Semi-infinite programming and applications

A. Ismael F. Vaz

Production and Systems Department
Engineering School
Minho University - Braga - Portugal
aivaz@dps.uminho.pt

with special thanks to Eugénio Ferreira and Edite Fernandes.

Universidade Federal do Rio de Janeiro

12 November 2007

Outline

Outline

1. Semi-Infinite Programming (SIP) Notation
2. Numerical methods for SIP
3. Some practical applications
4. The particle swarm algorithm
5. Modification of PSOA for multi-local optimization

Outline

Outline

1. Semi-Infinite Programming (SIP) Notation
2. Numerical methods for SIP
3. Some practical applications
4. The particle swarm algorithm
5. Modification of PSOA for multi-local optimization

Outline

Outline

General formulation - Nonlinear semi-infinite programming

Problem

$$\begin{aligned} \min_{x \in R^n} \; & f(x) \\ s.t. \; & g(x,t) \leq 0 \\ & \forall t \in T \end{aligned} \qquad \text{(NLSIP)}$$

- $f(x)$ is the objective function
- $g(x,t)$ is the *infinite* constraint function
- $T \subset R^p$ is, usually, a cartesian product of intervals ($[\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \ldots \times [\alpha_p, \beta_p]$)

Note

A more general problem could be defined, but the extension is straightforward.

General formulation - Nonlinear semi-infinite programming

Problem

$$\min_{x \in R^n} \ f(x)$$
$$s.t. \ \ g(x,t) \leq 0$$
$$\forall t \in T \qquad \text{(NLSIP)}$$

- $f(x)$ is the objective function
- $g(x,t)$ is the *infinite* constraint function
- $T \subset R^p$ is, usually, a cartesian product of intervals ($[\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \ldots \times [\alpha_p, \beta_p]$)

Note

A more general problem could be defined, but the extension is straightforward.

General formulation - Nonlinear semi-infinite programming

Problem

$$\min_{x \in R^n} f(x)$$
$$s.t. \; g(x,t) \leq 0$$
$$\forall t \in T \qquad \text{(NLSIP)}$$

- $f(x)$ is the objective function
- $g(x,t)$ is the *infinite* constraint function
- $T \subset R^p$ is, usually, a cartesian product of intervals $([\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \ldots \times [\alpha_p, \beta_p])$

Note

A more general problem could be defined, but the extension is straightforward.

General formulation - Nonlinear semi-infinite programming

Problem

$$\begin{aligned} \min_{x \in R^n} \ & f(x) \\ s.t. \ & g(x,t) \leq 0 \\ & \forall t \in T \end{aligned} \qquad \text{(NLSIP)}$$

- $f(x)$ is the objective function
- $g(x,t)$ is the *infinite* constraint function
- $T \subset R^p$ is, usually, a cartesian product of intervals $([\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \ldots \times [\alpha_p, \beta_p])$

Note

A more general problem could be defined, but the extension is straightforward.

General formulation - Nonlinear semi-infinite programming

Problem

$$\begin{array}{rl} \min\limits_{x\in R^n} & f(x) \\ s.t. & g(x,t) \le 0 \\ & \forall t \in T \end{array} \qquad \text{(NLSIP)}$$

- $f(x)$ is the objective function
- $g(x,t)$ is the *infinite* constraint function
- $T \subset R^p$ is, usually, a cartesian product of intervals ($[\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \ldots \times [\alpha_p, \beta_p]$)

Note

A more general problem could be defined, but the extension is straightforward.

Why semi-infinite programming?

The infinite set T may be viewed as an infinite index set, *i.e.*,

Index set

$$\min_{x \in R^n} \; f(x)$$
$$s.t. \; g_t(x) \le 0 \qquad \forall t \in T \qquad \text{(NLSIP)}$$

Semi-infinite

The problem has a finite number of variables subject to an infinite number of constraints.

Practical applications

In practical applications the index set T is related with time or space.

Why semi-infinite programming?

The infinite set T may be viewed as an infinite index set, *i.e.*,

Index set

$$\begin{array}{rl} \min\limits_{x \in R^n} & f(x) \\ s.t. & g_t(x) \leq 0 \qquad \forall t \in T \end{array} \qquad \text{(NLSIP)}$$

Semi-infinite

The problem has a finite number of variables subject to an infinite number of constraints.

Practical applications

In practical applications the index set T is related with time or space.

Why semi-infinite programming?

The infinite set T may be viewed as an infinite index set, *i.e.*,

Index set

$$\begin{aligned} \min_{x \in R^n} \ & f(x) \\ s.t. \ & g_t(x) \leq 0 \qquad \forall t \in T \end{aligned} \tag{NLSIP}$$

Semi-infinite

The problem has a finite number of variables subject to an infinite number of constraints.

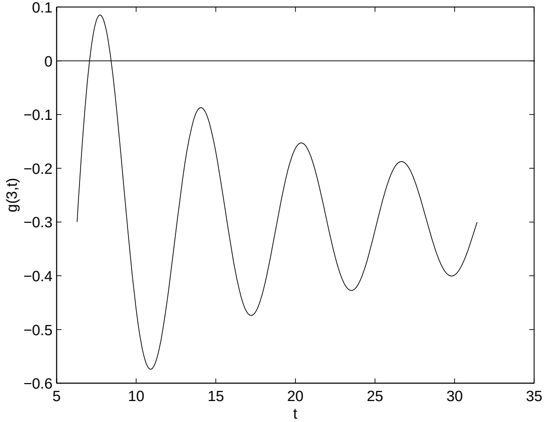
Practical applications

In practical applications the index set T is related with time or space.

An very simple academic example ($n = 1$ and $p = 1$)

Example

$$\min_{x \in R} \quad x^2, \quad s.t. \ \frac{x}{t}\sin(t) - \frac{x}{10} \leq 0, \quad \forall t \in [2\pi, 10\pi]$$



$$g(3,t) = \frac{3}{t}\sin(t) - \frac{3}{10}$$

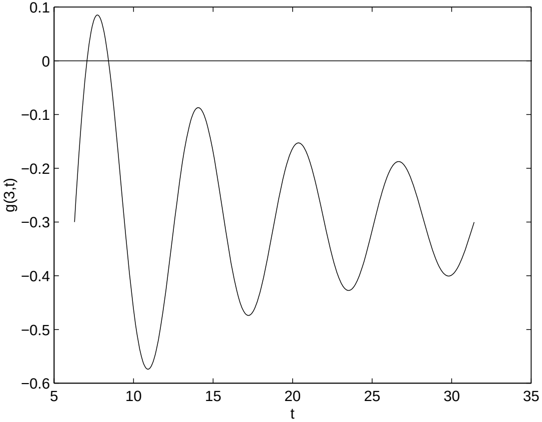
Feasibility

Is $\bar{x} = 3$ feasible?

An very simple academic example ($n = 1$ and $p = 1$)

Example

$$\min_{x \in R} \quad x^2, \quad s.t. \ \frac{x}{t}\sin(t) - \frac{x}{10} \leq 0, \quad \forall t \in [2\pi, 10\pi]$$



$$g(3,t) = \tfrac{3}{t}\sin(t) - \tfrac{3}{10}$$

Feasibility

Is $\bar{x} = 3$ feasible?

Another example – Chebyshev approximation problem

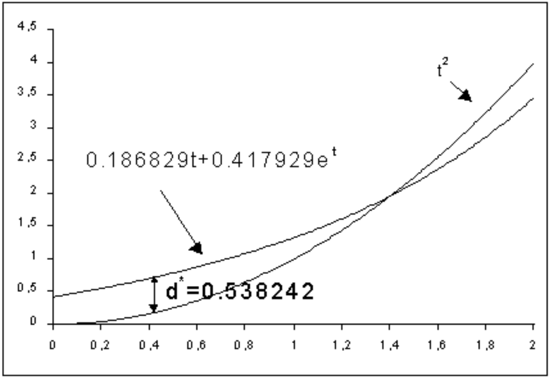
To approximate the function t^2 by a combination of t and e^t function in a given set. d is the minimum distance.

$$\min_{p,d \in R^{2+1}} \quad d$$

$s.t.$

$$|t^2 - (p_1 t + p_2 e^t)| \leq d$$

$$\forall t \in [0, 2]$$



Definition of stationary point

Let $x^* \in R^n$ be a point such that

$$g(x^*, t) \leq 0, \ \forall t \in T,$$

and there exists t^1, t^2, $\ldots$, t^{m^*} $(\in T)$ and non negative numbers $\lambda_*^0, \lambda_*^1, \lambda_*^2, ..., \lambda_*^{m^*}$ such that

$$\lambda_*^0 \nabla_x f(x^*) + \sum_{i=1}^{m^*} \lambda_*^i \nabla_x g(x^*, t^i) = 0.$$

with

$$g(x^*, t^i) = 0, \ \ i = 1, ..., m^*.$$

Then x^* is a stationary point for the (NLSIP).

Where global (multi-local) optimization plays a role?

The t^i, $i = 1, \ldots, m^*$, points are global solutions of the problem

Multi-local problem (also called lower level problem)

$$\max_{t \in T} g(x^*, t)$$

- The simple check for feasibility requests the computation of the global solutions for the lower level problem (not complectly true).
- In order to obtain global convergence for some methods the computation of all the global and local solutions for the lower level problem is necessary.

Where global (multi-local) optimization plays a role?

The t^i, $i = 1, \ldots, m^*$, points are global solutions of the problem

Multi-local problem (also called lower level problem)

$$\max_{t \in T} g(x^*, t)$$

- The simple check for feasibility requests the computation of the global solutions for the lower level problem (not complectly true).
- In order to obtain global convergence for some methods the computation of all the global and local solutions for the lower level problem is necessary.

Where global (multi-local) optimization plays a role?

The t^i, $i = 1, \ldots, m^*$, points are global solutions of the problem

Multi-local problem (also called lower level problem)

$$\max_{t \in T} g(x^*, t)$$

- The simple check for feasibility requests the computation of the global solutions for the lower level problem (not complectly true).
- In order to obtain global convergence for some methods the computation of all the global and local solutions for the lower level problem is necessary.

Outline

Available numerical methods

Methods

- Discretization
- Exchange
- Reduction
- Constraints transcription
- Dual

Available numerical methods

Methods

- Discretization
- Exchange
- Reduction
- Constraints transcription
- Dual

Available numerical methods

Methods

- Discretization
- Exchange
- Reduction
- Constraints transcription
- Dual

Available numerical methods

Methods

- Discretization
- Exchange
- Reduction
- Constraints transcription
- Dual

Available numerical methods

Methods

- Discretization
- Exchange
- Reduction
- Constraints transcription
- Dual

Available numerical methods

Methods

- Discretization
- Exchange
- Reduction
- Constraints transcription
- Dual

Discretization methods

First approach

A natural way to address problem (NLSIP) is to consider a discretization of the set T (in an equally spaced grid of points).
Usually discretization methods solve a sequence of finite (discretized) problems for successive grid refinements.

Bad properties

- These methods are *Outer approximation* methods and an infeasible solution is usually obtained.
- The problems solution is only obtained when the grid is *close* to the set T.
- High number of constraints to be considered if an accurate solution is requested (ill conditioning can occur).

Discretization methods

First approach

A natural way to address problem (NLSIP) is to consider a discretization of the set T (in an equally spaced grid of points).
Usually discretization methods solve a sequence of finite (discretized) problems for successive grid refinements.

Bad properties

- These methods are *Outer approximation* methods and an infeasible solution is usually obtained.
- The problems solution is only obtained when the grid is *close* to the set T.
- High number of constraints to be considered if an accurate solution is requested (ill conditioning can occur).

Discretization methods

First approach

A natural way to address problem (NLSIP) is to consider a discretization of the set T (in an equally spaced grid of points).
Usually discretization methods solve a sequence of finite (discretized) problems for successive grid refinements.

Bad properties

- These methods are *Outer approximation* methods and an infeasible solution is usually obtained.
- The problems solution is only obtained when the grid is *close* to the set T.
- High number of constraints to be considered if an accurate solution is requested (ill conditioning can occur).

Discretization methods

First approach

A natural way to address problem (NLSIP) is to consider a discretization of the set T (in an equally spaced grid of points).
Usually discretization methods solve a sequence of finite (discretized) problems for successive grid refinements.

Bad properties

- These methods are *Outer approximation* methods and an infeasible solution is usually obtained.
- The problems solution is only obtained when the grid is *close* to the set T.
- High number of constraints to be considered if an accurate solution is requested (ill conditioning can occur).

Discretization methods

First approach

A natural way to address problem (NLSIP) is to consider a discretization of the set T (in an equally spaced grid of points).
Usually discretization methods solve a sequence of finite (discretized) problems for successive grid refinements.

Bad properties

- These methods are *Outer approximation* methods and an infeasible solution is usually obtained.
- The problems solution is only obtained when the grid is *close* to the set T.
- High number of constraints to be considered if an accurate solution is requested (ill conditioning can occur).

Discretization methods

Good properties

It is easy to implement and a solver for finite optimization can be used.

Algorithm (h is a grid space parameter)

S0: Define $T[h^0]$. Let $\widetilde{T}[h^0] = T[h^0]$. Solve NLP($\widetilde{T}[h^0]$) and let x_0 be the found solution.

Sk: If x_{k-1} is not feasible $\forall t \in T[h^{k-1}]$

Go to Step k

Step $k + 1$

Discretization methods

Good properties

It is easy to implement and a solver for finite optimization can be used.

Algorithm (h is a grid space parameter)

S0: Define $T[h^0]$. Let $\widetilde{T}[h^0] = T[h^0]$. Solve NLP($\widetilde{T}[h^0]$) and let x_0 be the found solution.

Sk: If x_{k-1} is not feasible $\forall t \in T[h^{k-1}]$

Discretization methods

Good properties

It is easy to implement and a solver for finite optimization can be used.

