \quad b_1^3 = \frac{1}{3}, \quad b_2^3 = \frac{2}{3} \left(1 - \frac{1}{2}\right) = \frac{1}{3}, \quad b_3^3 = 0$$

and

$$\min_j b_j^3 = 0, \quad \max_j b_j^3 = \frac{1}{3}$$

Cont...

- By range lemma,

$$\bar{p}([0, 1]) \subseteq \left[0, \frac{1}{3}\right]$$

- Ratschek and Rokne prove that the range overestimation can be made small as the degree k is elevated:

$$w\left(\left[\min_j b_j^k, \max_j b_j^k\right]\right) - w(\bar{p}([0, 1])) \leq \frac{A}{k}$$

where

$$A = 2 \sum_{s=2}^n (s-1)^2 \left| p^{(s)}(0) \right| / s!$$

with $p^{(s)}$ denoting the s th derivative of p .

Cont...

- Degree of polynomial stays fixed at n .
 - Use the Bernstein forms of higher degree $k \geq n$.
 - Even get convergence of $[\min_j b_j^k, \max_j b_j^k]$ to the range $\bar{p}([0, 1])$.
- In general, for a given k the computation takes about k^2 arithmetic operations.
- No evaluations of given polynomial are required.

Degree Elevation Analysis

- Range enclosures obtained with Bernstein form of various degrees.

Degree k	Range Enclosure	index j for min b_j^k	index j for max b_j^k	Range overestimation
2	$[0, 0.5]$	0	1	0.2500
3	$[0, \frac{1}{3}]$	0	1	0.0833
4	$[0, \frac{1}{3}]$	0	2	0.0833
5	$[0, 0.3]$	0	2	0.0500
6	$[0, 0.3]$	0	3	0.0500
7	$[0, 0.2857]$	0	3	0.0357
10	$[0, 0.2778]$	0	5	0.0278
20	$[0, 0.2632]$	0	10	0.0132
30	$[0, 0.2586]$	0	15	0.0086
100	$[0, 0.2525]$	0	50	0.0025
1000	$[0, 0.2503]$	0	500	0.00025

Cont...

- As convergence is linear in degree k , degree elevation is not really efficient.

Vertex Property of Bernstein Form

- Remarkable feature: Bernstein form provides us with a criterion to indicate if calculated estimation is range or not.
- Cargo and Shisha (1966) give such a criterion based on the *vertex* property.
- The *upper bound* or *lower bound* is sharp if and only if $\min b_j^k(\mathbf{U})_{I \in S_0}$ (resp. $\max b_j^k(\mathbf{U})_{I \in S_0}$) is attained at the indices of vertices of Bernstein coefficient array ($B(\mathbf{U})$).

Cont...

Lemma 2 (**Vertex lemma**)

$$\bar{p}([0, 1]) = \left[\min_j b_j^k, \max_j b_j^k \right]$$

if and only if

$$\min_j b_j^k = \min \{b_0^k, b_k^k\}$$

and

$$\max_j b_j^k = \max \{b_0^k, b_k^k\}$$

- Vertex lemma also holds for any subinterval of $[0, 1]$.

Illustration

- Consider again the simple polynomial

$$p(x) = x(1 - x)$$

whose range $\bar{p}([0, 1])$ is $[0, \frac{1}{4}]$.

- For $k = 4$, Bernstein coefficients are

$$b_0^4 = 0, \quad b_1^4 = \frac{1}{4}, \quad b_2^4 = \frac{1}{3}, \quad b_3^4 = \frac{1}{4}, \quad b_4^4 = 0$$

- Range lemma gives

$$\bar{p}([0, 1]) \subseteq \left[0, \frac{1}{3}\right]$$

Cont...

- Check if above enclosure is the range itself or not.
- How? Apply Vertex lemma
 - Minimum Bernstein coefficient is b_0^4 or b_4^4 - occurs at vertices $j \in \{0, 4\}$.
 - Maximum Bernstein coefficient is b_2^4 , occurs at $j = 2$ that is not a vertex.
 - Vertex lemma is satisfied for the minimum,
 - Vertex lemma is not satisfied for the maximum - as $\max_j b_j^k \neq \max \{b_0^4, b_4^4\}$.
 - So, by vertex lemma, above enclosure is *not* the range.

Cont...

- Now, we check if any of the range enclosures obtained in previous table for elevated degree of Bernstein form is range or not.
- Table is reproduced below.

Degree k	Range Enclosure	index j for $\min b_j^k$	index j for $\max b_j^k$	Range overestimation
2	$[0, 0.5]$	0	1	0.2500
3	$[0, \frac{1}{3}]$	0	1	0.0833
4	$[0, \frac{1}{3}]$	0	2	0.0833
5	$[0, 0.3]$	0	2	0.0500
6	$[0, 0.3]$	0	3	0.0500
7	$[0, 0.2857]$	0	3	0.0357
10	$[0, 0.2778]$	0	5	0.0278
20	$[0, 0.2632]$	0	10	0.0132
30	$[0, 0.2586]$	0	15	0.0086
100	$[0, 0.2525]$	0	50	0.0025
1000	$[0, 0.2503]$	0	500	0.00025

Cont...

- We find from the table that for any k , the index j for $\max b_j^k$ (in column 4) is not from the vertex set $\{0, k\}$.
- By vertex lemma, *none* of the enclosures in column 2 is the range !

Subdivision procedure

- *Tightening* of bounds is possible by subdivision.
- Bernstein coefficients of the subdivided boxes can be computed from the Bernstein coefficients of the original box.
- Thus avoids the repeated computation of Bernstein coefficients of the function.
- Subdivision direction can be selected based on any one of the existing subdivision *direction selection* rules (T. Csendes and D. Ratz, 1997).
- Simplest one is subdivide along direction of *maximum width*

$$y(r) = \max(w(d))$$

where d box to be subdivided, and r is a direction in which it is subdivided.

Subdivision of Bernstein Form

- A generally more efficient approach than degree elevation of the Bernstein form is subdivision.
- Let $\mathbf{D} = [\underline{d}, \bar{d}] \subseteq \mathbf{U}$ and assume we have already the Bernstein coefficients on \mathbf{D} .
- Suppose \mathbf{D} is bisected to produce two subintervals \mathbf{D}_A and \mathbf{D}_B given by

$$\mathbf{D}_A = [\underline{d}, m(\mathbf{D})]; \mathbf{D}_B = [m(\mathbf{D}), \bar{d}]$$

Cont...

- Then, the Bernstein coefficients on the subintervals \mathbf{D}_A and \mathbf{D}_B can be obtained from those on \mathbf{D} , by executing the following algorithm.

Subdivision Algorithm

- Inputs: The interval $D \subseteq U$ and its Bernstein coefficients (b_j^k).
- Outputs: Subintervals D_A and D_B and their Bernstein coefficients \tilde{b}_j^k and \hat{b}_j^k

START

- Bisect D to produce the two subintervals D_A and D_B .
- Compute the Bernstein coefficients on subinterval D_A as follows.
 - Set: $b_j^k \leftarrow \bar{b}_j^k$, for $j = 0, 1, \dots, k$
 - For $i = 1, 2, \dots, k$ DO

$$b_j^k = \begin{cases} b_j^{i-1} & \text{for } j < i \\ \frac{1}{2} \{ b_{j-1}^{i-1} + b_j^{i-1} \} & \text{for } j \geq i \end{cases}$$

Cont...

To obtain the new coefficients apply formula in (b) for $j=0,1,\dots,k$.

- Find the Bernstein coefficients on subinterval D_A as

$$\tilde{b}_j^k = b_j^k, \quad \text{for } j=0,1,\dots,k$$

- Find the Bernstein coefficients on subinterval D_B from intermediate values in above step, as follows.

$$\hat{b}_j^k = b_k^j, \quad \text{for } j=0,1,\dots,k$$

- Return D_A, D_B and the associated Bernstein coefficients \tilde{b}_j^k and \hat{b}_j^k .
END

Illustration

- Let us run through Algorithm Subdivision for Example 1.
- For $k = 4$, we have already the Bernstein coefficients \bar{b}_j^k for the interval $\mathbf{D} = [0, 1]$.
- With these as the inputs to Algorithm subdivision, the results at the various steps are

Cont...

- step 1: \mathbf{D} is bisected to produce two subintervals $\mathbf{D}_A = [0, 0.5]$ and $\mathbf{D}_B = [0.5, 1]$.

- step 2: The Bernstein coefficients on subinterval \mathbf{D}_A are computed as follows.

– step 2a: Set : $b_j^0 \leftarrow \bar{b}_j^4$, for $j = 0, \dots, 4$,

$$b_0^0 = \bar{b}_0^4 = 0;$$

$$b_1^0 = \bar{b}_1^4 = \frac{1}{4};$$

$$b_2^0 = \bar{b}_2^4 = \frac{1}{3};$$

$$b_3^0 = \bar{b}_3^4 = \frac{1}{4};$$

$$b_4^0 = \bar{b}_4^4 = 0$$

Cont...

– step 2b:

* for $i = 1$:

$$b_0^1 = b_0^0 = 0$$

$$b_1^1 = \frac{1}{2} (b_0^0 + b_1^0) = \frac{1}{2} \left(0 + \frac{1}{4} \right) = \frac{1}{8}$$

$$b_2^1 = \frac{1}{2} (b_1^0 + b_2^0) = \frac{1}{2} \left(\frac{1}{4} + \frac{1}{3} \right) = \frac{7}{24}$$

$$b_3^1 = \frac{1}{2} (b_2^0 + b_3^0) = \frac{1}{2} \left(\frac{1}{3} + \frac{1}{4} \right) = \frac{7}{24}$$

$$b_4^1 = \frac{1}{2} (b_3^0 + b_4^0) = \frac{1}{2} \left(\frac{1}{4} + 0 \right) = \frac{1}{8}$$

Cont...

* for $i = 2$:

$$b_0^2 = b_0^1 = 0$$

$$b_1^2 = b_1^1 = \frac{1}{8}$$

$$b_2^2 = \frac{1}{2} (b_1^1 + b_2^1) = \frac{1}{2} \left(\frac{1}{8} + \frac{7}{24} \right) = \frac{10}{48}$$

$$b_3^2 = \frac{1}{2} (b_2^1 + b_3^1) = \frac{1}{2} \left(\frac{7}{24} + \frac{7}{24} \right) = \frac{7}{24}$$

$$b_4^2 = \frac{1}{2} (b_3^1 + b_4^1) = \frac{1}{2} \left(\frac{7}{24} + \frac{1}{8} \right) = \frac{10}{48}$$

Cont...

* for $i = 3$:

$$b_0^3 = b_0^2 = 0$$

$$b_1^3 = b_1^2 = \frac{1}{8}$$

$$b_2^3 = b_2^2 = \frac{10}{48}$$

$$b_3^3 = \frac{1}{2} (b_2^2 + b_3^2) = \frac{1}{2} \left(\frac{10}{48} + \frac{7}{24} \right) = \frac{1}{4}$$

$$b_4^3 = \frac{1}{2} (b_3^2 + b_4^2) = \frac{1}{2} \left(\frac{7}{24} + \frac{10}{48} \right) = \frac{1}{4}$$

Cont...

