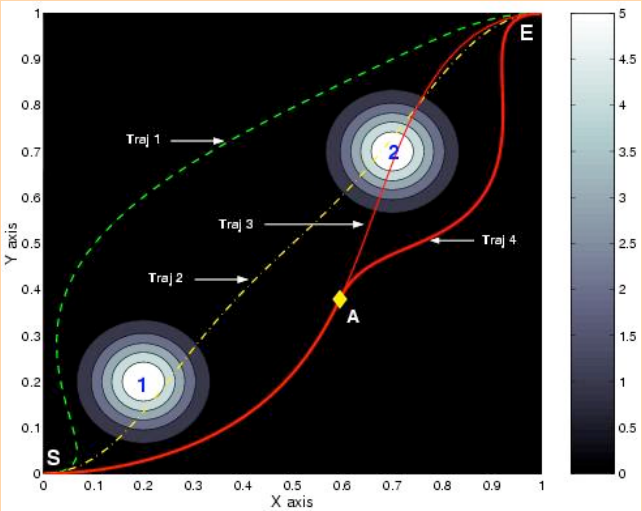
Nonlinear Trajectory Generation Workshop



Control and Dynamical Systems

California Institute of Technology

Sep. 15 - 16

Layout for NTG Workshop

Day 1: Basic 1:00 pm, Sept. 15th, 2003

- **Welcome and Introduction** (Tamer Inanc) 1:00 – 1:15
- **Define Optimal Control Problem (OCP)** (Melvin Flores) 1:15 – 1:45
  - Functions involved – Cost and constraints
  - Variables involved – Horizon length, bounds
- **Transcription to Nonlinear Programming** (Melvin Flores) 1:45 – 2:45
  - Control Space Formulation using B-Splines
  - Smoothness, break points, etc
  - Tricks and wisdom in choosing order, smoothness, etc
- **BREAK** 2:45 – 3:00
- **OCP with spatial constraints – Non-flat systems** (Sérgio Fraga) 3:00 – 3:30
  - Mathematical formulation
  - Use NTGML to define OCP
  - Code generate, run simulations, see plots
- **OCP with spatial constraints – Flat systems** (Sérgio Fraga) 3:30 – 4:00
  - Mathematical formulation
  - Use NTGML to define OCP
  - Code generate, run simulations, see plots
- **OCP with open final time** (Raktim Bhattacharya) 4:00 – 5:00
  - Mathematical formulation
  - Use NTGML to define OCP
  - Code generate, run simulations, see plots
- **Limits of NTGML** (Raktim Bhattacharya) 5:00 – 5:15
- **Installation of NTG / NTGML** 5:15 – ??

Layout for NTG Workshop

Day 2: Advanced 8:30 am, Sept. 16th, 2003

- **Walk through vanderpol.c example** (Tamer Inanc) 08:30 – 10:00
  - Explain nlicf( ),nltcf( ), etc
  - NTGML generates these
  - Arguments of ntg(…)
  - Tuning NPSOL
  - Warm start, initial guess, breaks, knots
  - Output: Coefficients or Trajectories
  - Using MATLAB to construct trajectories from coefficients
- **BREAK** 10:00 – 10:15
- **B-Splines revisited** (Tamer Inanc) 10:15 – 11:30
  - Table lookup, why convert tables to Splines?
  - Properties of B-Splines
  - Local support
  - Multiplicity, Order, Knots, Smoothness
  - Examples of MATLAB manipulating Splines.
- **NTGML** (Raktim Bhattacharya) 11:30 – 12:00
  - Explain the full specification of NTGML
- **Discussion** 12:00 – 12:30
- **Tech Support** 12:30 – ??
  - Help in downloading, installation, etc

Introduction to NTG

Melvin E. Flores
melvin@caltech.edu

Control and Dynamical Systems
California Institute of Technology

The Technology of NTG

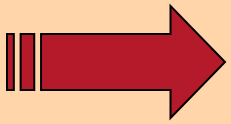
Goal: To Generate Optimal Trajectories for Constrained Dynamical Systems in Real Time.

OC Problem

$$\min_{\mathbf{x},\mathbf{u}} \; g(t_0, \mathbf{x}(t_0), t_f, \mathbf{x}(t_f)) + \int_{t_0}^{t_f} f(s, \mathbf{x}(s), \mathbf{u}(s)) \; ds$$

$$\dot{\mathbf{x}} = \mathbf{F}(t, \mathbf{x}, \mathbf{u})$$

$$\begin{aligned} (t_0, \mathbf{x}(t_0), t_f, \mathbf{x}(t_f)) &\in \mathcal{B} \\ (t, \mathbf{x}(t)) &\in \mathcal{S} \\ \mathbf{u}(t) &\in \mathcal{U}(t, \mathbf{x}(t)) \end{aligned}$$



NLP Problem

$$\min_{\mathbf{c}} f(\mathbf{c})$$

$$\begin{aligned} C_E(\mathbf{c}) &= 0 \\ C_I(\mathbf{c}) &\leq 0 \end{aligned}$$

- B-Spline Parameterization of unknown functions
- Reduction or Elimination of Dynamic Constraints
- Discretization of time (Collocation)

The OC Problem:

Objective Function

$$\min_{\mathbf{x},\mathbf{u}} \; \mathcal{F}_i(t_0, \mathbf{x}(t_0), \mathbf{u}(t_0)) + \int_{t_0}^{t_f} \mathcal{F}_t(s, \mathbf{x}(s), \mathbf{u}(s))\, ds + \mathcal{F}_f(t_f, \mathbf{x}(t_f), \mathbf{u}(t_f))$$