Algorithm (h is a grid space parameter)

S0: Define $T[h^0]$. Let $\widetilde{T}[h^0] = T[h^0]$. Solve NLP($\widetilde{T}[h^0]$) and let x_0 be the found solution.

Sk: If x_{k-1} is not feasible $\forall t \in T[h^{k-1}]$

then: Include all infeasible points into $\widetilde{T}[h^{k-1}]$. Solve NLP($\widetilde{T}[h^{k-1}]$) and let x_{k-1} be the found solution. Go to Step k.

If the maximum number of refinements is attained then stop.

Otherwise build another set $\widetilde{T}[h^k]$ from $T[h^k]$ and $\widetilde{T}[h^{k-1}]$. Solve NLP($\widetilde{T}[h^k]$) and let x_k be the found solution. Go to Step $k+1$.

Discretization methods

Good properties

It is easy to implement and a solver for finite optimization can be used.

Algorithm (h is a grid space parameter)

S0: Define $T[h^0]$. Let $\widetilde{T}[h^0] = T[h^0]$. Solve NLP($\widetilde{T}[h^0]$) and let x_0 be the found solution.

Sk: If x_{k-1} is not feasible $\forall t \in T[h^{k-1}]$

then: Include all infeasible points into $\widetilde{T}[h^{k-1}]$. Solve NLP($\widetilde{T}[h^{k-1}]$) and let x_{k-1} be the found solution. Go to Step k.

else: If the maximum number of refinements is attained then stop. Otherwise build another set $\widetilde{T}[h^k]$ from $T[h^k]$ and $\widetilde{T}[h^{k-1}]$. Solve NLP($\widetilde{T}[h^k]$) and let x_k be the found solution. Go to Step $k+1$.

Discretization methods

Good properties

It is easy to implement and a solver for finite optimization can be used.

Algorithm (h is a grid space parameter)

S0: Define $T[h^0]$. Let $\widetilde{T}[h^0] = T[h^0]$. Solve NLP($\widetilde{T}[h^0]$) and let x_0 be the found solution.

Sk: If x_{k-1} is not feasible $\forall t \in T[h^{k-1}]$

then: Include all infeasible points into $\widetilde{T}[h^{k-1}]$. Solve NLP($\widetilde{T}[h^{k-1}]$) and let x_{k-1} be the found solution. Go to Step k.

else: If the maximum number of refinements is attained then stop. Otherwise build another set $\widetilde{T}[h^k]$ from $T[h^k]$ and $\widetilde{T}[h^{k-1}]$. Solve NLP($\widetilde{T}[h^k]$) and let x_k be the found solution. Go to Step $k+1$.

Discretization methods

Good properties

It is easy to implement and a solver for finite optimization can be used.

Algorithm (h is a grid space parameter)

S0: Define $T[h^0]$. Let $\widetilde{T}[h^0] = T[h^0]$. Solve NLP($\widetilde{T}[h^0]$) and let x_0 be the found solution.

Sk: If x_{k-1} is not feasible $\forall t \in T[h^{k-1}]$

then: Include all infeasible points into $\widetilde{T}[h^{k-1}]$. Solve NLP($\widetilde{T}[h^{k-1}]$) and let x_{k-1} be the found solution. Go to Step k.

else: If the maximum number of refinements is attained then stop. Otherwise build another set $\widetilde{T}[h^k]$ from $T[h^k]$ and $\widetilde{T}[h^{k-1}]$. Solve NLP($\widetilde{T}[h^k]$) and let x_k be the found solution. Go to Step $k+1$.

Exchange methods

In exchange methods approximate solution(s) to the problem (we have as many subproblems as infinite constraints in the (NLSIP)).

Lower level subproblem (multi-local)

$$\max_{t \in T} g(\bar{x}, t)$$

Key idea

The solution(s) (points) of the lower level subproblem are added to a set $\tilde{T}$ while previous added points may be dropped (exchange of points).
A sequence of finite problems is solved considering the set T replaced by the finite set $\tilde{T}$.

Exchange methods

In exchange methods approximate solution(s) to the problem (we have as many subproblems as infinite constraints in the (NLSIP)).

Lower level subproblem (multi-local)

$$\max_{t \in T} g(\bar{x}, t)$$

Key idea

The solution(s) (points) of the lower level subproblem are added to a set $\tilde{T}$ while previous added points may be dropped (exchange of points).
A sequence of finite problems is solved considering the set T replaced by the finite set $\tilde{T}$.

Properties and algorithm

Bad properties

Slow rate of convergence.

Exchange algorithm

1. Let $T^0 = \emptyset$, x^0 be an initial guess and $k = 0$.
2. Approximately solve the lower level subproblem $S^k = \arg\max_{t\in T} g(x^k, t)$
3. If $g(x^k, t) \le 0$, $\forall t \in S^k$ then stop
4. Add the new constraints and eventually drop others $(T^{k+1} \subseteq T^k \bigcup S^k)$.
5. Solve NLP(T^{k+1}) and let x^{k+1} be the solution found
6. Set $k = k + 1$ and go to step 2

Properties and algorithm

Bad properties

Slow rate of convergence.

Exchange algorithm

1. Let $\tilde{T}^0 = \emptyset$, x^0 be an initial guess and $k = 0$.
2. Approximately solve the lower level subproblem $S^k = arg\max_{t\in T} g(x^k, t)$.
3. if $g(x^k, t) \leq 0$, $\forall t \in S^k$ then stop.
4. Add the new constraints and eventually drop others ($\tilde{T}^{k+1} \subseteq \tilde{T}^k \bigcup S^k$).
5. Solve NLP($\tilde{T}^{k+1}$) and let x^{k+1} be the solution found.
6. Set $k = k + 1$ and go to step 2.

Properties and algorithm

Bad properties

Slow rate of convergence.

Exchange algorithm

1. Let $\tilde{T}^0 = \emptyset$, x^0 be an initial guess and $k = 0$.
2. Approximately solve the lower level subproblem $S^k = arg\max_{t \in T} g(x^k, t)$.
3. if $g(x^k, t) \leq 0$, $\forall t \in S^k$ then stop.
4. Add the new constraints and eventually drop others $(\tilde{T}^{k+1} \subseteq \tilde{T}^k \bigcup S^k)$.
5. Solve NLP($\tilde{T}^{k+1}$) and let x^{k+1} be the solution found.
6. Set $k = k + 1$ and go to step 2.

Properties and algorithm

Bad properties

Slow rate of convergence.

Exchange algorithm

1. Let $\tilde{T}^0 = \emptyset$, x^0 be an initial guess and $k = 0$.
2. Approximately solve the lower level subproblem $S^k = arg\max_{t \in T} g(x^k, t)$.
3. if $g(x^k, t) \leq 0$, $\forall t \in S^k$ then stop.
4. Add the new constraints and eventually drop others ($\tilde{T}^{k+1} \subseteq \tilde{T}^k \bigcup S^k$).
5. Solve NLP($\tilde{T}^{k+1}$) and let x^{k+1} be the solution found.
6. Set $k = k + 1$ and go to step 2.

Properties and algorithm

Bad properties

Slow rate of convergence.

Exchange algorithm

1. Let $\tilde{T}^0 = \emptyset$, x^0 be an initial guess and $k = 0$.
2. Approximately solve the lower level subproblem $S^k = arg\max_{t \in T} g(x^k, t)$.
3. if $g(x^k, t) \leq 0$, $\forall t \in S^k$ then stop.
4. Add the new constraints and eventually drop others ($\tilde{T}^{k+1} \subseteq \tilde{T}^k \bigcup S^k$).
5. Solve NLP($\tilde{T}^{k+1}$) and let x^{k+1} be the solution found.
6. Set $k = k + 1$ and go to step 2.

Properties and algorithm

Bad properties

Slow rate of convergence.

Exchange algorithm

1. Let $\tilde{T}^0 = \emptyset$, x^0 be an initial guess and $k = 0$.
2. Approximately solve the lower level subproblem $S^k = arg\max_{t \in T} g(x^k, t)$.
3. if $g(x^k, t) \leq 0$, $\forall t \in S^k$ then stop.
4. Add the new constraints and eventually drop others ($\tilde{T}^{k+1} \subseteq \tilde{T}^k \bigcup S^k$).
5. Solve NLP($\tilde{T}^{k+1}$) and let x^{k+1} be the solution found.
6. Set $k = k + 1$ and go to step 2.

Properties and algorithm

Bad properties

Slow rate of convergence.

Exchange algorithm

1. Let $\tilde{T}^0 = \emptyset$, x^0 be an initial guess and $k = 0$.
2. Approximately solve the lower level subproblem $S^k = arg\max_{t \in T} g(x^k, t)$.
3. if $g(x^k, t) \leq 0$, $\forall t \in S^k$ then stop.
4. Add the new constraints and eventually drop others $(\tilde{T}^{k+1} \subseteq \tilde{T}^k \bigcup S^k)$.
5. Solve NLP($\tilde{T}^{k+1}$) and let x^{k+1} be the solution found.
6. Set $k = k + 1$ and go to step 2.

Properties and algorithm

Bad properties

Slow rate of convergence.

Exchange algorithm

1. Let $\tilde{T}^0 = \emptyset$, x^0 be an initial guess and $k = 0$.
2. Approximately solve the lower level subproblem $S^k = arg\max_{t \in T} g(x^k, t)$.
3. if $g(x^k, t) \leq 0$, $\forall t \in S^k$ then stop.
4. Add the new constraints and eventually drop others $(\tilde{T}^{k+1} \subseteq \tilde{T}^k \bigcup S^k)$.
5. Solve NLP($\tilde{T}^{k+1}$) and let x^{k+1} be the solution found.
6. Set $k = k + 1$ and go to step 2.

Reduction type methods

Reduction methods

Reduction type methods use the more accurate solutions to the subproblem $\max_{t \in T} g(\bar{x}, t)$, computing all the global solutions and as much as possible the local ones (multi-local optimization).

Bad properties

Obtaining all the global and local maximizer is not an easy task (even for problems with only bound constraints).

Good properties

Good theoretical properties with fast convergence.

Reduction type methods

Reduction methods

Reduction type methods use the more accurate solutions to the subproblem $\max_{t \in T} g(\bar{x}, t)$, computing all the global solutions and as much as possible the local ones (multi-local optimization).

Bad properties

Obtaining all the global and local maximizer is not an easy task (even for problems with only bound constraints).

Good properties

Good theoretical properties with fast convergence.

Reduction type methods

Reduction methods

Reduction type methods use the more accurate solutions to the subproblem $\max_{t \in T} g(\bar{x}, t)$, computing all the global solutions and as much as possible the local ones (multi-local optimization).

Bad properties

Obtaining all the global and local maximizer is not an easy task (even for problems with only bound constraints).

Good properties

Good theoretical properties with fast convergence.

Conceptual algorithm

Reduction type algorithm

1. Let x^0 be an initial guess and $k = 0$.
2. Obtain all the global and local solutions to the lower level subproblem. Let $\mathcal{A}^k = arg\max_{t \in T} g(x^k, t)$.
3. if $g(x^k, t) \leq 0$, $\forall t \in \mathcal{A}^k$ then stop.
4. Solve NLP($\mathcal{A}^k$) and let x^{k+1} be the solution found.
5. Set $k = k + 1$ and go to step 2.

Conceptual algorithm

Reduction type algorithm

1. Let x^0 be an initial guess and $k = 0$.
2. Obtain all the global and local solutions to the lower level subproblem. Let $\mathcal{A}^k = arg\max_{t \in T} g(x^k, t)$.
3. if $g(x^k, t) \leq 0$, $\forall t \in \mathcal{A}^k$ then stop.
4. Solve NLP($\mathcal{A}^k$) and let x^{k+1} be the solution found.
5. Set $k = k + 1$ and go to step 2.

Conceptual algorithm

Reduction type algorithm

1. Let x^0 be an initial guess and $k = 0$.
2. Obtain all the global and local solutions to the lower level subproblem. Let $\mathcal{A}^k = arg\max_{t \in T} g(x^k, t)$.
3. if $g(x^k, t) \leq 0$, $\forall t \in \mathcal{A}^k$ then stop.
4. Solve NLP($\mathcal{A}^k$) and let x^{k+1} be the solution found.
5. Set $k = k + 1$ and go to step 2.

Conceptual algorithm

Reduction type algorithm

1. Let x^0 be an initial guess and $k = 0$.
2. Obtain all the global and local solutions to the lower level subproblem. Let $\mathcal{A}^k = arg\max_{t\in T} g(x^k, t)$.
3. if $g(x^k, t) \leq 0$, $\forall t \in \mathcal{A}^k$ then stop.
4. Solve NLP($\mathcal{A}^k$) and let x^{k+1} be the solution found.
5. Set $k = k + 1$ and go to step 2.

Conceptual algorithm

Reduction type algorithm

1. Let x^0 be an initial guess and $k = 0$.
2. Obtain all the global and local solutions to the lower level subproblem. Let $\mathcal{A}^k = arg\max_{t \in T} g(x^k, t)$.
3. if $g(x^k, t) \leq 0$, $\forall t \in \mathcal{A}^k$ then stop.
4. Solve NLP($\mathcal{A}^k$) and let x^{k+1} be the solution found.
5. Set $k = k + 1$ and go to step 2.

Conceptual algorithm

Reduction type algorithm

1. Let x^0 be an initial guess and $k = 0$.
2. Obtain all the global and local solutions to the lower level subproblem. Let $\mathcal{A}^k = arg\max_{t\in T} g(x^k, t)$.
3. if $g(x^k, t) \leq 0$, $\forall t \in \mathcal{A}^k$ then stop.
4. Solve NLP($\mathcal{A}^k$) and let x^{k+1} be the solution found.
5. Set $k = k + 1$ and go to step 2.