* for $i = 4$:

$$b_0^4 = b_0^3 = 0$$

$$b_1^4 = b_1^3 = \frac{1}{8}$$

$$b_2^4 = b_2^3 = \frac{10}{48}$$

$$b_3^4 = b_3^3 = \frac{1}{4}$$

$$b_4^4 = \frac{1}{2} (b_3^3 + b_4^3) = \frac{1}{2} \left(\frac{1}{4} + \frac{1}{4} \right) = \frac{1}{4}$$

Cont...

- Step 2c: The Bernstein coefficients on the subinterval \mathbf{D}_A are

$$\begin{aligned}\tilde{b}_0^4 &= b_0^4 = 0; & \tilde{b}_1^4 &= b_1^4 = \frac{1}{8}; & \tilde{b}_2^4 &= b_2^4 = \frac{10}{48} \\ \tilde{b}_3^4 &= b_3^4 = \frac{1}{4}; & \tilde{b}_4^4 &= b_4^4 = \frac{1}{4}\end{aligned}$$

- step 3: The Bernstein coefficients on the neighboring subinterval \mathbf{D}_B are

$$\begin{aligned}\hat{b}_0^4 &= b_4^0 = 0; & \hat{b}_1^4 &= b_4^1 = \frac{1}{8}; & \hat{b}_2^4 &= b_4^2 = \frac{10}{48}; \\ \hat{b}_3^4 &= b_4^3 = \frac{1}{4}; & \hat{b}_4^4 &= b_4^4 = \frac{1}{4}\end{aligned}$$

Cont...

- step 4: Finally,
 - For subinterval \mathbf{D}_A , Bernstein coefficients are
$$\left(0, \frac{1}{8}, \frac{10}{48}, \frac{1}{4}, \frac{1}{4}\right)$$
 - For subinterval \mathbf{D}_B , Bernstein coefficients are
$$\left(0, \frac{1}{8}, \frac{10}{48}, \frac{1}{4}, \frac{1}{4}\right)$$
 - It is coincidental here that Bernstein coefficients for both the subintervals are the same.

Cont...

- By range lemma

$$\bar{p}(\mathbf{D}_A) \subseteq \left[0, \frac{1}{4}\right]$$

$$\bar{p}(\mathbf{D}_B) \subseteq \left[0, \frac{1}{4}\right]$$

Bernstein Subdivision

- Consider the Bernstein coefficients given a few slides earlier.
- For subinterval \mathbf{D}_A ,
 - The minimum Bernstein coefficient is \tilde{b}_0^4
 - The maximum Bernstein coefficient is \hat{b}_4^4 .
- Both these occur at the vertices, i.e., for $j \in \{0, 4\}$.
- By the vertex lemma, the range of $\bar{p}(\mathbf{D}_A)$ is $[0, \frac{1}{4}]$.

Cont...

- An identical situation holds for other subinterval \mathbf{D}_B .
- Thus, we obtain the range $\bar{p}([0, 1]) = [0, \frac{1}{4}]$.
- In this example, using just *one* subdivision and application of the vertex lemma to the subintervals, we have been able to obtain the range of the given polynomial.
- We are also able to *assert* that obtained enclosure is indeed the range.

Cont...

- It was not possible to get the range through degree elevation, even with Bernstein form of as high a degree as $k = 1000$.
- From Table 1, this high degree Bernstein form still produced an overestimation of about $2.5e - 04$!

Numerical Examples

- Consider the simple polynomial as

$$p(x) = x^2 - x = \sum_{i=0}^n a_i x_i$$

whose range $\bar{p}([0,1])$ is $[-0.25, 0]$.

- Here $n = 2$ $a_0 = 0$, $a_1 = -1$, $a_2 = 1$

- For $k=2$ this gives

$$b_0^2 = 0, \quad b_1^2 = -0.5, \quad b_2^2 = 0$$

so that

$$\min_j b_j^2 = -0.5, \quad \max_j b_j^2 = 0$$

Cont...

- Range lemma implies

$$\bar{p}([0,1]) \subseteq [-0.5, 0]$$

Apply Subdivision Algorithm

- Let us run through Algorithm Subdivision for Example 1.
- For $k = 2$ we have already the Bernstein coefficients \bar{b}_j^k for the interval $\mathbf{D} = [0, 1]$.
- With these as the inputs to Algorithm subdivision, the results at the various steps are

Cont...

- Bernstein coefficients on subinterval D_A are

$$\tilde{b}_0^2 = 0, \quad \tilde{b}_1^2 = -0.25, \quad \tilde{b}_2^2 = -0.25$$

- Bernstein coefficients on subinterval D_B are

$$\tilde{b}_0^2 = -0.25, \quad \tilde{b}_1^2 = -0.25, \quad \tilde{b}_2^2 = 0$$

- Therefore by range lemma, we have

$$\bar{p}(D_A) \subseteq [-0.25, 0]$$

$$\bar{p}(D_B) \subseteq [-0.25, 0]$$

Subdivision of Bernstein Coefficients

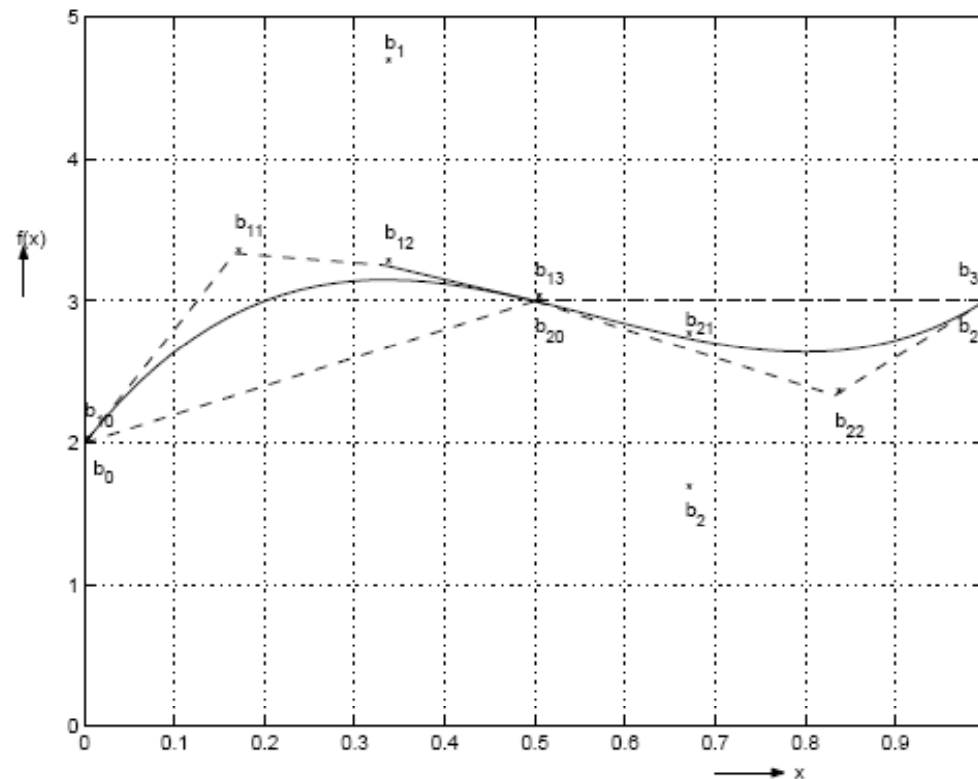


Figure : Improvement in range enclosure with subdivision



Cut-off test

- In global minimization algorithms, cut-off test is used to *delete boxes*, which are sure not to contain the global minima.
- The usual procedure is to assign the maximum value of Bernstein coefficients of the objective function as initial cut-off test value.
- Any box whose minimum Bernstein coefficients value is greater than this will be deleted.
- If the maximum Bernstein coefficients value of any box is lesser than the present cut-off test value, the cut-off test value is updated using this value.

Global optimization of MINLPs: the Bernstein polynomial approach

Prof. P.S.V. Nataraj

Bhagyesh V. Patil

Systems and Control Engineering Group

IIT Bombay

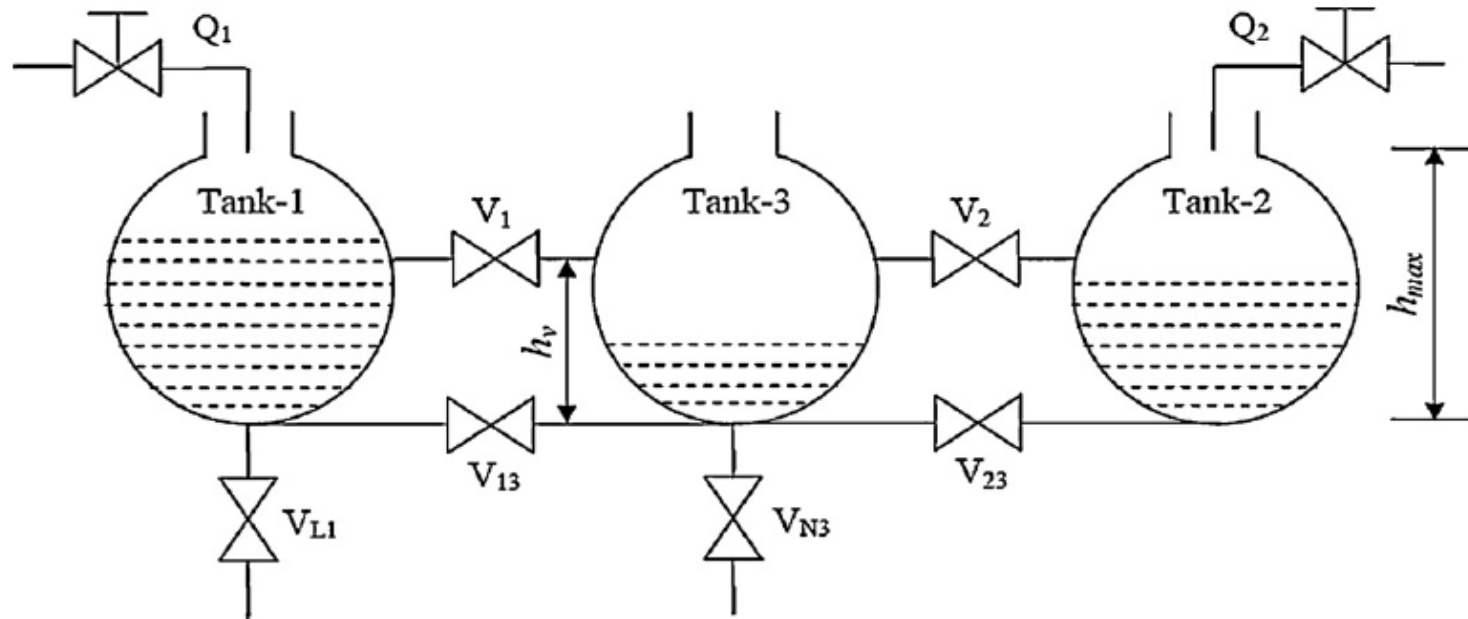
Outline

- ❖ Introduction
- ❖ Mathematical formulation
- ❖ Solution approaches and solvers
- ❖ Why need of an alternate approach?
- ❖ Bernstein global optimization algorithms
- ❖ Numerical experiments with state-of-the-art solvers
- ❖ Case studies
 - a) Spherical three-tank system
 - b) Trim-loss minimization problem

Introduction

- Various qualitative decision issues from science and engineering can be perceived as optimization problems.
- Minimizing cost during the production process, or maximizing the profit in the sale of some products.
- Min. and max. processes in practical optimization may sometime be subjected to some constraints.
- Minimize cost during the production process of tube lights such that at least 100 tube lights should be produced per hour.

Practical Example (Three-spherical tank system)



- Aim is to fill the tanks up to the desired level.
- Optimization task is to achieve above aim by minimizing error between actual plant output (y) and the reference trajectory (y_{ref}) and the penalizing control moves (u), subjected to following constraints:

$$y_{\max} \leq y \leq y_{\min}$$

$$u_{\max} \leq u \leq u_{\min}$$

- It has two inflow paths (Q_1, Q_2) to fill the tanks, and six solenoid (on/off) ($V_1, V_2, V_{13}, V_{23}, V_{L1}$ and V_{N3}) valves to control the flow between the intermediate tanks.

- Optimization problem formulation:

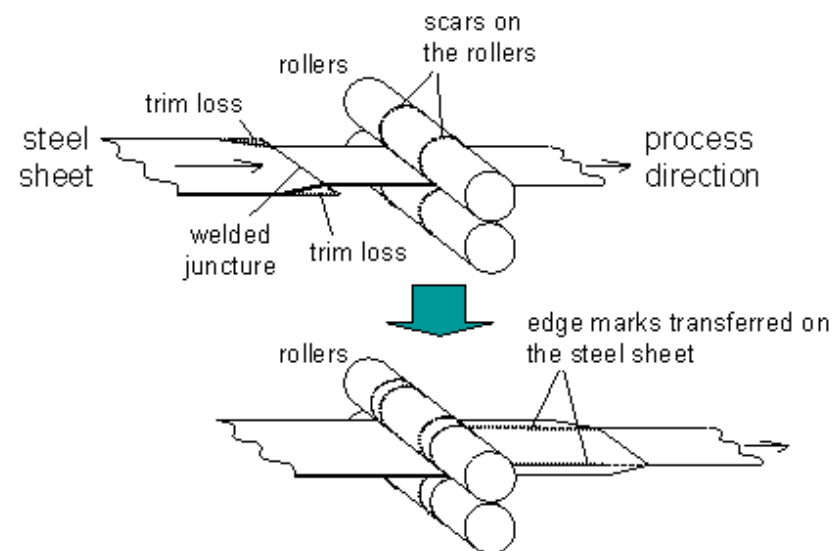
$$\min_{u_1, u_2, \dots, u_n} \sum_{i=1}^p (y_{k+i} - y_{ref}) w_y + \sum_{i=0}^{m-1} (u_{k+i} - u_{k+i-1}) w_u$$

u_1, u_2, \dots, u_n are the decision variables, w_y and w_u are penalizing weights, p and m are prediction and control horizons .

- $u_1, u_2 = \{Q_1, Q_2\}$ are *continuous* decision variables.
- $u_3, \dots, u_n = \{V_1, \dots, V_{N3}\}$ are *binary* (0/1) decision variables.
- Above formulation termed as *Mixed-Integer Nonlinear Programming MINLP*.

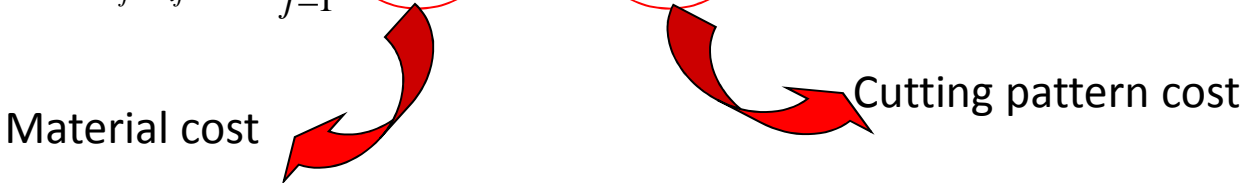
Practical Example (Trim-loss minimization)

- Common problem in the paper industry, glass industry.
- Aim is to cut different sized ordered product rolls from raw paper rolls to satisfy customer demands (Westerlund et al., 1998).
- Optimization task is to obtain *optimal cutting* pattern such that the cost function including the *material cost* and *cost* for changing *cutting* patterns is minimized.



- Optimization problem formulation:

$$x = \min_{m_j, n_{ij}, y} \sum_{j=1}^p c_j m_j + c_j y_j$$



- Decision variables are

m_j the multiple cutting pattern (integer in nature)

n_{ij} the number of product in the cutting pattern in y_j (*integer* in nature)

y_j the *binary* variable indicates cutting pattern j used or not

c_j cost of change of the cutting pattern

x total cost required to get optimal cutting pattern (*continuous*)

- Typical constraints are integer restrictions on cutting pattern, widths, etc.
- Above formulation termed as *Mixed-Integer Nonlinear Programming MINLP*.

Mathematical formulation

- MINLP is a optimization problem of the following form:

$$\begin{aligned} & \min_x f(x) \\ & \text{subject to } g(x) \leq 0 \\ & \quad h(x) = 0 \\ & \quad x_k \in X \subseteq \mathbb{R}, \quad k = 1, 2, \dots, l_d \\ & \quad x_k \in \{0, 1, \dots, q\} \subset \mathbb{Z}, \quad k = l_d + 1, \dots, l \\ & \quad q \geq 0, q \in \mathbb{Z} \end{aligned}$$

- We consider MINLPs, that are *polynomial* in nature (f , g and h are polynomial functions).
- Minimize above MINLP *globally*.

Solution approaches

- Generalized Benders Decomposition (Geoffrion, 1972).
- Branch-and-Bound (Gupta and Ravindran, 1985).
- Outer Approximation (Duran and Grossman, 1986).
- Branch-and-Cut (Padberg and Rinaldi, 1987).
- Extended Cutting Plane method (Westerlund and Pettersson, 1995).
- An LP/NLP-based-Branch-and-Bound (Quesada and Grossmann, 1992).
- Alpha Extended Cutting Plane (Westerlund and Pörn, 2002).

Remark:

State-of-the-art solvers generally use one of the above solution approaches.

Solvers available for solving MINLPs

Solver	Supports		Availability		Interfaces				NEOS server availability
	Convex	Nonconvex	Commercial	Open source	AIMMS	AMPL	GAMS	MATLAB	
<u>alphaBB</u>	✓	✓							
AlphaECP	✓		✓				✓		✓
AOA	✓			✓	✓				
BARON	✓	✓	✓		✓		✓		✓
BNB20	✓			✓				✓	
Bonmin	✓			✓		✓	✓		✓
<u>Couenne</u>	✓	✓		✓		✓			✓
DICOPT	✓		✓				✓		✓
<u>FilMINT</u>	✓		✓			✓			✓
<u>fminconset</u>	✓			✓				✓	
KNITRO	✓		✓		✓	✓			✓
<u>LaGO</u>	✓			✓		✓	✓		
LINDOBB	✓		✓				✓		
<u>LINDOGlobal</u>	✓	✓	✓				✓		✓
MILANO	✓			✓				✓	
MINLP	✓		✓			✓			✓
SBB	✓		✓				✓		✓

Background (Bernstein form)

- Consider the polynomial p in a single variable

$$p(x) = \sum_{i=0}^l a_i x^i, \quad a_i \in R$$

- Bernstein form of degree k is

$$p(x) = \sum_{i=0}^k b_i^k B_i^k(x), \quad k \geq l$$

$B_i^k(x)$ are the Bernstein basis polynomials of degree k .

- b_i^k are the Bernstein coefficients

$$b_i^k = \sum_{j=0}^i \frac{\binom{i}{j}}{\binom{l}{j}} a_j, \quad i = 0, 1, \dots, l$$

Properties of Bernstein coefficients

The range enclosure property of the Bernstein form

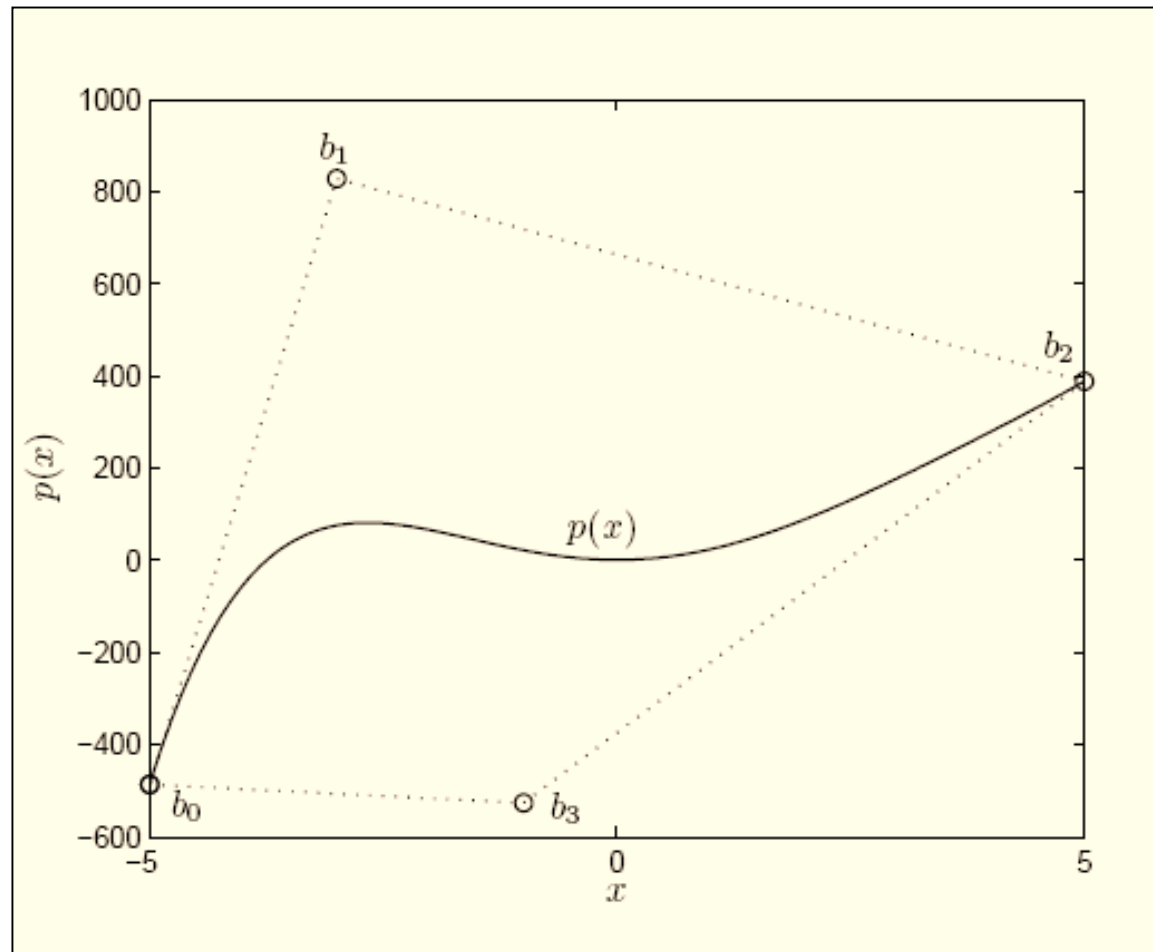
- The Bernstein coefficients provide bounds for range p of over $x=[0,1]$.
- (*Cargo and Shisha*, 1966): The range $\bar{p}([0,1])$ is bounded by the Bernstein coefficients as:

$$\bar{p}([0,1]) \subseteq \left[\min_i b_i^k, \max_i b_i^k \right]$$

- Convex hull property:

$$\left\{ \begin{pmatrix} x \\ p(x) \end{pmatrix} : x \in U \right\} \subseteq \text{conv} \left\{ \begin{pmatrix} \frac{i}{l} \\ b_i \end{pmatrix} : i = 0, \dots, l \right\}$$

Enclosure property of the Bernstein polynomials



The polynomial function, its Bernstein coefficients, and the convex hull.

Bernstein global optimization so far ...

- Taylor-Bernstein form to compute *range enclosures* of multivariate polynomials. (Kotecha and Nataraj, 2004)
- *Unconstrained* Bernstein global *optimization* algorithm to compute range enclosures of multivariate polynomials. (Ray and Nataraj, 2007)
- *Unconstrained and constrained global optimization* algorithms to compute the global minimum of multivariate polynomial NLPs. (Arounassalame and Nataraj, 2009).
- Constrained *global optimization* algorithms to compute the global minimum of multivariate polynomial *MINLPs* (Bhagyesh and Nataraj, 2007-2012 ongoing).
- Development of the *parallel* Bernstein global optimization algorithms using GPU *computing* (Dhabe and Nataraj, 2010-2012 ongoing).

Basic Bernstein constrained global optimization algorithm for NLPs

- Step 1. Compute the Bernstein coefficient arrays of the objective and constraint polynomials on \mathbf{y} .
- Step 2. Set $\tilde{p} : \infty$ and y as the min. Bernstein value over Bernstein coefficient arrays of the objective polynomial.
- Step 3. Set flag vector $R=(R_1, \dots, R_{m+n}) := 0$ (to indicate status of particular constraints).

Basic Bernstein constrained global optimization algorithm for NLPs

- Step 4. Initialize the list with the item as (\mathbf{y}, y, R) , solution list as empty.
- Step 5. Pick the first item from list deleting its entry. If list is empty go to step 11.
- Step 6. Subdivide the box y into two boxes, obtain Bernstein coefficients arrays of objective and constraint polynomials over them.
- Step 7. Test the constraint feasibility over the each box by checking the feasibility of the Bernstein coefficients of the constraint functions.

Basic Bernstein constrained global optimization algorithm for NLPs

- Step 8. If the box is infeasible, delete it. Else (\tilde{p} update) store the item (\mathbf{y}_k, y_k, R^k) in the list.
- Step 9. Discard all items from list whose current min. (y_k) greater than \tilde{p}
- Step 10. If the width of the box is within the prescribed accuracy, put the result in the solution list. Else go to step 5.
- Step 11. Analyze the solution list and return the global minimum and global minimizers.

Bernstein global optimization algorithm for MINLPs

Relaxation

- We use a *continuous relaxation* of the original problem (P), that arises when integrality restriction are relaxed on the integer variables.

$$\begin{aligned} & \min_x f(x) \\ & \text{subject to } g(x) \leq 0 \\ & h(x) = 0 \\ & x^L \leq x_k \in R^n \leq x^U, \quad k = 1, 2, \dots, n \end{aligned} \quad (\text{RP})$$

Remarks:

- This continuous relaxation is generally a *nonconvex NLP* problem.
- Relaxed problem (RP) is valid, if the set of feasible solutions (FS) of (P) is a subset of the set of feasible solutions of (RP).

$$FS(P) \subseteq FS(RP).$$

Cont....

$$\min_x f(x) = (x-0.5)^2$$

subject to $x \in \{0,1,2\}$

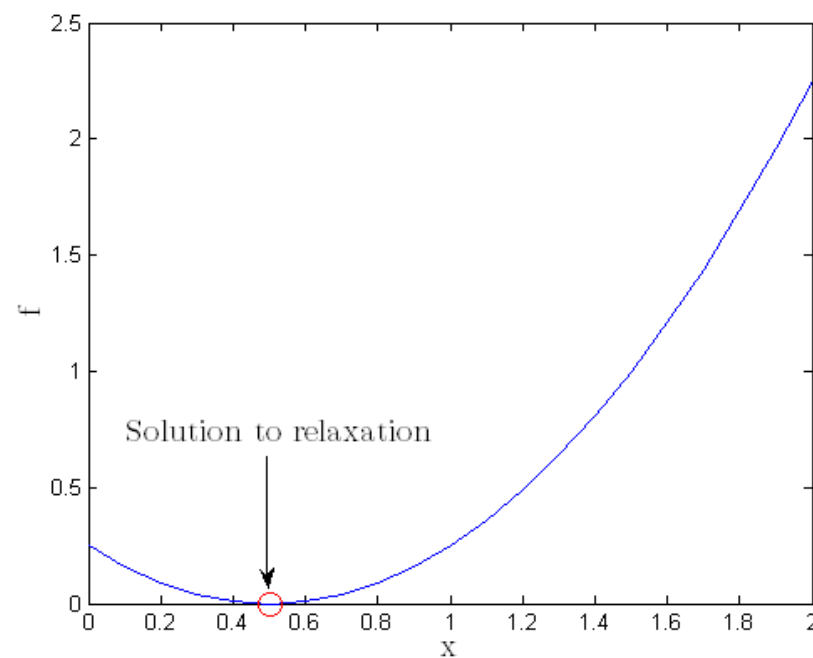
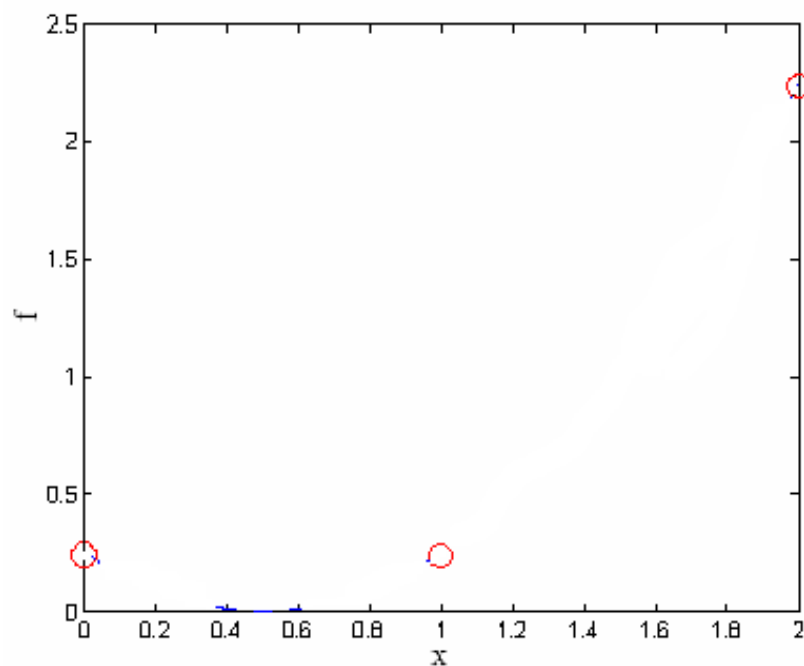
(P)



$$\min_x f(x) = (x-0.5)^2$$

subject to $0 \leq x \leq 2$

(RP)



Fathoming

- If a solution for any of the sub problems is greater than or equal to the best solution obtained so far, it is *discarded* from the search.
- Similarly, if a solution for any of the sub problems is less than best solution obtained so far, then best solution obtained can be *updated* with this obtained result.

(CS) : candidate sub problem (or current sub problem)

(RCS) : relaxed candidate sub problem

z_{inc} : best solution (or current incumbent)

Fathoming criteria's (C. A. Floudas,1995)

FC1: If (RCS) infeasible, then (CS) has no feasible solution.

FC2: If $z_{RCS} \geq z_{inc}$, then (CS) can be fathomed.

FC3: If $z_{RCS} < z_{inc}$ and integer $x_k \in \{L, U\}$, then (CS) can be fathomed

Branching

- We split the given problem (P) into two (or more) sub problems (called as candidate sub problems (CS)).
- Different rules are available such as, *maximal fractional branching, strong branching, and pseudo-costs branching* (Gupta and Ravindran, 1985).
- Strong and pseudo-costs branching are *computationally expensive*.

- We use *maximal fractional branching rule*. It selects variable for which maximum integer violation occurs. The idea is to get largest degradation of objective function when branching is carried out so that more sub problems can be discarded at an early stage.

$$\arg \max_k \{ \min(x_k - \lfloor x_k \rfloor, \lfloor x_k \rfloor + 1 - x_k) \}$$

where x_k are the integer variables to be branched, $\lfloor \cdot \rfloor$ rounds the elements to the nearest integer.