Constraint transcription

The technique

The constraint transcription is based the idea that

$$g(\bar{x},t) \le 0 \equiv \int_T [g(\bar{x},t)]_+ dt = 0$$

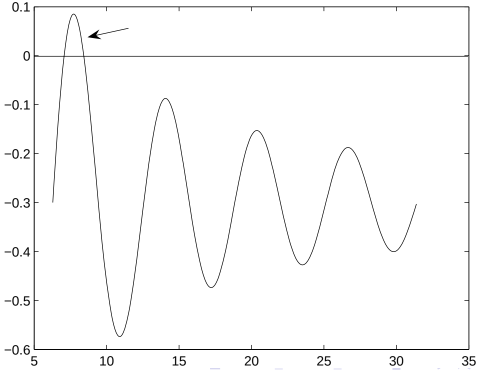
Used methods

Regular finite optimization methods can be applied.

- Penalty methods
- Interior point methods

Bad properties

LICQ is violated.



Constraint transcription

The technique

The constraint transcription is based the idea that

$$g(\bar{x}, t) \leq 0 \equiv \int_T [g(\bar{x}, t)]_+ dt = 0$$

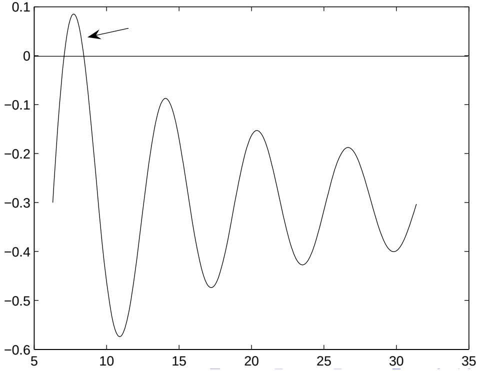
Used methods

Regular finite optimization methods can be applied.

- Penalty methods
- Interior point methods

Bad properties

LICQ is violated.



Constraint transcription

The technique

The constraint transcription is based the idea that

$$g(\bar{x}, t) \leq 0 \equiv \int_T [g(\bar{x}, t)]_+ dt = 0$$

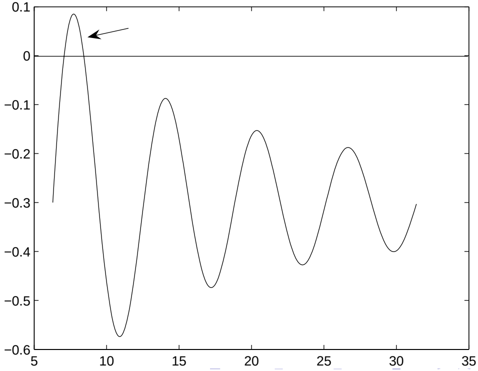
Used methods

Regular finite optimization methods can be applied.

- Penalty methods
- Interior point methods

Bad properties

LICQ is violated.



Constraint transcription

The technique

The constraint transcription is based the idea that

$$g(\bar{x}, t) \leq 0 \equiv \int_T [g(\bar{x}, t)]_+ dt = 0$$

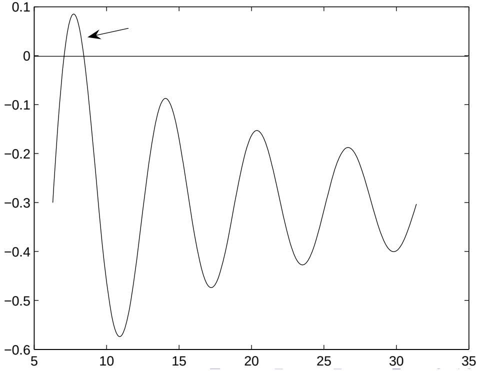
Used methods

Regular finite optimization methods can be applied.

- Penalty methods
- Interior point methods

Bad properties

LICQ is violated.



Constraint transcription

The technique

The constraint transcription is based the idea that

$$g(\bar{x},t) \leq 0 \equiv \int_T [g(\bar{x},t)]_+ dt = 0$$

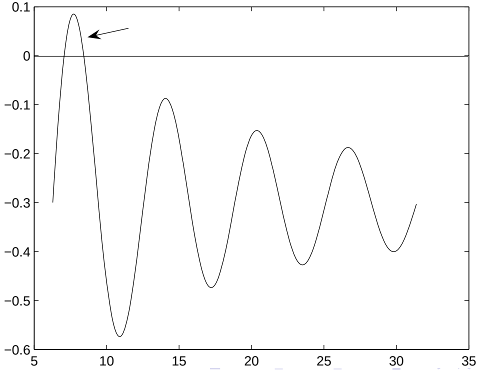
Used methods

Regular finite optimization methods can be applied.

- Penalty methods
- Interior point methods

Bad properties

LICQ is violated.



Dual type methods

A local quadratic approximation to the (NLSIP) problem is:

$$\min_{d \in R^n} f_Q(d) \equiv \frac{1}{2} d^T H^k d + d^T \nabla f(x^k)$$
$$s.t. \; d^T \nabla_x g(x^k, t) + g(x^k, t) \leq 0, \quad \forall t \in [a, b] \;,$$

where H_k is a symmetric positive definite approximation to the Lagrangian Hessian matrix.

The Lagrangian function

$$\mathcal{L}(d, v) = \frac{1}{2} d^T H^k d + d^T \nabla f(x^k) + \int_a^b \left(d^T \nabla_x g(x^k, t) + g(x^k, t) \right) dv(t)$$

Dual type methods

A local quadratic approximation to the (NLSIP) problem is:

$$\min_{d \in R^n} \; f_Q(d) \equiv \frac{1}{2} d^T H^k d + d^T \nabla f(x^k)$$
$$s.t. \;\; d^T \nabla_x g(x^k, t) + g(x^k, t) \leq 0, \quad \forall t \in [a, b] \;\; ,$$

where H_k is a symmetric positive definite approximation to the Lagrangian Hessian matrix.

The Lagrangian function

$$\mathcal{L}(d, v) = \frac{1}{2} d^T H^k d + d^T \nabla f(x^k) + \int_a^b \left(d^T \nabla_x g(x^k, t) + g(x^k, t) \right) dv(t)$$

Dual type methods

Solution method

The dual problem $\min \mathcal{L}(d, v)$ is solved by approximate the Lagrange multipliers function $v(t)$ by linear segments.

Conceptual algorithm

The local quadratic approximation is used in a sequential quadratic programming (SQP) algorithm.

Dual type methods

Solution method

The dual problem $\min \mathcal{L}(d, v)$ is solved by approximate the Lagrange multipliers function $v(t)$ by linear segments.

Conceptual algorithm

The local quadratic approximation is used in a sequential quadratic programming (SQP) algorithm.

Available solvers

Commercial software

MATLAB implements a discretization method in the optimization toolbox (`fseminf` function).

Public domain software

The Nonlinear Semi-Infinite Programming Solver (NSIPS) is publicly available. It implements:

- A discretization method (two version).
- A penalty function method (with 5 penalty functions), based on the constraint transcription technique.
- An interior point method, based on the constraint transcription technique.
- A SQP method, based on a dual local quadratic approximation.

Available solvers

Commercial software

MATLAB implements a discretization method in the optimization toolbox (`fseminf` function).

Public domain software

The Nonlinear Semi-Infinite Programming Solver (NSIPS) is publicly available. It implements:

- A discretization method (two version).
- A penalty function method (with 5 penalty functions), based on the constraint transcription technique.
- An interior point method, based on the constraint transcription technique.
- A SQP method, based on a dual local quadratic approximation.

Available solvers

Commercial software

MATLAB implements a discretization method in the optimization toolbox (`fseminf` function).

Public domain software

The Nonlinear Semi-Infinite Programming Solver (NSIPS) is publicly available. It implements:

- A discretization method (two version).
- A penalty function method (with 5 penalty functions), based on the constraint transcription technique.
- An interior point method, based on the constraint transcription technique.
- A SQP method, based on a dual local quadratic approximation.

Available solvers

Commercial software

MATLAB implements a discretization method in the optimization toolbox (`fseminf` function).

Public domain software

The Nonlinear Semi-Infinite Programming Solver (NSIPS) is publicly available. It implements:

- A discretization method (two version).
- A penalty function method (with 5 penalty functions), based on the constraint transcription technique.
- An interior point method, based on the constraint transcription technique.
- A SQP method, based on a dual local quadratic approximation.

Available solvers

Commercial software

MATLAB implements a discretization method in the optimization toolbox (`fseminf` function).

Public domain software

The Nonlinear Semi-Infinite Programming Solver (NSIPS) is publicly available. It implements:

- A discretization method (two version).
- A penalty function method (with 5 penalty functions), based on the constraint transcription technique.
- An interior point method, based on the constraint transcription technique.
- A SQP method, based on a dual local quadratic approximation.

Available solvers

Commercial software

MATLAB implements a discretization method in the optimization toolbox (`fseminf` function).

Public domain software

The Nonlinear Semi-Infinite Programming Solver (NSIPS) is publicly available. It implements:

- A discretization method (two version).
- A penalty function method (with 5 penalty functions), based on the constraint transcription technique.
- An interior point method, based on the constraint transcription technique.
- A SQP method, based on a dual local quadratic approximation.

Available tools

The SIPAMPL

To provide a database with SIP problems an extension to AMPL was developed.

SIPAMPL currently provides:

- A database with over than 160 SIP problems
- An interface between AMPL and MATLAB
- A `select` tool to allow queries to the database

Available tools

The SIPAMPL

To provide a database with SIP problems an extension to AMPL was developed.

SIPAMPL currently provides:

- A database with over than 160 SIP problems
- An interface between AMPL and MATLAB
- A `select` tool to allow queries to the database

Available tools

The SIPAMPL

To provide a database with SIP problems an extension to AMPL was developed.

SIPAMPL currently provides:

- A database with over than 160 SIP problems
- An interface between AMPL and MATLAB
- A `select` tool to allow queries to the database

Available tools

The SIPAMPL

To provide a database with SIP problems an extension to AMPL was developed.

SIPAMPL currently provides:

- A database with over than 160 SIP problems
- An interface between AMPL and MATLAB
- A `select` tool to allow queries to the database

Outline

Practical application I

Fed-batch fermentation process

Fed-batch fermentation process

- A great number of valuable products are produced using fermentation processes and thus optimizing such processes is of great economic importance.
- Fermentation modeling process involves, in general, highly nonlinear and complex differential equations.
- Often optimizing these processes results in control optimization problems for which an analytical solution is not possible.

Fed-batch fermentation process

- A great number of valuable products are produced using fermentation processes and thus optimizing such processes is of great economic importance.
- Fermentation modeling process involves, in general, highly nonlinear and complex differential equations.
- Often optimizing these processes results in control optimization problems for which an analytical solution is not possible.

Fed-batch fermentation process

- A great number of valuable products are produced using fermentation processes and thus optimizing such processes is of great economic importance.
- Fermentation modeling process involves, in general, highly nonlinear and complex differential equations.
- Often optimizing these processes results in control optimization problems for which an analytical solution is not possible.

The control problem

- The optimal control problem is described by a set of differential equations $\dot{\chi} = h(\chi, u, t), \ \ \chi(t^0) = \chi^0, \ \ t^0 \leq t \leq t^f$, where χ represent the state variables and u the control variables.
- The performance index J can be generally stated as

$$J(t^f) = \varphi(\chi(t^f), t^f) + \int_{t^0}^{t^f} \phi(\chi, u, t) dt,$$

  where φ is the performance index of the state variables at final time t^f and ϕ is the integrated performance index during the operation.
- Additional constraints that often reflet some physical limitation of the system can be imposed.

The control problem

- The optimal control problem is described by a set of differential equations $\dot{\chi} = h(\chi, u, t), \;\; \chi(t^0) = \chi^0, \;\; t^0 \leq t \leq t^f$, where χ represent the state variables and u the control variables.
- The performance index J can be generally stated as

$$J(t^f) = \varphi(\chi(t^f), t^f) + \int_{t^0}^{t^f} \phi(\chi, u, t)dt,$$

  where φ is the performance index of the state variables at final time t^f and ϕ is the integrated performance index during the operation.
- Additional constraints that often reflet some physical limitation of the system can be imposed.

The control problem

- The optimal control problem is described by a set of differential equations $\dot{\chi} = h(\chi, u, t), \;\; \chi(t^0) = \chi^0, \;\; t^0 \leq t \leq t^f$, where χ represent the state variables and u the control variables.
- The performance index J can be generally stated as

$$J(t^f) = \varphi(\chi(t^f), t^f) + \int_{t^0}^{t^f} \phi(\chi, u, t) dt,$$

  where φ is the performance index of the state variables at final time t^f and ϕ is the integrated performance index during the operation.
- Additional constraints that often reflet some physical limitation of the system can be imposed.

The control problem

The general maximization problem (P) can be posed as

problem (P)

$$\max \quad J(t^f) \tag{1}$$

$$s.t. \quad \dot{\chi} = h(\chi, u, t) \tag{2}$$

$$\underline{\chi} \leq \chi(t) \leq \overline{\chi}, \tag{3}$$

$$\underline{u} \leq u(t) \leq \overline{u}, \tag{4}$$

$$\forall t \in [t^0, t^f] \tag{5}$$

Where the state constraints (3) and control constraints (4) are to be understood as componentwise inequalities.

How we addressed problem (P)?

The control problem

The general maximization problem (P) can be posed as

problem (P)

$$\max \; J(t^f) \tag{1}$$

$$s.t. \quad \dot{\chi} = h(\chi, u, t) \tag{2}$$

$$\underline{\chi} \leq \chi(t) \leq \overline{\chi}, \tag{3}$$

$$\underline{u} \leq u(t) \leq \overline{u}, \tag{4}$$

$$\forall t \in [t^0, t^f] \tag{5}$$

Where the state constraints (3) and control constraints (4) are to be understood as componentwise inequalities.