Illustrative example

- Consider the following simple four variable mixed-integer problem to be minimized.

$$\min z = 2x_1 - 3y_2 - 2y_3 - 3y_4$$

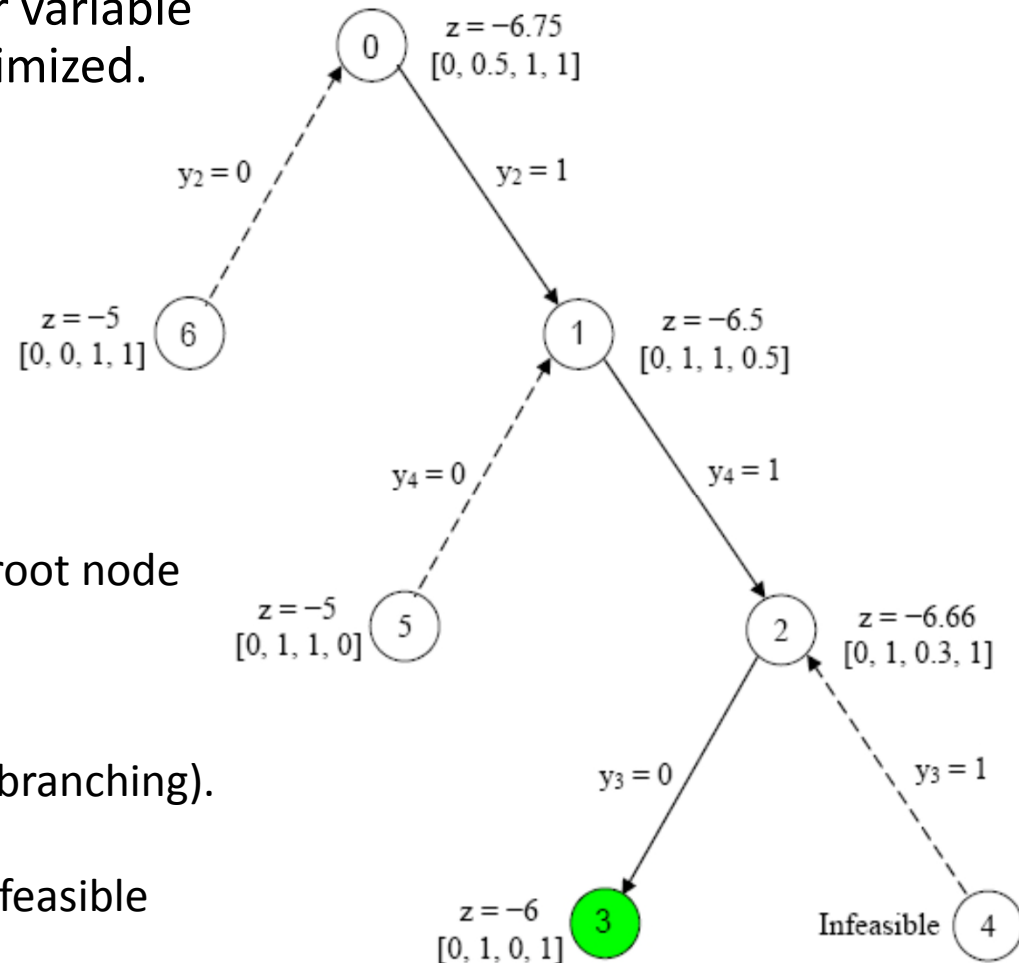
subject to

$$y_2 + y_3 + y_4 + x_1 \geq 2$$

$$5y_2 + 3y_3 + 4y_4 + 10x_1 \leq 10$$

$$x_1 \geq 0, y_k \in \{0,1\} \quad k = 2,3,4$$

- Circle and zero within itself indicates root node (original problem (P) to be solved).
- Below root node are children nodes (sub problems derived from (P) after branching).
- Green colored node indicates integer feasible solution found in the search tree.



Remark:

For simplicity in mixed-integer problems, we represent *continuous* variables by x and *integer/binary* variables by y .

MPL based predictive control using the Bernstein global optimization (Case study I)

- Chemical process plants involves discrete decisions (*on/off valves, speed selectors*) and continuous states (tank level, process temp.).
- Such systems results into unified framework known as hybrid system.
- Modeling of such systems is governed by interdependent physical laws, logic rules, and operating constraints.
- Mixed logical dynamical (MLD) [*Morari et al.*, 1999] and the multiple model approach [*Bhartiya et al.*, 2008] framework exist for modeling such systems.

Motivation

- Logical decision making and continuous dynamics results into a control law having both continuous and discrete variables.
- Implementation of control law requires online solution of MIQP/MINLP [*Morari et al.* 1999, and *Bhartiya et al.*, 2008].
- Solvers BNB20 (Kuipers, 2003) and GOA (*Duran et al.*, 1986) used for online control law implementation (cf. *Bhartiya, et al.* 2008, 2009).
- Limited to convex problems and may not *guarantee* optimality for nonconvex problems.
- Suggests need of the alternate approach.

Case Study: Spherical three-tank system

- Aim is to fill the tanks up to desired level, followed by multiple set-point changes.
- We pose this as an optimization problem: cost function to be minimized subjected to constraints on the input and output of the system.
- It has two inflow paths (Q_1, Q_2) to fill the tanks, and six solenoid valves (on/off) ($V_1, V_2, V_{13}, V_{23}, V_{L1}$ and V_{N3}) to control the flow between the intermediate tanks.
- Following are the process variables

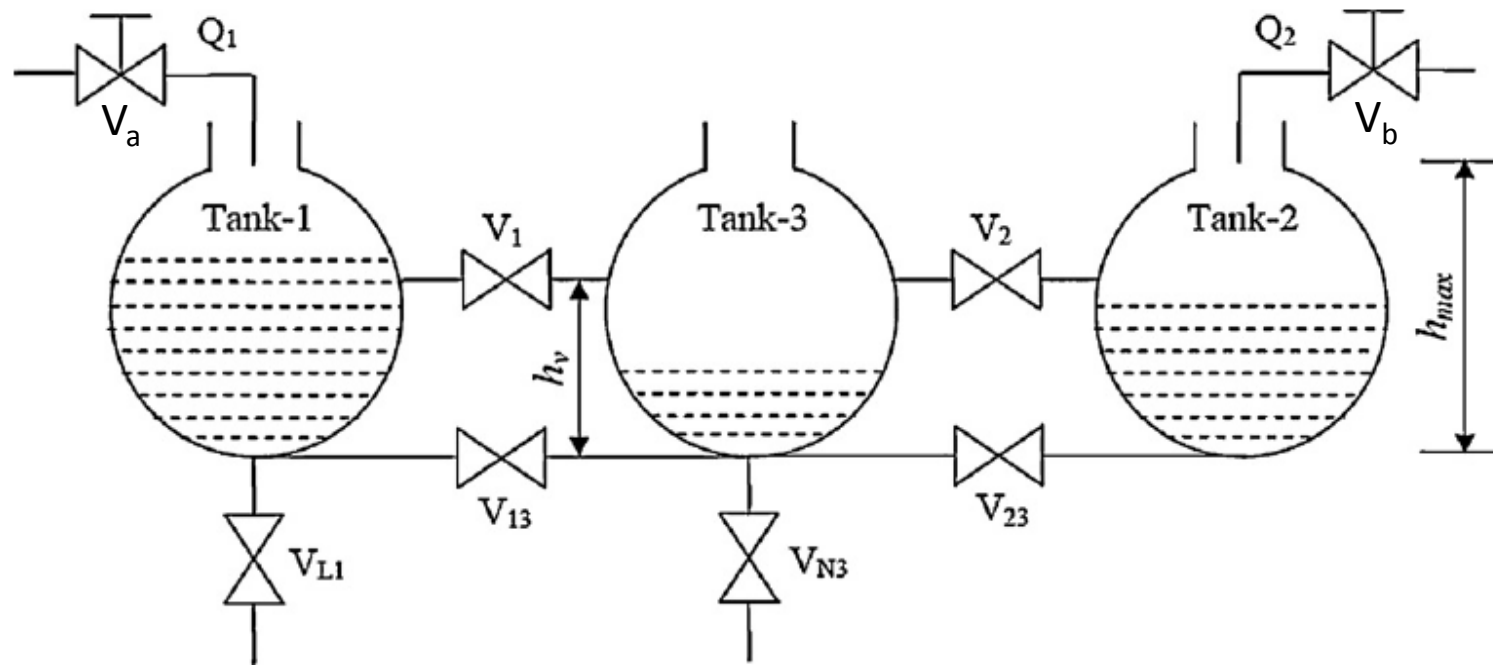
$$x^c = x = y = [h_1 \ h_2 \ h_3]^T$$

$$u^c = [Q_1 \ Q_2]^T$$

$$u^d = \delta = [V_1 \ V_2 \ V_{13} \ V_{23} \ V_{L1} \ V_{N3}]^T$$

- Design data used are $h_v = 0.3$ m, $h_{\max} = 0.6$ m, and the cross-sectional areas of valves $S_i = S_{i3} = S_{L1} = S_N = 0.95 \text{ cm}^2$

Cont...



- First principles model is given as follows:

$$\pi h_1 (h_{\max} - h_1) \frac{dh_1}{dt} = (Q_1 - V_{13}Q_{13V_{13}} - V_1Q_{13V_1} - V_{L1}Q_{L1})$$

$$\pi h_2 (h_{\max} - h_2) \frac{dh_2}{dt} = (Q_2 - V_{23}Q_{23V_{23}} - V_2Q_{23V_2})$$

$$\pi h_3 (h_{\max} - h_3) \frac{dh_3}{dt} = (V_{13}Q_{13V_{13}} + V_{23}Q_{23V_{23}} + V_1Q_{13V_1} + V_2Q_{23V_2} - V_{N3}Q_{N})$$

where V_1 , V_2 , V_{13} , V_{23} , V_{L1} and V_{N3} represents binary indicator variables

- Variables Q_i represent flowrates through valves V_i

$$Q_{i3V_{13}} = a_z S_{i3} \text{sign}(h_i - h_3) \sqrt{|2g(h_i - h_3)|}, \quad i = 1, 2$$

$$Q_{L1} = a_z S_{L1} \sqrt{2gh_1}$$

$$Q_N = a_z S_N \sqrt{2gh_3}$$

$$Q_{i3V_i} = V_i a_z S_i \text{sign}(\max\{h_i, h_v\} - \max\{h_3, h_v\}) \times \sqrt{|2g(\max\{h_i, h_v\} - \max\{h_3, h_v\})|}$$

- We considered three different points of linearization as follows:
 - Model-I: $h_1 = h_2 = 0.15$ ($25\%h_{\max}$), $h_3 = 0.14$ ($23\%h_{\max}$) and $Q_1 = Q_2 = 0$
 - Model-II: $h_1 = h_2 = 0.25$ ($42\%h_{\max}$), $h_3 = 0.24$ ($40\%h_{\max}$) and $Q_1 = Q_2 = 0$
 - Model-III: $h_1 = h_2 = 0.35$ ($58\%h_{\max}$), $h_3 = 0.34$ ($57\%h_{\max}$) and $Q_1 = Q_2 = 0$

Remarks:

- We have used *heuristics information* about levels of the tank such as, low, medium and high for choosing the *local operating points*.
- Models I and II correspond to levels below the upper pipe connections while Model III corresponds to a level above the upper pipe.