How we addressed problem (P)?

Approaches - Fed trajectory $u(t)$ approximated by a Linear spline $w(t)$.

- Penalty function for state constraints
- The multi-local (getting all local optima) problem is easy to solve

Objective function

$$\hat{J}(t^f) = \begin{cases} J(t^f) & \text{if } \underline{\chi} \le \chi(t) \le \overline{\chi}, \\ & \forall t \in [t^0, t^f] \\ -\infty & \text{otherwise} \end{cases}$$

State constraints

$$\underline{u} \le w(t^i) \le \overline{u}, \quad i = 1, \dots, n$$

Where t^i are the spline knots.

The maximization NLP problem is

$$\max_{w(t^i)} \hat{J}(t^f), \quad s.t. \ \underline{u} \le w(t^i) \le \overline{u}, \quad i = 1, \dots, n$$

Approaches - Fed trajectory $u(t)$ approximated by a Linear spline $w(t)$.

- Penalty function for state constraints
- The multi-local (getting all local optima) problem is easy to solve

Objective function

$$\hat{J}(t^f) = \begin{cases} J(t^f) & \text{if } \underline{\chi} \le \chi(t) \le \overline{\chi}, \\ & \forall t \in [t^0, t^f] \\ -\infty & \text{otherwise} \end{cases}$$

State constraints

$$\underline{u} \le w(t^i) \le \overline{u}, \quad i = 1, \ldots, n$$

Where t^i are the spline knots.

The maximization NLP problem is

$$\max_{w(t^i)} \hat{J}(t^f), \quad s.t. \ \underline{u} \le w(t^i) \le \overline{u}, \quad i = 1, \ldots, n$$

Approaches - Fed trajectory $u(t)$ approximated by a Linear spline $w(t)$.

- Penalty function for state constraints
- The multi-local (getting all local optima) problem is easy to solve

Objective function

$$\hat{J}(t^f) = \begin{cases} J(t^f) & \text{if } \underline{\chi} \leq \chi(t) \leq \overline{\chi}, \\ & \forall t \in [t^0, t^f] \\ -\infty & \text{otherwise} \end{cases}$$

State constraints

$$\underline{u} \leq w(t^i) \leq \overline{u}, \quad i = 1, \dots, n$$

Where t^i are the spline knots.

The maximization NLP problem is

$$\max_{w(t^i)} \hat{J}(t^f), \quad s.t. \ \underline{u} \leq w(t^i) \leq \overline{u}, \quad i = 1, \dots, n$$

Approaches - Fed trajectory $u(t)$ approximated by a Cubic spline $s(t)$.

- Penalty function for state constraints
- The multi-local (getting all local optima) problem is hard to solve
- No of-the-shelf software to address this problem
- A new penalty function defined for control constraints

Objective function

$$\hat{J}(t^f) = \begin{cases} J(t^f) & \text{if } \underline{\chi} \le \chi(t) \le \overline{\chi}, \\ & \forall t \in [t^0, t^f] \\ -\infty & \text{otherwise} \end{cases}$$

New objective function

$$\bar{J}(t^f) = \begin{cases} \hat{J}(t^f) & \text{if } \underline{u} \le w(t) \le \overline{u}, \\ & \forall t \in [t^0, t^f] \\ -\infty & \text{otherwise} \end{cases}$$

Approaches - Fed trajectory $u(t)$ approximated by a Cubic spline $s(t)$.

- Penalty function for state constraints
- The multi-local (getting all local optima) problem is hard to solve
- No of-the-shelf software to address this problem
- A new penalty function defined for control constraints

Objective function

$$\hat{J}(t^f) = \begin{cases} J(t^f) & \text{if } \underline{\chi} \leq \chi(t) \leq \overline{\chi}, \\ & \forall t \in [t^0, t^f] \\ -\infty & \text{otherwise} \end{cases}$$

New objective function

$$\bar{J}(t^f) = \begin{cases} \hat{J}(t^f) & \text{if } \underline{u} \leq w(t) \leq \overline{u}, \\ & \forall t \in [t^0, t^f] \\ -\infty & \text{otherwise} \end{cases}$$

Approaches - Fed trajectory $u(t)$ approximated by a Cubic spline $s(t)$.

- Penalty function for state constraints
- The multi-local (getting all local optima) problem is hard to solve
- No of-the-shelf software to address this problem
- A new penalty function defined for control constraints

Objective function

$$\hat{J}(t^f) = \begin{cases} J(t^f) & \text{if } \underline{\chi} \leq \chi(t) \leq \overline{\chi}, \\ & \forall t \in [t^0, t^f] \\ -\infty & \text{otherwise} \end{cases}$$

New objective function

$$\bar{J}(t^f) = \begin{cases} \hat{J}(t^f) & \text{if } \underline{u} \leq w(t) \leq \overline{u}, \\ & \forall t \in [t^0, t^f] \\ -\infty & \text{otherwise} \end{cases}$$

Approaches - Fed trajectory $u(t)$ approximated by a Cubic spline $s(t)$.

- Penalty function for state constraints
- The multi-local (getting all local optima) problem is hard to solve
- No of-the-shelf software to address this problem
- A new penalty function defined for control constraints

Objective function

$$\hat{J}(t^f) = \begin{cases} J(t^f) & \text{if } \underline{\chi} \leq \chi(t) \leq \overline{\chi}, \\ & \forall t \in [t^0, t^f] \\ -\infty & \text{otherwise} \end{cases}$$

New objective function

$$\bar{J}(t^f) = \begin{cases} \hat{J}(t^f) & \text{if } \underline{u} \leq w(t) \leq \overline{u}, \\ & \forall t \in [t^0, t^f] \\ -\infty & \text{otherwise} \end{cases}$$

Implementation details

- The AMPL modeling language:
  - was used to model five optimal control problems
  - dynamic external library facility was used to solve the ordinary differentiable equations

  AMPL - A Modeling Programming Language
  www.ampl.com

- The ordinary differentiable equations were solved using the CVODE software package.

  http://www.llnl.gov/casc/sundials/

- A stochastic algorithm based on particle swarm was used to solve the non-differentiable optimization problem. We address this algorithm later on.

Implementation details

- The AMPL modeling language:
  - was used to model five optimal control problems
  - dynamic external library facility was used to solve the ordinary differentiable equations

AMPL - A Modeling Programming Language
www.ampl.com

- The ordinary differentiable equations were solved using the CVODE software package.

http://www.llnl.gov/casc/sundials/

- A stochastic algorithm based on particle swarm was used to solve the non-differentiable optimization problem. We address this algorithm later on.

Implementation details

- The AMPL modeling language:
  - was used to model five optimal control problems
  - dynamic external library facility was used to solve the ordinary differentiable equations

AMPL - A Modeling Programming Language
www.ampl.com

- The ordinary differentiable equations were solved using the CVODE software package.

http://www.llnl.gov/casc/sundials/

- A stochastic algorithm based on particle swarm was used to solve the non-differentiable optimization problem. We address this algorithm later on.

Implementation details

- The AMPL modeling language:
  - was used to model five optimal control problems
  - dynamic external library facility was used to solve the ordinary differentiable equations

AMPL - A Modeling Programming Language
`www.ampl.com`

- The ordinary differentiable equations were solved using the CVODE software package.

http://www.llnl.gov/casc/sundials/

- A stochastic algorithm based on particle swarm was used to solve the non-differentiable optimization problem. We address this algorithm later on.

Implementation details

- The AMPL modeling language:
  - was used to model five optimal control problems
  - dynamic external library facility was used to solve the ordinary differentiable equations

AMPL - A Modeling Programming Language
`www.ampl.com`

- The ordinary differentiable equations were solved using the CVODE software package.

http://www.llnl.gov/casc/sundials/

- A stochastic algorithm based on particle swarm was used to solve the non-differentiable optimization problem. We address this algorithm later on.

The problems set

- We obtained numerical results for five case studies.
- Problem
  - `penicillin` refers to a problem of fed-batch fermentation process where the optimal feed trajectory is to be computed while the penicillin production is to be maximized.
  - `ethanol` refers to a similar optimal control problem where the ethanol production is to be maximized.
  - `chemotherapy` is the only optimal control problem that does not refers to a fed-batch fermentation processe. It is a problem of drug administration in chemotherapy. The optimal trajectory to be computed is the quantity of drug that must be present in order to achieve a specified tumor reduction.
  - `hprotein` optimal control problem is to compute a unique trajectory (substrate to be fed) problem `rprotein` includes also a trajectory for an inducer. Both problems refer to a maximization for protein production.

The problems set

- We obtained numerical results for five case studies.
- Problem
  - `penicillin` refers to a problem of fed-batch fermentation process where the optimal feed trajectory is to be computed while the penicillin production is to be maximized.
  - `ethanol` refers to a similar optimal control problem where the ethanol production is to be maximized.
  - `chemotherapy` is the only optimal control problem that does not refers to a fed-batch fermentation processe. It is a problem of drug administration in chemotherapy. The optimal trajectory to be computed is the quantity of drug that must be present in order to achieve a specified tumor reduction.
  - `hprotein` optimal control problem is to compute a unique trajectory (substrate to be fed) problem `rprotein` includes also a trajectory for an inducer. Both problems refer to a maximization for protein production.

The problems set

- We obtained numerical results for five case studies.
- Problem
  - `penicillin` refers to a problem of fed-batch fermentation process where the optimal feed trajectory is to be computed while the penicillin production is to be maximized.
  - `ethanol` refers to a similar optimal control problem where the ethanol production is to be maximized.
  - `chemotherapy` is the only optimal control problem that does not refers to a fed-batch fermentation processe. It is a problem of drug administration in chemotherapy. The optimal trajectory to be computed is the quantity of drug that must be present in order to achieve a specified tumor reduction.
  - `hprotein` optimal control problem is to compute a unique trajectory (substrate to be fed) problem `rprotein` includes also a trajectory for an inducer. Both problems refer to a maximization for protein production.

The problems set

- We obtained numerical results for five case studies.
- Problem
  - `penicillin` refers to a problem of fed-batch fermentation process where the optimal feed trajectory is to be computed while the penicillin production is to be maximized.
  - `ethanol` refers to a similar optimal control problem where the ethanol production is to be maximized.
  - `chemotherapy` is the only optimal control problem that does not refers to a fed-batch fermentation processe. It is a problem of drug administration in chemotherapy. The optimal trajectory to be computed is the quantity of drug that must be present in order to achieve a specified tumor reduction.
  - `hprotein` optimal control problem is to compute a unique trajectory (substrate to be fed) problem `rprotein` includes also a trajectory for an inducer. Both problems refer to a maximization for protein production.

The problems set

- We obtained numerical results for five case studies.
- Problem
  - `penicillin` refers to a problem of fed-batch fermentation process where the optimal feed trajectory is to be computed while the penicillin production is to be maximized.
  - `ethanol` refers to a similar optimal control problem where the ethanol production is to be maximized.
  - `chemotherapy` is the only optimal control problem that does not refers to a fed-batch fermentation processe. It is a problem of drug administration in chemotherapy. The optimal trajectory to be computed is the quantity of drug that must be present in order to achieve a specified tumor reduction.
  - `hprotein` optimal control problem is to compute a unique trajectory (substrate to be fed) problem `rprotein` includes also a trajectory for an inducer. Both problems refer to a maximization for protein production.

The problems set

- We obtained numerical results for five case studies.
- Problem
  - `penicillin` refers to a problem of fed-batch fermentation process where the optimal feed trajectory is to be computed while the penicillin production is to be maximized.
  - `ethanol` refers to a similar optimal control problem where the ethanol production is to be maximized.
  - `chemotherapy` is the only optimal control problem that does not refers to a fed-batch fermentation processe. It is a problem of drug administration in chemotherapy. The optimal trajectory to be computed is the quantity of drug that must be present in order to achieve a specified tumor reduction.
  - `hprotein` optimal control problem is to compute a unique trajectory (substrate to be fed) problem `rprotein` includes also a trajectory for an inducer. Both problems refer to a maximization for protein production.

Characteristics and parameters

- The time displacement (h_i) are fixed while the optimal trajectory values are to be approximated.
- Particle swarm is a population based optimization algorithm and a population size of 60 was used with a maximum of 1000 iterations.
- Since a stochastic algorithm was used we performed 10 runs of the solver and the best solution is reported.

Characteristics and parameters

- The time displacement (h_i) are fixed while the optimal trajectory values are to be approximated.
- Particle swarm is a population based optimization algorithm and a population size of 60 was used with a maximum of 1000 iterations.
- Since a stochastic algorithm was used we performed 10 runs of the solver and the best solution is reported.

Characteristics and parameters

- The time displacement (h_i) are fixed while the optimal trajectory values are to be approximated.
- Particle swarm is a population based optimization algorithm and a population size of 60 was used with a maximum of 1000 iterations.
- Since a stochastic algorithm was used we performed 10 runs of the solver and the best solution is reported.

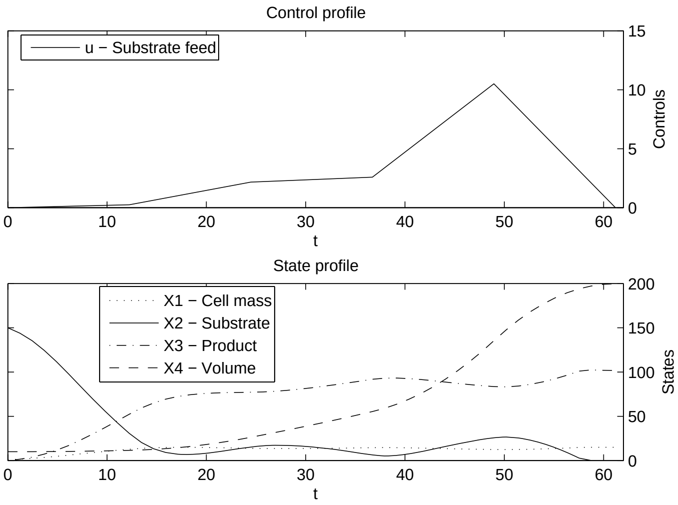
Numerical results

Problema	NT	n	t^f	Cubic $J(t^f)$	Linear $J(t^f)$	Literature $J(t^f)$
`penicillin`	1	5	132.00	87.70	88.29	87.99
`ethanol`	1	5	61.20	20550.70	20379.50	20839.00
`chemotherapy`	1	4	84.00	15.75	16.83	14.48
`hprotein`	1	5	15.00	38.86	32.73	32.40
`rprotein`	2	5	10.00	0.13	0.12	0.16

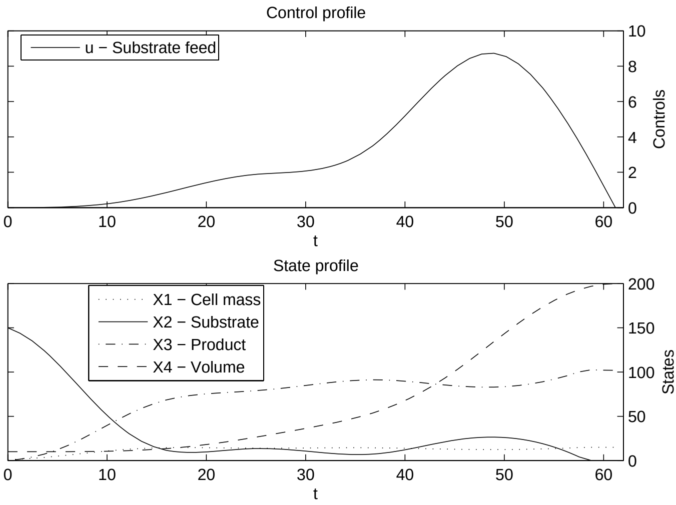
$$J(t^f) = \hat{J}(t^f) = \bar{J}(t^f), \quad \text{for all feasible points - splines}$$

Similar results between approaches. A new solution for the ethanol case.

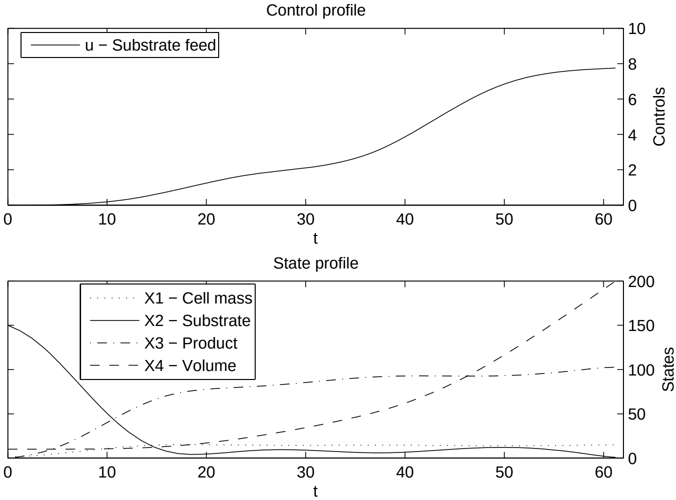
Plots - Linear spline approximation - `ethanol` case



Plots - Cubic spline approximation - Similar result



Plots - Cubic spline approximation - Best result



Practical application II

Robot trajectory planning

Robot trajectory definition

Notions:

- Links
- Joints
- Degree of freedom (d.o.f.)

Example

Two links (d_i - length, m_i - mass), three d.o.f. (θ_1 - rotation about Y, θ_2 - aperture of link 1, θ_3 - aperture of link 2)

Example

Robot trajectory definition

Notions:

- Links
- Joints
- Degree of freedom (d.o.f.)

Example

Two links (d_i - length, m_i - mass), three d.o.f. (θ_1 - rotation about Y, θ_2 - aperture of link 1, θ_3 - aperture of link 2)

Example

Robot trajectory definition

Notions:

- Links
- Joints
- Degree of freedom (d.o.f.)

Example

Two links (d_i - length, m_i - mass), three d.o.f. (θ_1 - rotation about Y, θ_2 - aperture of link 1, θ_3 - aperture of link 2)

Example

Robot trajectory definition

Notions:

- Links
- Joints
- Degree of freedom (d.o.f.)

Example

Example

Two links (d_i - length, m_i - mass), three d.o.f. (θ_1 - rotation about Y, θ_2 - aperture of link 1, θ_3 - aperture of link 2)

Robot trajectory definition

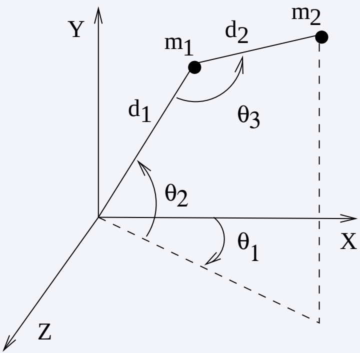
Notions:

- Links
- Joints
- Degree of freedom (d.o.f.)

Example

Two links (d_i - length, m_i - mass), three d.o.f. (θ_1 - rotation about Y, θ_2 - aperture of link 1, θ_3 - aperture of link 2)

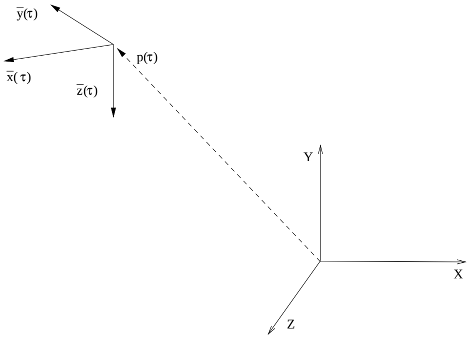
Example



Ways to defining a trajectory

In Cartesian space

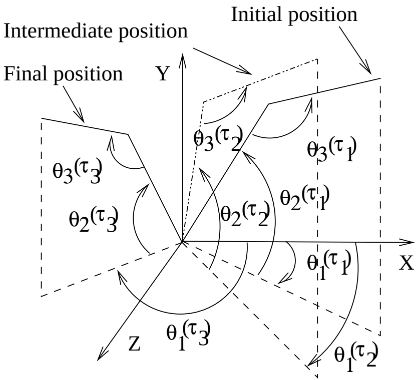
$$(\overline{x}(\tau), \overline{y}(\tau), \overline{z}(\tau), p(\tau))^T$$



Ways to defining a trajectory

In Joint space

$$\theta(\tau) = (\theta_1(\tau), \theta_2(\tau), \theta_3(\tau))^T$$



Optimize trajectory

We can optimize the trajectory for:

- **Minimum trajectory time**
- Minimum energy consumption
- Maximum load capacity

Optimize trajectory

We can optimize the trajectory for:

- **Minimum trajectory time**
- Minimum energy consumption
- Maximum load capacity

Optimize trajectory

We can optimize the trajectory for:

- **Minimum trajectory time**
- Minimum energy consumption
- Maximum load capacity

Robot limitations

- Maximum velocity in each joint $\left|\frac{d\theta_i(\tau)}{d\tau}\right| \leq C_{i,1},\ i = 1, ..., l$
- Maximum acceleration in each joint $\left|\frac{d^2\theta_i(\tau)}{d\tau^2}\right| \leq C_{i,2},\ i = 1, ..., l$
- Maximum jerk in each joint $\left|\frac{d^3\theta_i(\tau)}{d\tau^3}\right| \leq C_{i,3},\ i = 1, ..., l$

or

- Maximum joint torque in each joint

$$|F_i(\tau)| \leq C_i, \quad \tau \in [0, \tau_f], \quad i = 1, ..., l$$

Robot limitations

- Maximum velocity in each joint $\left|\frac{d\theta_i(\tau)}{d\tau}\right| \leq C_{i,1},\ i = 1, ..., l$
- Maximum acceleration in each joint $\left|\frac{d^2\theta_i(\tau)}{d\tau^2}\right| \leq C_{i,2},\ i = 1, ..., l$
- Maximum jerk in each joint $\left|\frac{d^3\theta_i(\tau)}{d\tau^3}\right| \leq C_{i,3},\ i = 1, ..., l$

or

- Maximum joint torque in each joint

$$|F_i(\tau)| \leq C_i, \quad \tau \in [0, \tau_f], \quad i = 1, ..., l$$

Robot limitations

- Maximum velocity in each joint $\left|\frac{d\theta_i(\tau)}{d\tau}\right| \leq C_{i,1},\ i = 1, \ldots, l$
- Maximum acceleration in each joint $\left|\frac{d^2\theta_i(\tau)}{d\tau^2}\right| \leq C_{i,2},\ i = 1, \ldots, l$
- Maximum jerk in each joint $\left|\frac{d^3\theta_i(\tau)}{d\tau^3}\right| \leq C_{i,3},\ i = 1, \ldots, l$

or

- Maximum joint torque in each joint

$$|F_i(\tau)| \leq C_i, \quad \tau \in [0, \tau_f], \quad i = 1, \ldots, l$$

Robot limitations

- Maximum velocity in each joint $\left|\frac{d\theta_i(\tau)}{d\tau}\right| \leq C_{i,1},\ i = 1, ..., l$
- Maximum acceleration in each joint $\left|\frac{d^2\theta_i(\tau)}{d\tau^2}\right| \leq C_{i,2},\ i = 1, ..., l$
- Maximum jerk in each joint $\left|\frac{d^3\theta_i(\tau)}{d\tau^3}\right| \leq C_{i,3},\ i = 1, ..., l$

or

- Maximum joint torque in each joint

$$|F_i(\tau)| \leq C_i, \quad \tau \in [0, \tau_f], \quad i = 1, ..., l$$

Trajectory limitations

- Robot is in movement

$$\sum_{i=1}^{l} \left(\frac{d\theta_i}{d\tau} \right)^2 > 0, \quad \tau \in (0, \tau_f)$$

- except in initial and end positions

$$\frac{d\theta}{d\tau}(0) = \frac{d\theta}{d\tau}(\tau_f) = \mathbf{0}$$

- Acceleration in initial and end positions should not be zero

$$\frac{d^2\theta}{d\tau^2}(0), \quad \frac{d^2\theta}{d\tau^2}(\tau_f) \neq \mathbf{0}$$

Trajectory limitations

- Robot is in movement

$$\sum_{i=1}^{l}\left(\frac{d\theta_i}{d\tau}\right)^2 > 0, \quad \tau \in (0,\tau_f)$$

- except in initial and end positions

$$\frac{d\theta}{d\tau}(0) = \frac{d\theta}{d\tau}(\tau_f) = \mathbf{0}$$

- Acceleration in initial and end positions should not be zero

$$\frac{d^2\theta}{d\tau^2}(0), \quad \frac{d^2\theta}{d\tau^2}(\tau_f) \neq \mathbf{0}$$

Trajectory limitations

- Robot is in movement

$$\sum_{i=1}^{l} \left(\frac{d\theta_i}{d\tau} \right)^2 > 0, \quad \tau \in (0, \tau_f)$$

- except in initial and end positions

$$\frac{d\theta}{d\tau}(0) = \frac{d\theta}{d\tau}(\tau_f) = \mathbf{0}$$

- Acceleration in initial and end positions should not be zero

$$\frac{d^2\theta}{d\tau^2}(0), \quad \frac{d^2\theta}{d\tau^2}(\tau_f) \neq \mathbf{0}$$

Optimal cubic polynomial joint trajectories

Given a set of via points defining a trajectory

Assume that $[\theta_1(\tau_0), \ldots, \theta_1(\tau_n)]$, $[\theta_2(\tau_0), \ldots, \theta_2(\tau_n)]$, $\ldots$, $[\theta_l(\tau_0), \ldots, \theta_l(\tau_n)]$ are the vectors of points (knots) where the joint trajectory passes through.

Find the best trajectory

The optimization consists of finding the optimum total displacements time that fits the joint trajectory by using cubic splines constrained to velocity, acceleration, jerk and torque bounds.

Optimal cubic polynomial joint trajectories

Given a set of via points defining a trajectory

Assume that $[\theta_1(\tau_0), \ldots, \theta_1(\tau_n)]$, $[\theta_2(\tau_0), \ldots, \theta_2(\tau_n)]$, $\ldots$, $[\theta_l(\tau_0), \ldots, \theta_l(\tau_n)]$ are the vectors of points (knots) where the joint trajectory passes through.

Find the best trajectory

The optimization consists of finding the optimum total displacements time that fits the joint trajectory by using cubic splines constrained to velocity, acceleration, jerk and torque bounds.

Additional notation

Let

- $t_0 < t_1 < \cdots < t_n$ be a time sequence where t_i is the time where the robot is in the joint position $[\theta_1(\tau_i), \ldots, \theta_l(\tau_i)]$
- $h_1 = t_1 - t_0$, $h_2 = t_2 - t_1$, $\ldots$, $h_n = t_n - t_{n-1}$ be the time displacements
- $Q_{ij}(t)$ be the cubic spline for joint i in $[t_{j-1}, t_j]$ and $Q_i(t)$ be the cubic spline for joint i.

We will use the notation $Q'(t) = \frac{dQ(t)}{dt}$ for the derivative.

Additional notation

Let

- $t_0 < t_1 < \cdots < t_n$ be a time sequence where t_i is the time where the robot is in the joint position $[\theta_1(\tau_i), \ldots, \theta_l(\tau_i)]$
- $h_1 = t_1 - t_0$, $h_2 = t_2 - t_1$, $\ldots$, $h_n = t_n - t_{n-1}$ be the time displacements
- $Q_{ij}(t)$ be the cubic spline for joint i in $[t_{j-1}, t_j]$ and $Q_i(t)$ be the cubic spline for joint i.