MPC formulation using MPL

We use a quadratic cost function of the form

$$\min_{\substack{u_k^c, \dots, u_{k+p-1}^c \\ \delta_k, \dots, \delta_{k+m-1}}} J = \sum_{i=1}^p \|y_{k+i} - y_{ref}\|_{\wedge_y}^2 + \sum_{i=0}^{m-1} \|u_{k+i}^c - u_{k+i-1}^c\|_{\wedge_u}^2$$

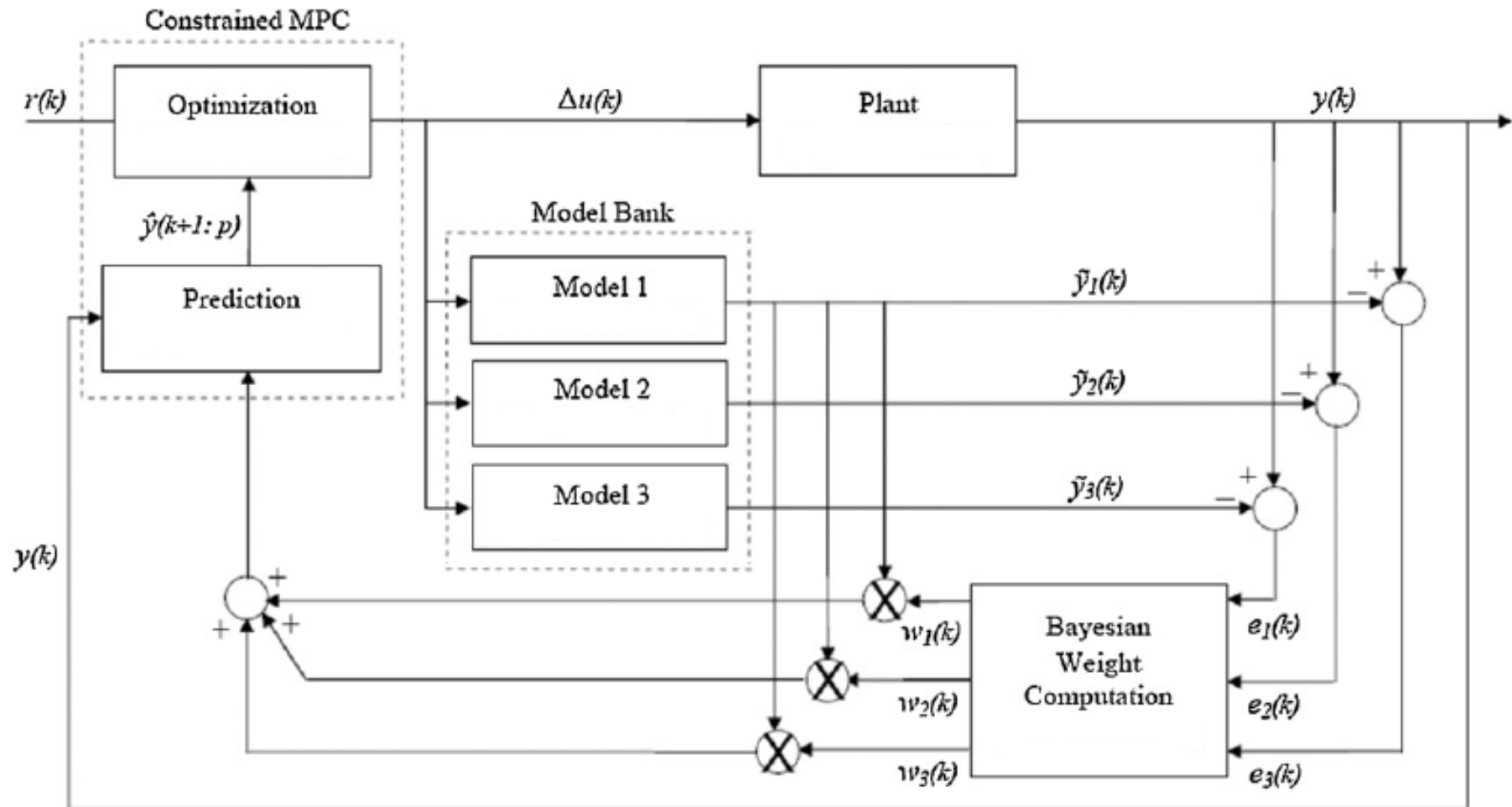
subject to mixed-integer constraints

$$y_{\min} \leq y \leq y_{\max}$$

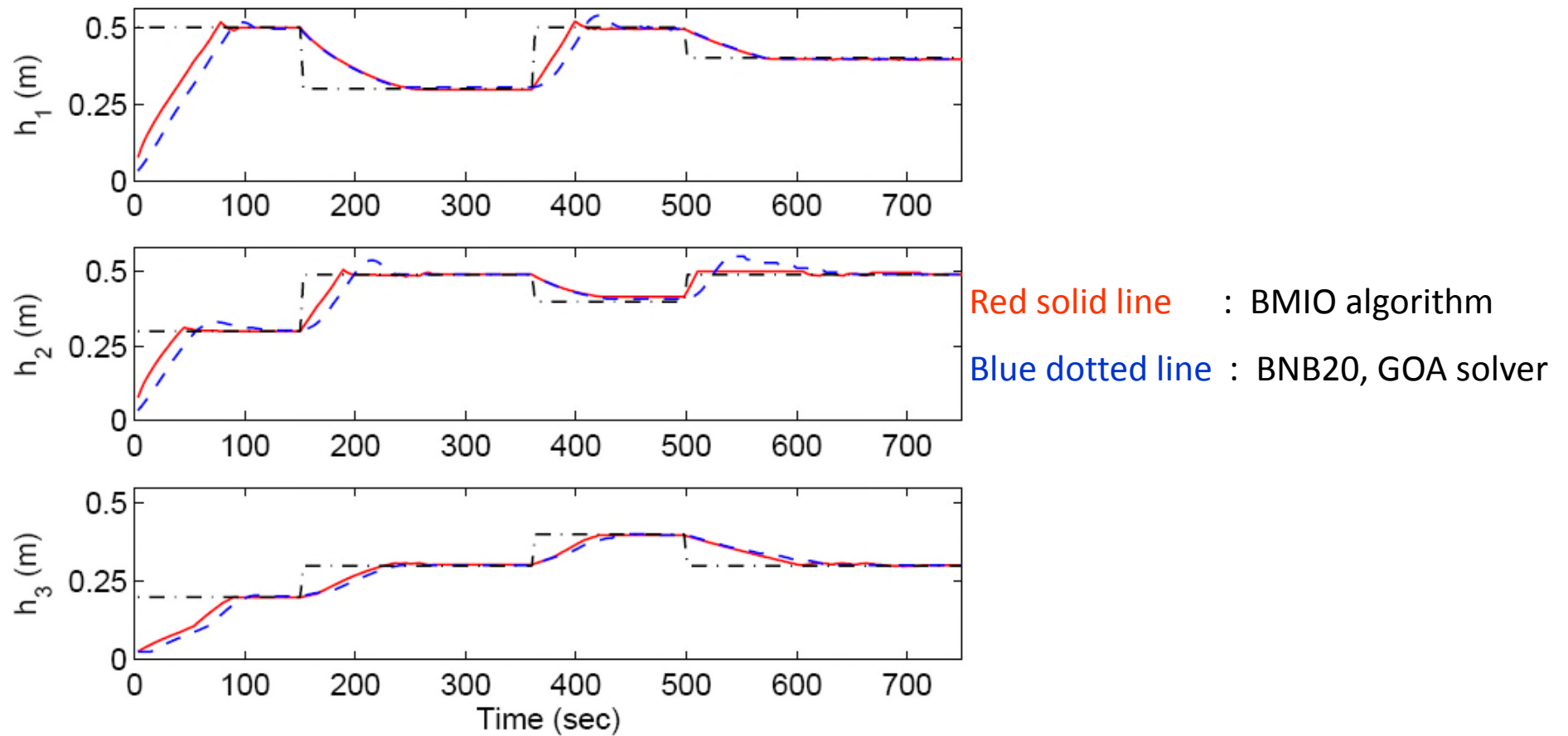
$$u_{\min}^c \leq u \leq u_{\max}^c$$

- y_{ref} is the reference set-point trajectory
- y_{k+i} is the predicted output
- u_{k+i}^c and u_{k+i-1}^c are the current and past control moves

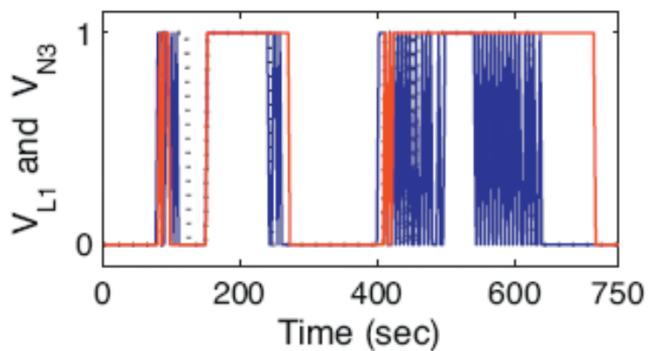
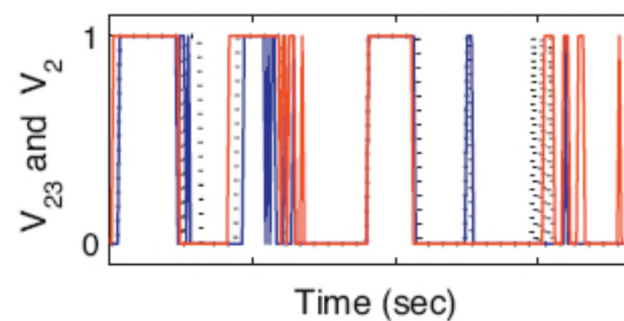
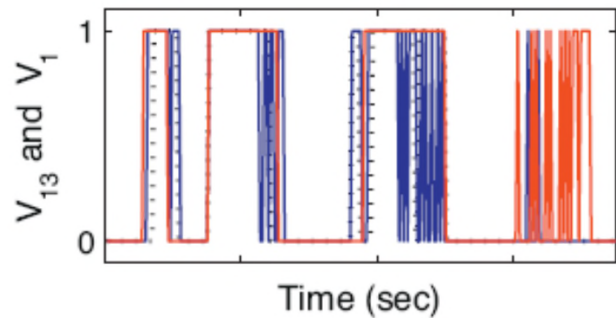
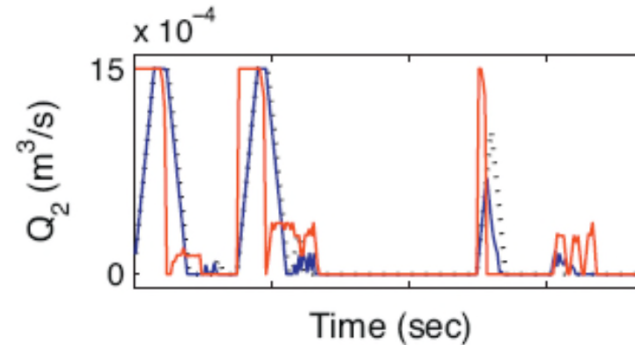
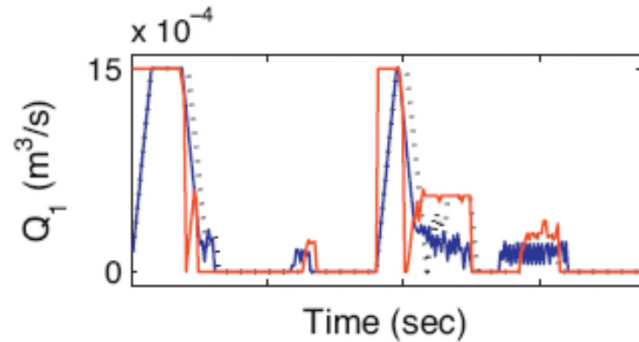
MPL model based predictive control scheme