We will use the notation $Q'(t) = \frac{dQ(t)}{dt}$ for the derivative.

Additional notation

Let

- $t_0 < t_1 < \cdots < t_n$ be a time sequence where t_i is the time where the robot is in the joint position $[\theta_1(\tau_i), \ldots, \theta_l(\tau_i)]$
- $h_1 = t_1 - t_0$, $h_2 = t_2 - t_1$, $\ldots$, $h_n = t_n - t_{n-1}$ be the time displacements
- $Q_{ij}(t)$ be the cubic spline for joint i in $[t_{j-1}, t_j]$ and $Q_i(t)$ be the cubic spline for joint i.

We will use the notation $Q'(t) = \frac{dQ(t)}{dt}$ for the derivative.

Generalized SIP

The SIP problem can be formulated in the following mathematical form:

$$
\begin{aligned}
\min \sum_{j=1}^{n} h_j \equiv t_n - t_0 & \\
s.t. \quad & |Q_i'(t)| \le C_{i,1} \\
& |Q_i''(t)| \le C_{i,2} \\
& |Q_i'''(t)| \le C_{i,3} \\
& |F_i(t)| \le C_i, \quad i = 1, ..., l \\
& h_j > 0 \quad j = 1, ..., n; \\
& \forall t \in [t_0, t_n]
\end{aligned}
$$

where $C_{i,1}$, $C_{i,2}$, $C_{i,3}$ and C_i are the bounds for the velocity, acceleration, jerk and torque, respectively, on joint i.

Torque expression

The expression for the manipulator's torque is

$$F_i(t) = J_i n_i Q_i''(t) + B_i n_i Q_i'(t) + \frac{1}{n_i}\left(\sum_{j=1}^{l} I_{ij}(Q(t))Q_j''(t) \right.$$
$$\left. + \sum_{j=1}^{l}\sum_{k=1}^{l} C_{ijk}(Q(t))Q_j'(t)Q_k'(t) + d_i(Q(t))\right)$$

where for the ith robot joint

Torque expression (cont.)

$$
\begin{aligned}
J_i =& \text{motor inertia } (J_i > 0,\ i = 1, \ldots, l);\\
n_i =& \text{gear ratio;}\\
B_i =& \text{viscous damping coefficient } (B_i > 0,\ i = 1, \ldots, l);\\
(I_{ij}(Q(t)))_{i,j=1,\ldots,l} =& \text{inertia matrix (positive definite);}\\
(C_{ijk}(Q(t)))_{i,j,k=1,\ldots,l} =& \text{Coriolis tensor;}\\
d_i(Q(t)) =& \text{gravitational torque.}
\end{aligned}
$$

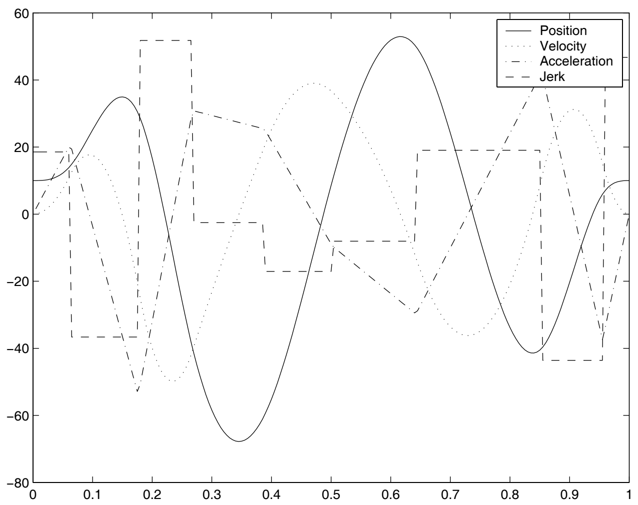
Reformulation as standard SIP

$$
\begin{aligned}
\min & \sum_{j=1}^{n} h_j \\
s.t. & \left| Q_i'\left(\tau \sum_{k=1}^{n} h_k + t_0\right)\right| \leq C_{i,1} \\
& \left| Q_i''\left(\tau \sum_{k=1}^{n} h_k + t_0\right)\right| \leq C_{i,2} \\
& \left| Q_i'''\left(\tau \sum_{k=1}^{n} h_k + t_0\right)\right| \leq C_{i,3} \\
& \left| F_i\left(\tau \sum_{k=1}^{n} h_k + t_0\right)\right| \leq C_i, \quad i = 1, \ldots, l \\
& h_j > 0, \quad j = 1, \ldots, n, \quad \forall \tau \in [0, 1].
\end{aligned}
$$

using the linear transformation $t = \tau \sum_{k=1}^{n} h_k + t_0$.

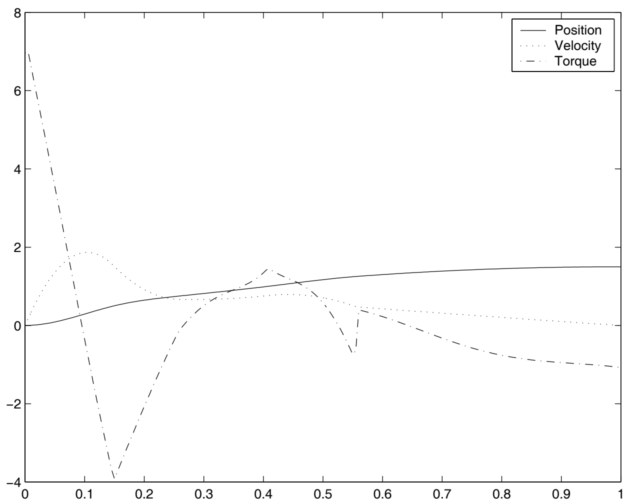
Some results with problems available at SIPAMPL

Plot - Joint 5 for problem `lin2`



Some more results

Plot - Joint 1 for problem `deluca1`



Optimal parametrization of curves for robot joint trajectories

Another trajectory optimization problem

The robot trajectory is known $(\theta_i(\tau))$ and a parametrization $(t = h(\tau))$ is to be computed.

Find a parametrization

$$t = h(\tau), \quad \tau \in [0, 1] \qquad \tau_f = 1$$

where $\theta_i^*(t) = \theta_i(h^{-1}(t))$, such that

Model 1	*Model 2*
$h(0) = 0$ $h(1)$ is minimum $h'(\tau) > 0, \quad \tau \in [0, 1]$	
$\left\lvert \frac{d\theta_i^*(t)}{dt} \right\rvert \leq C_{i,1}$ $\left\lvert \frac{d^2\theta_i^*(t)}{dt^2} \right\rvert \leq C_{i,2}$ $\left\lvert \frac{d^3\theta_i^*(t)}{dt^3} \right\rvert \leq C_{i,3}$	$\lvert F_i(t) \rvert \leq C_i$
$i = 1, \ldots, l$	

Optimal parametrization of curves for robot joint trajectories

Another trajectory optimization problem

The robot trajectory is known $(\theta_i(\tau))$ and a parametrization $(t = h(\tau))$ is to be computed.

Find a parametrization

$$t = h(\tau), \quad \tau \in [0,1] \quad \tau_f = 1$$

where $\theta_i^*(t) = \theta_i(h^{-1}(t))$, such that

Model 1	*Model 2*
$h(0) = 0$ $h(1)$ is minimum $h'(\tau) > 0, \quad \tau \in [0,1]$	
$\left\lvert \frac{d\theta_i^*(t)}{dt} \right\rvert \leq C_{i,1}$ $\left\lvert \frac{d^2\theta_i^*(t)}{dt^2} \right\rvert \leq C_{i,2}$ $\left\lvert \frac{d^3\theta_i^*(t)}{dt^3} \right\rvert \leq C_{i,3}$	$\lvert F_i(t) \rvert \leq C_i$
$i = 1, \ldots, l$	

Using B-splines

The objective is to find a parametrization ($h(\tau)$) that minimizes the total time travel.

Let

$$g(\tau) = h'(\tau)$$

be approximated by a B-Spline ($B_{k,\xi}(\tau)$)

The total time travel is simply the integral of the parametric curve:

$$\int_0^1 g(\tau) d\tau = \int_0^1 B_{k,\xi}(\tau) d\tau = \frac{1}{k} \sum_{i=1}^{n} x_i \left(\xi_{i+k} - \xi_i \right)$$

The complete problem formulation

Model 1

$$\min_{x\in R^n} \int_0^1 g(\tau)d\tau$$

$$s.t.\ \ g(\tau) > 0$$

$$\left|\frac{d\theta_i^*}{dt}\right| \leq C_{i,1}$$

$$\left|\frac{d^2\theta_i^*}{dt^2}\right| \leq C_{i,2}$$

$$\left|\frac{d^3\theta_i^*}{dt^3}\right| \leq C_{i,3} \quad i = 1, ..., l$$

$$\forall \tau \in [0, 1]\ \ ,$$

Model 2

$$\min_{x\in R^n} \int_0^1 g(\tau)d\tau$$

$$s.t.\ \ g(\tau) > 0$$

$$|F_i| \leq C_i \quad i = 1, ..., l$$

$$\forall \tau \in [0, 1]\ \ .$$

The complete problem formulation

Model 1

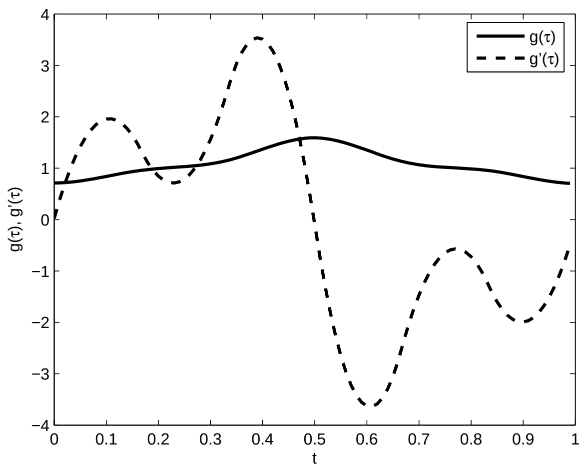
$$
\begin{aligned}
&\min_{x\in R^n} \int_0^1 g(\tau)d\tau \\
&s.t.\ \ g(\tau) > 0 \\
&\qquad \left|\frac{d\theta_i^*}{dt}\right| \le C_{i,1} \\
&\qquad \left|\frac{d^2\theta_i^*}{dt^2}\right| \le C_{i,2} \\
&\qquad \left|\frac{d^3\theta_i^*}{dt^3}\right| \le C_{i,3} \quad i = 1, \dots, l \\
&\qquad \forall \tau \in [0,1] \ ,
\end{aligned}
$$

Model 2

$$
\begin{aligned}
&\min_{x\in R^n} \int_0^1 g(\tau)d\tau \\
&s.t.\ \ g(\tau) > 0 \\
&\qquad |F_i| \le C_i \quad i = 1, \dots, l \\
&\qquad \forall \tau \in [0,1] \ .
\end{aligned}
$$

Some results with problems available at SIPAMPL

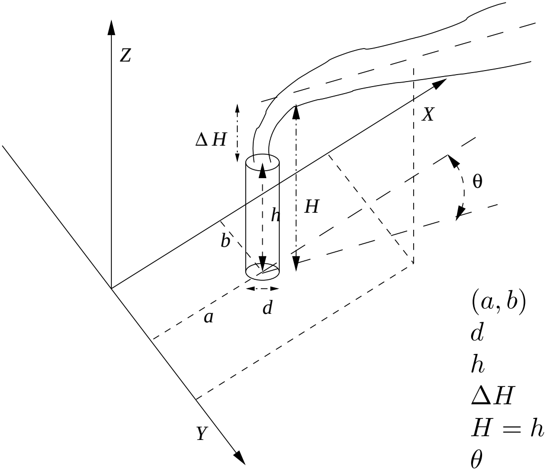
Plot - $g(\tau)$ and $g'(\tau)$ for problem `elke1std` (Model 1)



Practical application III

Air pollution control

Coordinate system



(a, b)	stack position
d	stack internal diameter
h	stack height
ΔH	plume rise
$H = h + \Delta H$	effective stack height
θ	mean wind direction

Dispersion model

Assuming that the plume has a Gaussian distribution, the concentration, of gas or aerosol (particles with diameter less than 20 microns) at position x, y and z of a continuous source with effective stack height $\mathcal{H}$, is given by

$$\mathcal{C}(x,y,z,\mathcal{H}) = \frac{\mathcal{Q}}{2\pi\sigma_y\sigma_z\mathcal{U}} e^{-\frac{1}{2}\left(\frac{\mathcal{Y}}{\sigma_y}\right)^2} \left(e^{-\frac{1}{2}\left(\frac{z-\mathcal{H}}{\sigma_z}\right)^2} + e^{-\frac{1}{2}\left(\frac{z+\mathcal{H}}{\sigma_z}\right)^2} \right)$$

where $\mathcal{Q}$ (gs^{-1}) is the pollution uniform emission rate, $\mathcal{U}$ (ms^{-1}) is the mean wind speed affecting the plume, σ_y (m) and σ_z (m) are the standard deviations in the horizontal and vertical planes, respectively.

Change of coordinates

The source change of coordinates to position (a, b), in the wind direction.
$\mathcal{Y}$ is given by

$$\mathcal{Y} = (x - a)\sin(\theta) + (y - b)\cos(\theta),$$

where θ (rad) is the wind direction $(0 \leq \theta \leq 2\pi)$.
σ_y and σ_z depend on $\mathcal{X}$ given by

$$\mathcal{X} = (x - a)\cos(\theta) - (y - b)\sin(\theta).$$

Plume rise

The effective emission height is the sum of the stack height, h (m), with the plume rise, $\Delta\mathcal{H}$ (m). The considered elevation is given by the Holland equation

$$\Delta\mathcal{H} = \frac{V_o d}{\mathcal{U}}\left(1.5 + 2.68\frac{T_o - T_e}{T_o}d\right),$$

where d (m) is the internal stack diameter, V_o (ms^{-1}) is the gas out velocity, T_o (K) is the gas temperature and T_e (K) is the environment temperature.