Process Responses (BNB20, GOA, BMIO)

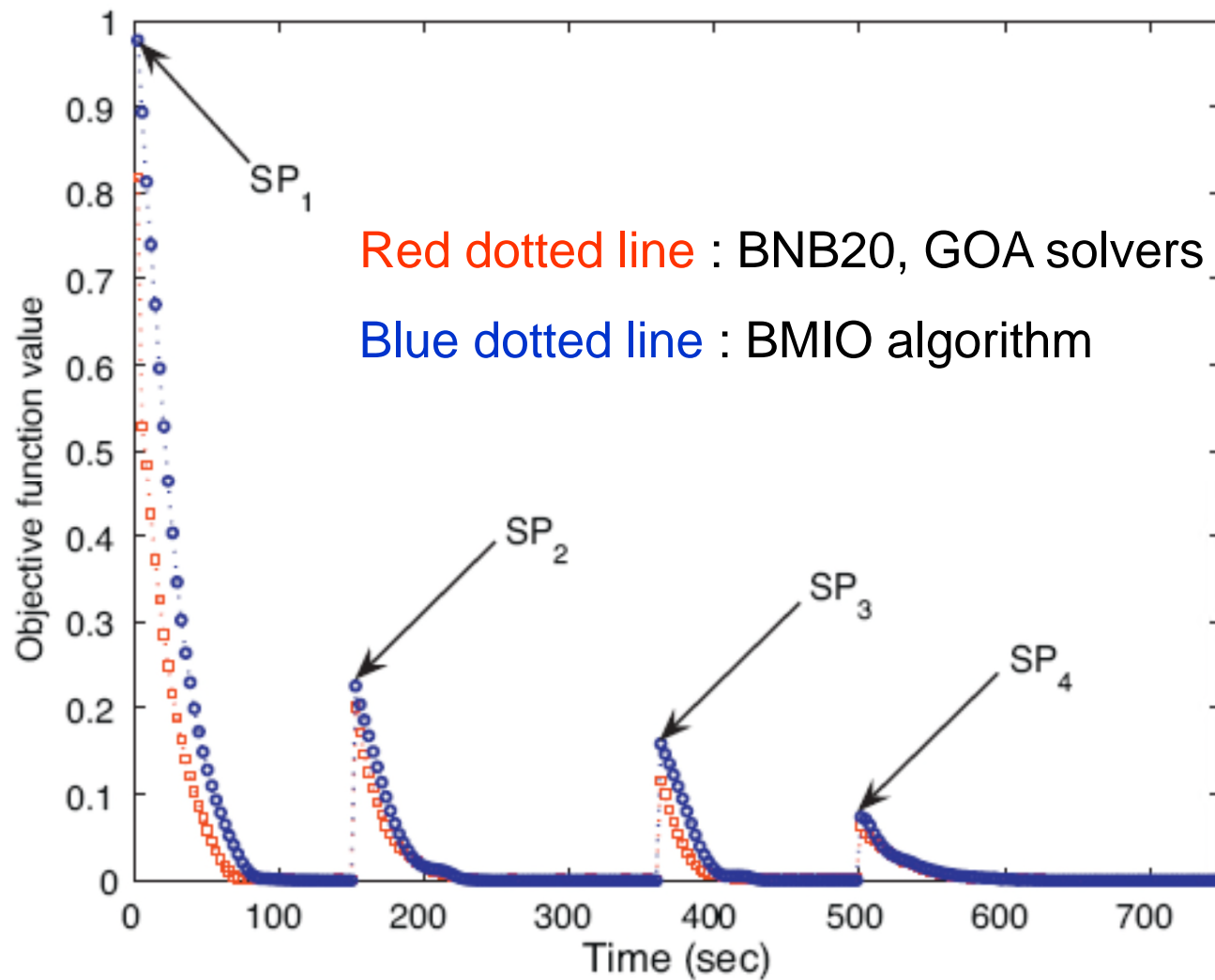


Computed control moves (BNB20, GOA, BMIO)

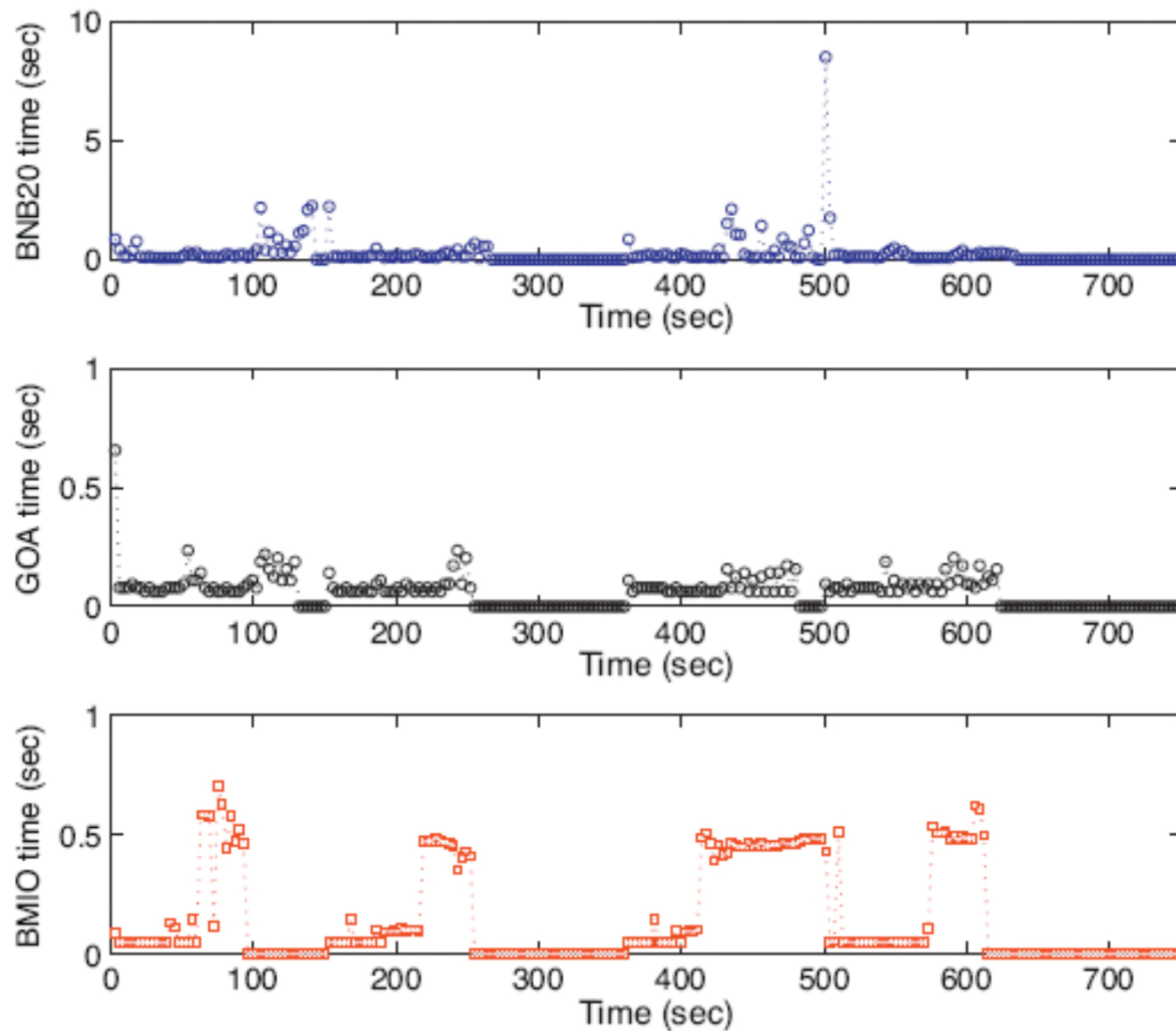


Red line : BMIO algorithm
Blue line : BNB20 solver
Black dotted line : GOA solver

Objective function values (BNB20, GOA, BMIO)



CPU time (BNB20, GOA, BMIO)



Findings

- Simulation duration 750 s, sampling time 3 s, prediction horizon 5, and control horizon 2.
- Average computational time taken to compute control moves are as follows:

BMIO algorithm : 0.19 s (*trade-off performed at the cost of the global optimization*)

BNB20 solver : 0.05 s

GOA solver : 0.02 s

Cont...

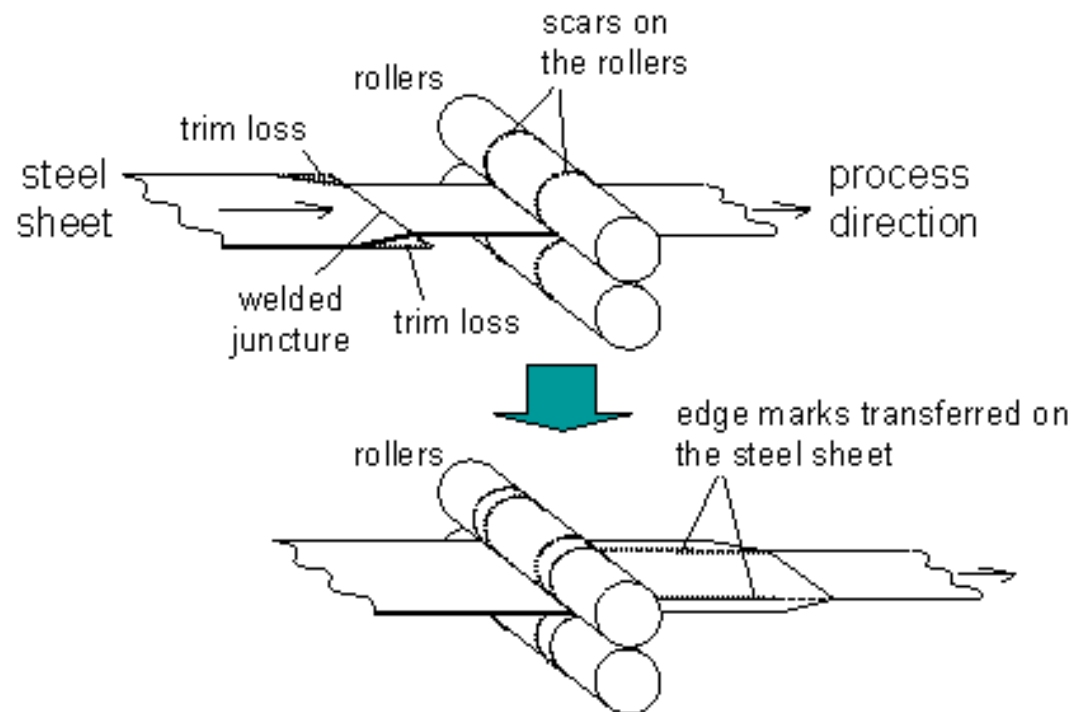
Case	Solver / Algorithm	Avg. objective function Value	Avg. reduction obtained in objective function value using BMIO algorithm
1	BNB20	0.0477	38 %
	GOA	0.0477	
	BMIO	0.0296	
2	BNB20	0.0477	10 %
	GOA	0.0477	
	BMIO	0.0430	

Cont...

Height (m)	Set-point change	Rise Time (s)			Avg. reduction obtained in the rise time using BMIO algorithm
		BNB20	GOA	BMIO	
h1	0-0.5	75	75	63	16 %
h2	0-0.3	45	45	33	27 %
h3	0-0.2	90	90	78	13 %

Trim-loss minimization (Case Study II)

- Aim is to cut required pieces from the given raw material with minimal wastage called as trim-loss minimization (Westerlund et al., 1998).
- Design variables include 5 binary (y_j) and 30 integer (m_j and n_{ij}).



Cont...

- Design data used

Parameters	Values				
b_i	330	360	370	415	435
n_i , order	12	6	15	6	9
B_{max}	2000 (mm)				
Δ	200 (mm)				
N_{max}	5				
$J(=I)$	5				
c_j	1				
C_j	$\frac{1}{10}j$				

Problem formulation (Trim-loss minimization)

$$\begin{aligned} \min_{m_j, y_j, n_{ij}} f(m_j, y_j, n_{ij}) &= \sum_{j=1}^J c_j m_j + C_j y_j \\ \text{subject to} \\ \sum_{i=1}^I b_i n_{ij} - B_{\max} &\leq 0, \quad j = 1, 2, \dots, J, \\ -\sum_{i=1}^I b_i n_{ij} + B_{\max} - \Delta &\leq 0, \quad j = 1, 2, \dots, J, \\ \sum_{i=1}^I n_{ij} - N_{\max} &\leq 0, \quad j = 1, 2, \dots, J, \\ y_j - m_j &\leq 0, \quad j = 1, 2, \dots, J, \\ m_j - M y_j &\leq 0, \quad j = 1, 2, \dots, J, \end{aligned}$$

*Nonlinear and nonconvex
due this bilinear inequality*

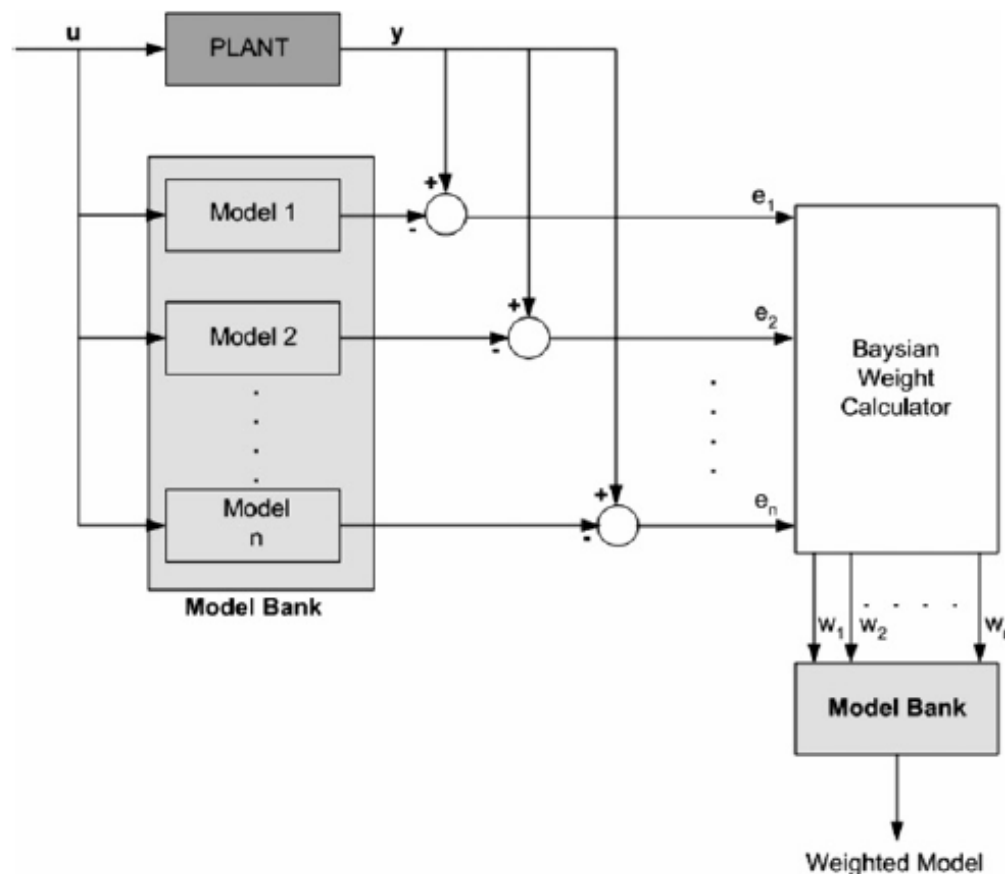
$$\begin{aligned} n_{i,\text{order}} - \sum_{j=1}^J m_j n_{ij} &\leq 0, \quad i = 1, 2, \dots, I, \\ n_{ij}, m_j &\in \mathbb{Z}^+, \quad y_j \in \{0, 1\}. \end{aligned}$$

Findings

- We compared the performance of the solvers BARON, Bonmin, and DICOPT with the Bernstein algorithm BMIO.
- Resulted obtained by BMIO agrees with literature reported results (MINLP library).

Solver/Algorithm	$f(m_j, y_j, n_{ij})$	Number of nodes	Time (s)
BARON	10.6	33,655	52.03
Bonmin	10.6	1,946	52.74
DICOPT	—	—	—
BMIO	10.3	8087	52.67

Bayesian weighting based multiple model scheme for obtaining multiple, partially linearized frameworks

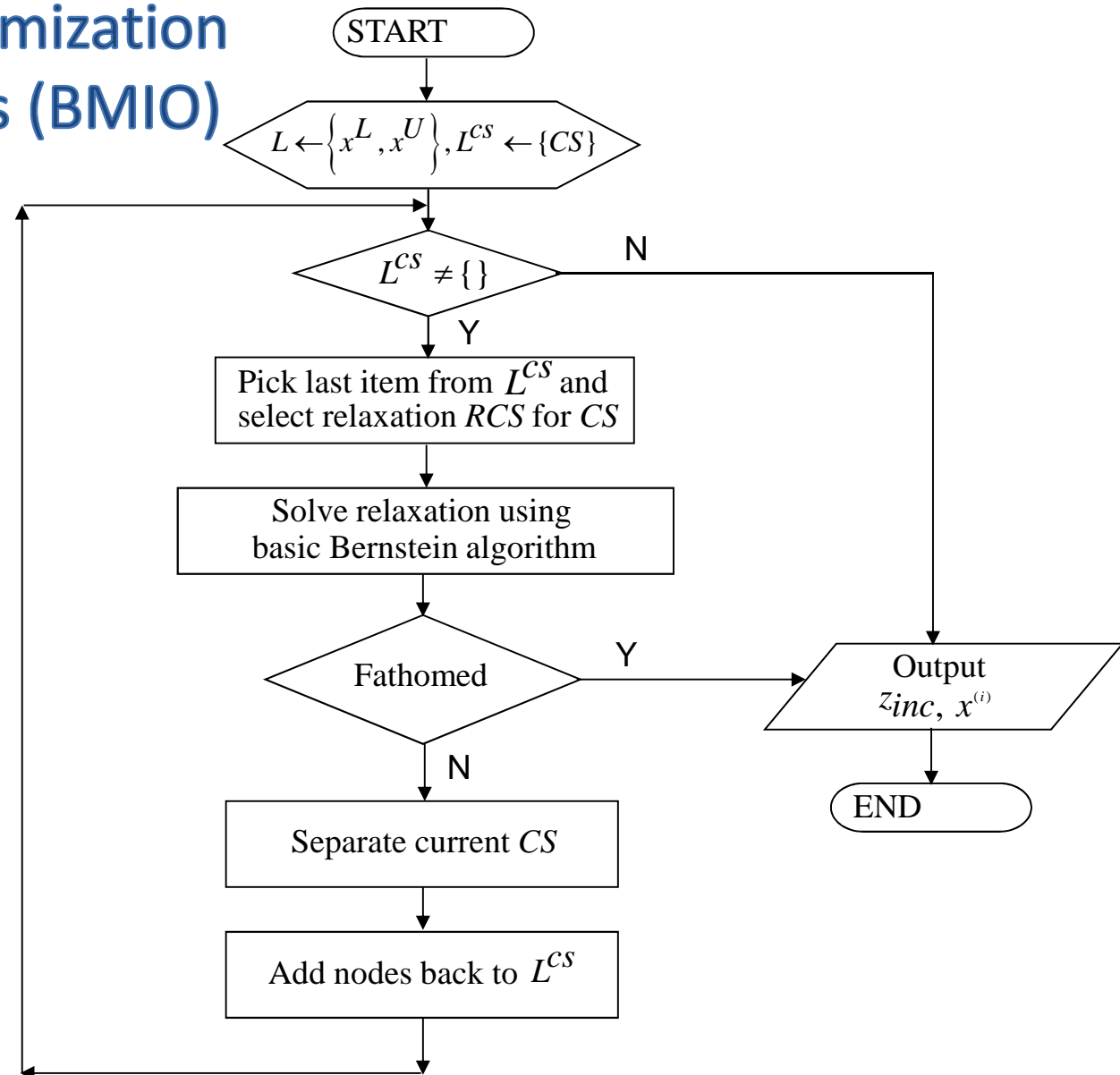


$$p_{ri,k} = \frac{\exp\left(-\frac{1}{2}\varepsilon_{i,k}^T K \varepsilon_{i,k}\right) p_{ri,k-1}}{\sum_{j=1}^{nl} \exp\left(-\frac{1}{2}\varepsilon_{j,k}^T K \varepsilon_{j,k}\right) p_{rj,k-1}}$$

$$w_{i,k} = \frac{p_{ri,k}}{\sum_{j=1}^{nl} p_{rj,k}}$$



Bernstein global optimization algorithm for MINLPs (BMIO)



Remark:

CS indicates the optimization problem (consisting of objective function to be minimized subjected to some constraints over domain of interest $X = [x^L, x^U]$).