Formulations

- Assuming n pollution sources distributed in a region;
- C_i is the source i contribution for the total concentration;
- Gas chemical inert.

We can derive three formulations:

- Minimize the stack height;
- Maximum pollution computation and sampling stations planning;
- Air pollution abatement.

Formulations

- Assuming n pollution sources distributed in a region;
- $\mathcal{C}_i$ is the source i contribution for the total concentration;
- Gas chemical inert.

We can derive three formulations:

- Minimize the stack height;
- Maximum pollution computation and sampling stations planning;
- Air pollution abatement.

Formulations

- Assuming n pollution sources distributed in a region;
- $\mathcal{C}_i$ is the source i contribution for the total concentration;
- Gas chemical inert.

We can derive three formulations:

- Minimize the stack height;
- Maximum pollution computation and sampling stations planning;
- Air pollution abatement.

Formulations

- Assuming n pollution sources distributed in a region;
- $\mathcal{C}_i$ is the source i contribution for the total concentration;
- Gas chemical inert.

We can derive three formulations:

- Minimize the stack height;
- Maximum pollution computation and sampling stations planning;
- Air pollution abatement.

Formulations

- Assuming n pollution sources distributed in a region;
- $\mathcal{C}_i$ is the source i contribution for the total concentration;
- Gas chemical inert.

We can derive three formulations:

- Minimize the stack height;
- Maximum pollution computation and sampling stations planning;
- Air pollution abatement.

Formulations

- Assuming n pollution sources distributed in a region;
- $\mathcal{C}_i$ is the source i contribution for the total concentration;
- Gas chemical inert.

We can derive three formulations:

- Minimize the stack height;
- Maximum pollution computation and sampling stations planning;
- Air pollution abatement.

Minimum stack height

Minimizing the stack height $u = (h_1, \dots, h_n)$, while the pollution ground pollution level is kept below a given threshold $\mathcal{C}_0$, in a given region $\mathcal{R}$, can be formulated as a SIP problem

$$\min_{u \in R^n} \sum_{i=1}^{n} c_i h_i$$

$$s.t.\;\; g(u, v \equiv (x,y)) \equiv \sum_{i=1}^{n} \mathcal{C}_i(x, y, 0, \mathcal{H}_i) \leq \mathcal{C}_0$$

$$\forall v \in \mathcal{R} \subset R^2,$$

where c_i, $i = 1, \dots, n$, are the construction costs.

Note: more complex objective function can be considered.

Maximum pollution and sampling stations planning

The maximum pollution concentration (l^*) in a given region can be obtained by solving the following SIP problem

$$\begin{aligned} &\min_{l \in R} \; l \\ s.t. \;\; & g(z, v \equiv (x,y)) \equiv \sum_{i=1}^{n} \mathcal{C}_i(x, y, 0, \mathcal{H}_i) \leq l \\ & \forall v \in \mathcal{R} \subset R^2. \end{aligned}$$

The active points $v^* \in \mathcal{R}$ where $g(z^*, v^*) = l^*$ are the global optima and indicate where the sampling (control) stations should be placed.

Air pollution abatement

Minimizing the pollution abatement (minimizing clean costs, maximizing the revenue, minimizing the economical impact) while the air pollution concentration is kept below a given threshold can be posed as a SIP problem

$$\begin{aligned} &\min_{u \in R^n} \sum_{i=1}^{n} p_i r_i \\ s.t.\ \ &g(u, v \equiv (x,y)) \equiv \sum_{i=1}^{n} (1 - r_i)\mathcal{C}_i(x, y, 0, \mathcal{H}_i) \leq \mathcal{C}_0 \\ &\forall v \in \mathcal{R} \subset R^2, \end{aligned}$$

where $u = (r_1, \ldots, r_n)$ is the pollution reduction and p_i, $i = 1, \ldots, n$, is the source i cost (cleaning or not producing).

Numerical results – Minimum stack height (vaz1)

	Instance 1	Instance 2	Instance 3
h_1	0.00	10.00	196.93
h_2	78.26	69.09	380.06
h_3	0.00	10.00	403.12
h_4	153.17	152.64	428.38
h_5	80.90	71.27	344.81
h_6	0.00	10.00	274.58
h_7	13.52	13.52	402.83
h_8	161.78	161.87	396.82
h_9	141.73	141.63	415.58
h_{10}	15.05	15.05	423.99
Total	644.40	655.06	3667.10

Instance 1 – no limit on stack, Instance 2 – limit of 10m, Instance 3 – Portuguese legislation.

Outline

We intended to solve the following global optimization problem with a particle swarm algorithm.

Global optimization problem

$$\max_{t \in T} \bar{g}(t) \equiv g(\bar{x}, t)$$

with $T \in R^p$.

The Particle Swarm Paradigm (PSP)

The PSP is a population (swarm) based algorithm that mimics the social behavior of a set of individuals (particles).

An individual behavior is a combination of its past experience (cognition influence) and the society experience (social influence).

In the optimization context a particle $\wp$, at time instant k, is represented by its current position ($t^{\wp}(k)$), its best ever position ($y^{\wp}(k)$) and its traveling velocity ($v^{\wp}(k)$).

The Particle Swarm Paradigm (PSP)

The PSP is a population (swarm) based algorithm that mimics the social behavior of a set of individuals (particles).

An individual behavior is a combination of its past experience (cognition influence) and the society experience (social influence).

In the optimization context a particle $\wp$, at time instant k, is represented by its current position ($t^{\wp}(k)$), its best ever position ($y^{\wp}(k)$) and its traveling velocity ($v^{\wp}(k)$).

The Particle Swarm Paradigm (PSP)

The PSP is a population (swarm) based algorithm that mimics the social behavior of a set of individuals (particles).

An individual behavior is a combination of its past experience (cognition influence) and the society experience (social influence).

In the optimization context a particle $\wp$, at time instant k, is represented by its current position ($t^{\wp}(k)$), its best ever position ($y^{\wp}(k)$) and its traveling velocity ($v^{\wp}(k)$).

The new travel position and velocity

The new particle position is updated by

Update position

$$t^{\wp}(k+1) = t^{\wp}(k) + v^{\wp}(k+1),$$

where $v^{\wp}(k+1)$ is the new velocity given by

Update velocity

$$v_j^{\wp}(k+1) = \iota(k) v_j^{\wp}(k) + \mu \omega_{1j}(k)\left(y_j^{\wp}(k) - t_j^{\wp}(k)\right) + \nu \omega_{2j}(k)\left(\hat{y}_j(k) - t_j^{\wp}(k)\right),$$

for $j = 1, \ldots, p$.

- $\iota(k)$ is a weighting factor (inertial)
- μ is the cognition parameter and ν is the social parameter
- $\omega_{1j}(k)$ and $\omega_{2j}(k)$ are random numbers drawn from the uniform (0,1) distribution.

The new travel position and velocity

The new particle position is updated by

Update position

$$t^{\wp}(k+1) = t^{\wp}(k) + v^{\wp}(k+1),$$

where $v^{\wp}(k+1)$ is the new velocity given by

Update velocity

$$v_j^{\wp}(k+1) = \iota(k) v_j^{\wp}(k) + \mu\omega_{1j}(k)\left(y_j^{\wp}(k) - t_j^{\wp}(k)\right) + \nu\omega_{2j}(k)\left(\hat{y}_j(k) - t_j^{\wp}(k)\right),$$

for $j = 1, \dots, p$.

- $\iota(k)$ is a weighting factor (inertial)
- μ is the *cognition* parameter and ν is the *social* parameter
- $\omega_{1j}(k)$ and $\omega_{2j}(k)$ are random numbers drawn from the uniform (0, 1) distribution

The new travel position and velocity

The new particle position is updated by

Update position

$$t^{\wp}(k+1) = t^{\wp}(k) + v^{\wp}(k+1),$$

where $v^{\wp}(k+1)$ is the new velocity given by

Update velocity

$$v_j^{\wp}(k+1) = \iota(k) v_j^{\wp}(k) + \mu \omega_{1j}(k) \left(y_j^{\wp}(k) - t_j^{\wp}(k) \right) + \nu \omega_{2j}(k) \left(\hat{y}_j(k) - t_j^{\wp}(k) \right),$$

for $j = 1, \ldots, p$.

- $\iota(k)$ is a weighting factor (inertial)
- μ is the *cognition* parameter and ν is the *social* parameter
- $\omega_{1j}(k)$ and $\omega_{2j}(k)$ are random numbers drawn from the uniform $(0, 1)$ distribution.

The new travel position and velocity

The new particle position is updated by

Update position

$$t^{\wp}(k+1) = t^{\wp}(k) + v^{\wp}(k+1),$$

where $v^{\wp}(k+1)$ is the new velocity given by

Update velocity

$$v_j^{\wp}(k+1) = \iota(k)v_j^{\wp}(k) + \mu\omega_{1j}(k)\left(y_j^{\wp}(k) - t_j^{\wp}(k)\right) + \nu\omega_{2j}(k)\left(\hat{y}_j(k) - t_j^{\wp}(k)\right),$$

for $j = 1, \ldots, p$.

- $\iota(k)$ is a weighting factor (inertial)
- μ is the *cognition* parameter and ν is the *social* parameter
- $\omega_{1j}(k)$ and $\omega_{2j}(k)$ are random numbers drawn from the uniform $(0,1)$ distribution.

The new travel position and velocity

The new particle position is updated by

Update position

$$t^{\wp}(k+1) = t^{\wp}(k) + v^{\wp}(k+1),$$

where $v^{\wp}(k+1)$ is the new velocity given by

Update velocity

$$v^{\wp}_j(k+1) = \iota(k) v^{\wp}_j(k) + \mu\omega_{1j}(k)\left(y^{\wp}_j(k) - t^{\wp}_j(k)\right) + \nu\omega_{2j}(k)\left(\hat{y}_j(k) - t^{\wp}_j(k)\right),$$

for $j = 1, \ldots, p$.

- $\iota(k)$ is a weighting factor (inertial)
- μ is the *cognition* parameter and ν is the *social* parameter
- $\omega_{1j}(k)$ and $\omega_{2j}(k)$ are random numbers drawn from the uniform $(0,1)$ distribution.

The best ever particle

$\hat{y}(k)$ is a particle position with global best function value so far, *i.e.*,

Best position

$$\hat{y}(k) \in \arg\min_{a \in \mathcal{A}} \bar{g}(a)$$
$$\mathcal{A} = \left\{ y^1(k), \ldots, y^s(k) \right\}.$$

where s is the number of particles in the swarm.

Note

In an algorithmic point of view we just have to keep track of the particle with the best ever function value.

The best ever particle

$\hat{y}(k)$ is a particle position with global best function value so far, *i.e.*,

Best position

$$\hat{y}(k) \in \arg\min_{a \in \mathcal{A}} \bar{g}(a)$$
$$\mathcal{A} = \left\{ y^1(k), \ldots, y^s(k) \right\}.$$

where s is the number of particles in the swarm.

Note

In an algorithmic point of view we just have to keep track of the particle with the best ever function value.

Features

Population based algorithm.

- Good
  - Easy to implement.
  - Easy to parallelize.
  - Easy to handle discrete variables.
  - Only uses objective function evaluations.
- Not so good
  - Slow rate of convergence near an optimum.
  - Quite large number of function evaluations.
  - In the presence of several global optima the algorithm may not converge.

Features

Population based algorithm.

- Good
  - Easy to implement.
  - Easy to parallelize.
  - Easy to handle discrete variables.
  - Only uses objective function evaluations.
- Not so good
  - Slow rate of convergence near an optimum.
  - Quite large number of function evaluations.
  - In the presence of several global optima the algorithm may not converge.

Features

Population based algorithm.

- Good
  - Easy to implement.
  - Easy to parallelize.
  - Easy to handle discrete variables.
  - Only uses objective function evaluations.
- Not so good
  - Slow rate of convergence near an optimum.
  - Quite large number of function evaluations.
  - In the presence of several global optima the algorithm may not converge.

Features

Population based algorithm.

- Good
  - Easy to implement.
  - Easy to parallelize.
  - Easy to handle discrete variables.
  - Only uses objective function evaluations.
- Not so good
  - Slow rate of convergence near an optimum.
  - Quite large number of function evaluations.
  - In the presence of several global optima the algorithm may not converge.

Features

Population based algorithm.

- Good
  - Easy to implement.
  - Easy to parallelize.
  - Easy to handle discrete variables.
  - Only uses objective function evaluations.
- Not so good
  - Slow rate of convergence near an optimum.
  - Quite large number of function evaluations.
  - In the presence of several global optima the algorithm may not converge.

Features

Population based algorithm.

- Good
  - Easy to implement.
  - Easy to parallelize.
  - Easy to handle discrete variables.
  - Only uses objective function evaluations.
- Not so good
  - Slow rate of convergence near an optimum.
  - Quite large number of function evaluations.
  - In the presence of several global optima the algorithm may not converge.

Features

Population based algorithm.

- Good
  - Easy to implement.
  - Easy to parallelize.
  - Easy to handle discrete variables.
  - Only uses objective function evaluations.
- Not so good
  - Slow rate of convergence near an optimum.
  - Quite large number of function evaluations.
  - In the presence of several global optima the algorithm may not converge.

Features

Population based algorithm.

- Good
  - Easy to implement.
  - Easy to parallelize.
  - Easy to handle discrete variables.
  - Only uses objective function evaluations.
- Not so good
  - Slow rate of convergence near an optimum.
  - Quite large number of function evaluations.
  - In the presence of several global optima the algorithm may not converge.

Features

Population based algorithm.

- Good
  - Easy to implement.
  - Easy to parallelize.
  - Easy to handle discrete variables.
  - Only uses objective function evaluations.
- Not so good
  - Slow rate of convergence near an optimum.
  - Quite large number of function evaluations.
  - In the presence of several global optima the algorithm may not converge.

Properties

- With a proper selection of the algorithm parameters finite termination of the algorithm can be established, in a probabilistic sense.
- Convergence for a global optimum is not guaranteed by this simple version of the particle swarm algorithm, but some adaption can be introduce to guarantee it.

Properties

- With a proper selection of the algorithm parameters finite termination of the algorithm can be established, in a probabilistic sense.
- Convergence for a global optimum is not guaranteed by this simple version of the particle swarm algorithm, but some adaption can be introduce to guarantee it.

Outline

Multi-local revisited

Given $\bar{x}$ the multi-local optimization problem is defined as

Multi-local optimization problem

$$\max_{t \in T} g(\bar{x}, t) \equiv \bar{g}(t)$$

with $T \in R^n$.

The multi-local concept

All the global and local optima are to be computed.

Some characteristics

These problems are mostly differentiable and the objective function computation is costless.

Multi-local revisited

Given $\bar{x}$ the multi-local optimization problem is defined as

Multi-local optimization problem

$$\max_{t \in T} g(\bar{x}, t) \equiv \bar{g}(t)$$

with $T \in R^n$.

The multi-local concept

All the global and local optima are to be computed.

Some characteristics

These problems are mostly differentiable and the objective function computation is costless.

Multi-local revisited

Given $\bar{x}$ the multi-local optimization problem is defined as

Multi-local optimization problem

$$\max_{t \in T} g(\bar{x}, t) \equiv \bar{g}(t)$$

with $T \in R^n$.

The multi-local concept

All the global and local optima are to be computed.

Some characteristics

These problems are mostly differentiable and the objective function computation is costless.

PSP with the steepest ascent (quasi-Newton) direction

The new particle position update equation is kept while the new velocity equation is given by

Steepest ascent velocity

$$v_j^{\wp}(k+1) = \iota(k) v_j^{\wp}(k) + \mu\omega_{1j}(k)\left(y_j^{\wp}(k) - t_j^{\wp}(k)\right) + \nu\omega_{2j}(t)\left(\nabla_j \bar{g}(y_j^{\wp}(k))\right),$$

for $j = 1, \ldots, p$, where $\nabla \bar{g}(t)$ is the gradient of the objective function.

Each particle uses the steepest ascent direction computed at each particle best position ($y^{\wp}(k)$).

The inclusion of the steepest ascent direction in the velocity equation aims to drive each particle to a neighbor local maximum and since we have a population of particles, each one will be driven to a local maximum.

PSP with an ascent direction

Other approach is to use

Ascent velocity formula

$$w^{\wp} = \frac{1}{\sum_{j=1}^{m} |\bar{g}(z_j^{\wp}) - \bar{g}(y^{\wp})|} \sum_{j=1}^{m} (\bar{g}(z_j^{\wp}) - \bar{g}(y^{\wp})) \frac{(z_j^{\wp} - y^{\wp})}{\|z_j^{\wp} - y^{\wp}\|}$$

as an ascent direction at $y^{\wp}$, in the velocity equation, to overcome the need to compute the gradient.

Where

- $y^{\wp}$ is the best position of particle $\wp$
- $\{z_j^{\wp}\}_{j=1}^{m}$ is a set of m (random) points close to y^p,

Under certain conditions $w^{\wp}$ simulates the steepest ascent direction.

Stopping criterion

We propose the stopping criterion

Minimum velocity attained

$$\max_{\wp}[v^{\wp}(k)]_{opt} \leq \epsilon_{\wp}$$

where

Constrained velocity

$$[v^{\wp}(k)]_{opt} = \left(\sum_{j=1}^{p} \begin{cases} 0 & \text{if } t_j^{\wp}(k) = \beta_j \text{ and } v_j^{\wp}(k) \geq 0 \\ 0 & \text{if } t_j^{\wp}(k) = \alpha_j \text{ and } v_j^{\wp}(k) \leq 0 \\ \left(v_j^{\wp}(k)\right)^2 & \text{otherwise} \end{cases} \right)^{1/2}$$

The stopping criterion is based on the optimality conditions for the multi-local optimization problem.

Numerical results

	Gradient version					Approximate descent direction version			
	F.O.	N_{afe}	N_{age}	g_a^*	g_{best}	F.O.	N_{afe}	g_a^*	g_{best}
17	0	10000000	177	4,652E+03	2,393E+03	40	10005589	2,203E-01	1,327E-01
18	100	10000000	1850	-9,160E+00	-1,026E+01	100	10004066	-1,052E+01	-1,052E+01
19	100	10000000	2126	-7,801E+00	-8,760E+00	100	10003906	-1,012E+01	-1,014E+01
20	100	10000000	1909	-9,401E+00	-9,997E+00	100	10004069	-1,037E+01	-1,039E+01
21	0	3600000	335	-1,024E+02	-1,648E+02	60	3600999	-1,867E+02	-1,867E+02
22	100	1366222	973	-4,075E-01	-4,075E-01	100	3600804	-4,075E-01	-4,075E-01
23	100	3600000	570	-1,806E+01	-1,806E+01	100	3600902	-1,806E+01	-1,806E+01
24	100	3600000	194	-2,278E+02	-2,278E+02	100	3601003	-2,278E+02	-2,278E+02
25	100	3600000	167	-2,429E+03	-2,429E+03	100	3601160	-2,429E+03	-2,429E+03
26	90	3600000	81	-2,477E+04	-2,478E+04	100	3601278	-2,478E+04	-2,478E+04
27	10	3600000	58	1,607E+05	-2,436E+05	100	3601418	-2,493E+05	-2,493E+05
28	0	10000000	141	4,470E+02	3,102E+01	60	10009759	3,977E-02	2,506E-02
29	0	10000000	135	1,289E+05	7,935E+02	0	10016905	3,633E-01	2,404E-01
30	100	1433664	16314	8,325E-112	0,000E+00	100	3601264	4,987E-07	4,464E-08
31	100	10000000	313	1,997E-13	2,780E-21	100	10005221	2,231E-04	6,612E-05
32	40	10000000	160	8,338E+00	3,031E-04	100	10006065	2,005E-03	1,186E-03

The test set for SIP

- The test problems were obtained from SIP where $\bar{x}$ was replaced by x^*, where x^* is the SIP solution included in the SIPAMPL database. SIPAMPL stands for SIP with AMPL and is a software package that provides, among other features, a database of SIP coded problems.
- All SIP problems considered have only one infinite constraint.

SIP problem	Test problem	p	Obs
`watson2`	`sip_wat2`	1	Unidimensional
`vaz3`	`sip_vaz3`	2	Air pollution abatement
`priceS6`	`sip_S6`	6	Higher dimension in SIPAMPL
`priceU`	`sip_U`	6	Higher dimension in SIPAMPL
`random`	`sip_rand`	6	Random generated with known solution

The test set for SIP

- The test problems were obtained from SIP where $\bar{x}$ was replaced by x^*, where x^* is the SIP solution included in the SIPAMPL database. SIPAMPL stands for SIP with AMPL and is a software package that provides, among other features, a database of SIP coded problems.
- All SIP problems considered have only one infinite constraint.

SIP problem	Test problem	p	Obs
`watson2`	`sip_wat2`	1	Unidimensional
`vaz3`	`sip_vaz3`	2	Air pollution abatement
`priceS6`	`sip_S6`	6	Higher dimension in SIPAMPL
`priceU`	`sip_U`	6	Higher dimension in SIPAMPL
`random`	`sip_rand`	6	Random generated with known solution

Numerical results

- A population of 40 particles and a maximum of 2000 iterations was used, with the steepest ascent direction version.
- sip_wat2 a global and a local maxima were found. 10 particles converged to the local maxima $t = 1$ with $\bar{g}(1) = -0.058594$ and the remaining 30 to the global one ($t = 0$) with $\bar{g}(0) = -2.5156e-08$
- In sip_vaz3 the objective function is flat (equal to zero) in a subregion.

t	$\bar{g}(t)$	$npar$
$(-0.783012, 2.172526)$	0.000000	1
$(-0.112199, -0.686259)$	0.000000	1
$(-0.278460, 0.095245)$	0.000000	1
$(-0.446057, 1.157275)$	0.000000	1
$(0.443709, 3.811052)$	0.000000	1
$(3.684002, -0.629689)$	0.500007	22
$(1.099826, 0.112477)$	0.500055	13

Numerical results

- A population of 40 particles and a maximum of 2000 iterations was used, with the steepest ascent direction version.
- sip_wat2 a global and a local maxima were found. 10 particles converged to the local maxima $t = 1$ with $\bar{g}(1) = -0.058594$ and the remaining 30 to the global one ($t = 0$) with $\bar{g}(0) = -2.5156e - 08$
- In sip_vaz3 the objective function is flat (equal to zero) in a subregion.

t	$\bar{g}(t)$	$npar$
$(-0.783012, 2.172526)$	0.000000	1
$(-0.112199, -0.686259)$	0.000000	1
$(-0.278460, 0.095245)$	0.000000	1
$(-0.446057, 1.157275)$	0.000000	1
$(0.443709, 3.811052)$	0.000000	1
$(3.684002, -0.629689)$	0.500007	22
$(1.099826, 0.112477)$	0.500055	13

Numerical results

- A population of 40 particles and a maximum of 2000 iterations was used, with the steepest ascent direction version.
- `sip_wat2` a global and a local maxima were found. 10 particles converged to the local maxima $t = 1$ with $\bar{g}(1) = -0.058594$ and the remaining 30 to the global one ($t = 0$) with $\bar{g}(0) = -2.5156e-08$
- In `sip_vaz3` the objective function is flat (equal to zero) in a subregion.

t	$\bar{g}(t)$	$npar$
$(-0.783012, 2.172526)$	0.000000	1
$(-0.112199, -0.686259)$	0.000000	1
$(-0.278460, 0.095245)$	0.000000	1
$(-0.446057, 1.157275)$	0.000000	1
$(0.443709, 3.811052)$	0.000000	1
$(3.684002, -0.629689)$	0.500007	22
$(1.099826, 0.112477)$	0.500055	13

Numerical results

- sip_S6 a reported global maximizer and two local with objective function values of 0.027092, -3.69008 and -1.95425 respectively.

t	$\bar{g}(t)$
...	
$(1.622134, 1.687810, 2.000000, 0.085439, 2.000000, 0.350174)$	0.024811
...	
$(1.634326, 1.671065, 2.000000, 0.054348, 2.000000, 2.000000)$	-1.954538
...	

- sip_U reported two global maximizers and eleven local maximizers

t	$\bar{g}(t)$	$npar$
(-0.665555,-1.000000,1.00,1.00,1.00,1.00)	-0.002587	1
(-0.689138,-0.933410,1.00,1.00,1.00,1.00)	-0.003319	1
(-0.890160,-1.000000,1.00,1.00,1.00,1.00)	-0.000225	1
(-0.894640,-1.000000,1.00,1.00,1.00,1.00)	-0.000103	1
(-0.897369,-1.000000,1.00,1.00,1.000,1.00)	-0.000648	1
(1.000000,1.000000,1.00,1.00,1.00,1.00)	0.239638e-07	35

Numerical results

- sip_S6 a reported global maximizer and two local with objective function values of 0.027092, -3.69008 and -1.95425 respectively.

t	$\bar{g}(t)$
...	
$(1.622134, 1.687810, 2.000000, 0.085439, 2.000000, 0.350174)$	0.024811
...	
$(1.634326, 1.671065, 2.000000, 0.054348, 2.000000, 2.000000)$	-1.954538
...	

- sip_U reported two global maximizers and eleven local maximizers

t	$\bar{g}(t)$	$npar$
(-0.665555,-1.000000,1.00,1.00,1.00,1.00)	-0.002587	1
(-0.689138,-0.933410,1.00,1.00,1.00,1.00)	-0.003319	1
(-0.890160,-1.000000,1.00,1.00,1.00,1.00)	-0.000225	1
(-0.894640,-1.000000,1.00,1.00,1.00,1.00)	-0.000103	1
(-0.897369,-1.000000,1.00,1.00,1.000,1.00)	-0.000648	1
(1.000000,1.000000,1.00,1.00,1.00,1.00)	0.239638e-07	35

Ongoing work

- A quasi-Newton approach is incorporated with a particle swarm strategy in order to reduce the number of function evaluations
- A line-search is being used in order to guarantee the converge to at least a local solutions with high accuracy
- A MATLAB version is already implemented
- To implement a reduction type method for SIP using the developed strategy for multi-local optimization

Ongoing work

- A quasi-Newton approach is incorporated with a particle swarm strategy in order to reduce the number of function evaluations
- A line-search is being used in order to guarantee the converge to at least a local solutions with high accuracy
- A MATLAB version is already implemented
- To implement a reduction type method for SIP using the developed strategy for multi-local optimization

Ongoing work

- A quasi-Newton approach is incorporated with a particle swarm strategy in order to reduce the number of function evaluations
- A line-search is being used in order to guarantee the converge to at least a local solutions with high accuracy
- A MATLAB version is already implemented
- To implement a reduction type method for SIP using the developed strategy for multi-local optimization

Ongoing work

- A quasi-Newton approach is incorporated with a particle swarm strategy in order to reduce the number of function evaluations
- A line-search is being used in order to guarantee the converge to at least a local solutions with high accuracy
- A MATLAB version is already implemented
- To implement a reduction type method for SIP using the developed strategy for multi-local optimization

THE END

email: aivaz@dps.uminho.pt
Web http://www.norg.uminho.pt/aivaz