Subject to:

Dynamics

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t, \mathbf{x}(t), \mathbf{u}(t))$$

Constraints

$$\begin{aligned} L_i \leq \; & C_i(t_0, \mathbf{x}(t_0), \mathbf{u}(t_0)) \; \leq U_i \\ L_t \leq \; & C_t(t, \mathbf{x}(t), \mathbf{u}(t)) \; \leq U_t \\ L_f \leq \; & C_f(t_f, \mathbf{x}(t_f), \mathbf{u}(t_f)) \; \leq U_f \end{aligned}$$

where $\mathbf{x} : [t_0, t_f] \to \mathbb{R}^n$ and $\mathbf{u} : [t_0, t_f] \to \mathbb{R}^m$.

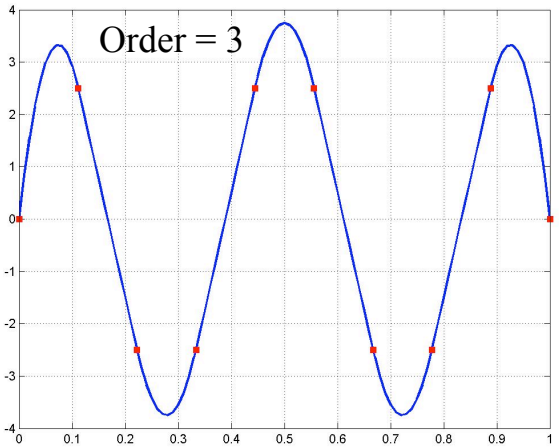
Optimal solution class:

Piecewise Polynomials Functions

Piecewise Polynomial Functions

Piecewise constant

Piecewise linear



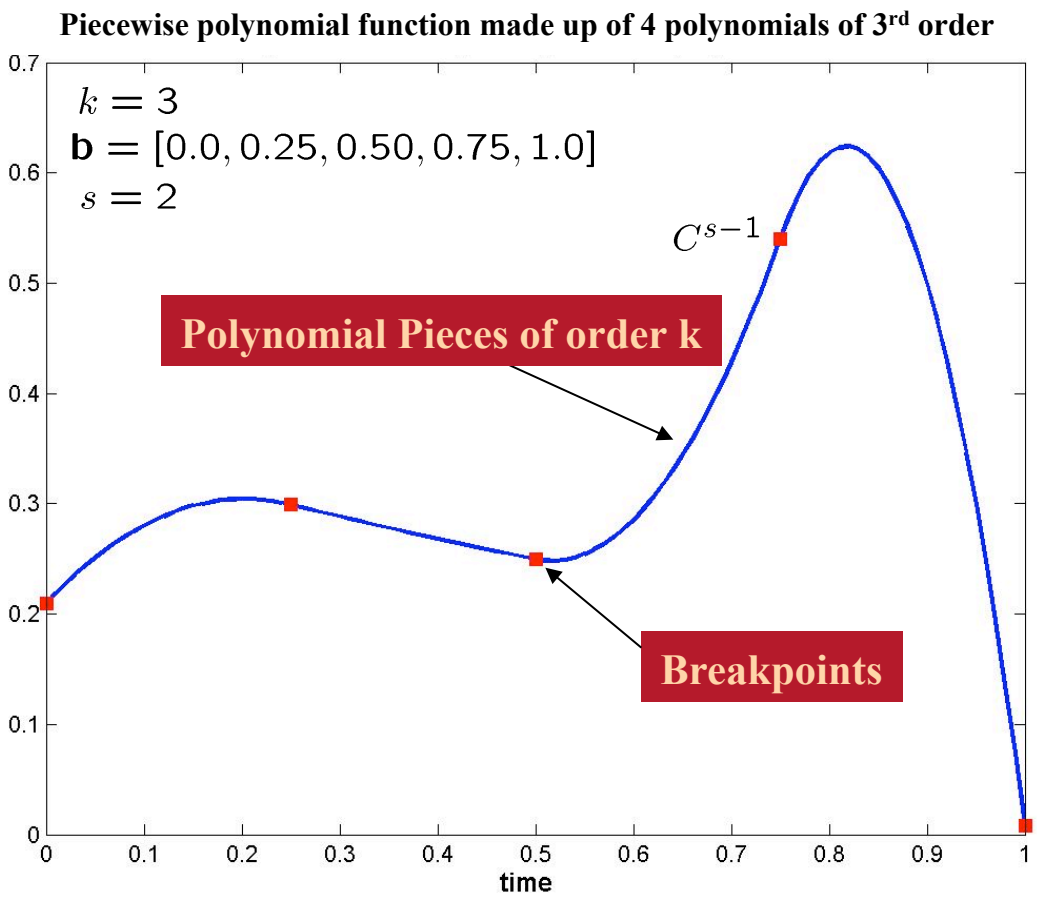
Piecewise quadratic

Piecewise quadratic

Piecewise cubic

Higher orders

Construction of PP Functions



Requirements:

$k \equiv$ Order

$\mathbf{b} \equiv$ Breakpoints

$\mathbf{b} = [b_0, b_1, \ldots, b_{N_b-1}]$

$b_0 < b_1 < \ldots < b_{N_b-1}$

$N_b \equiv$ # of Breakpoints

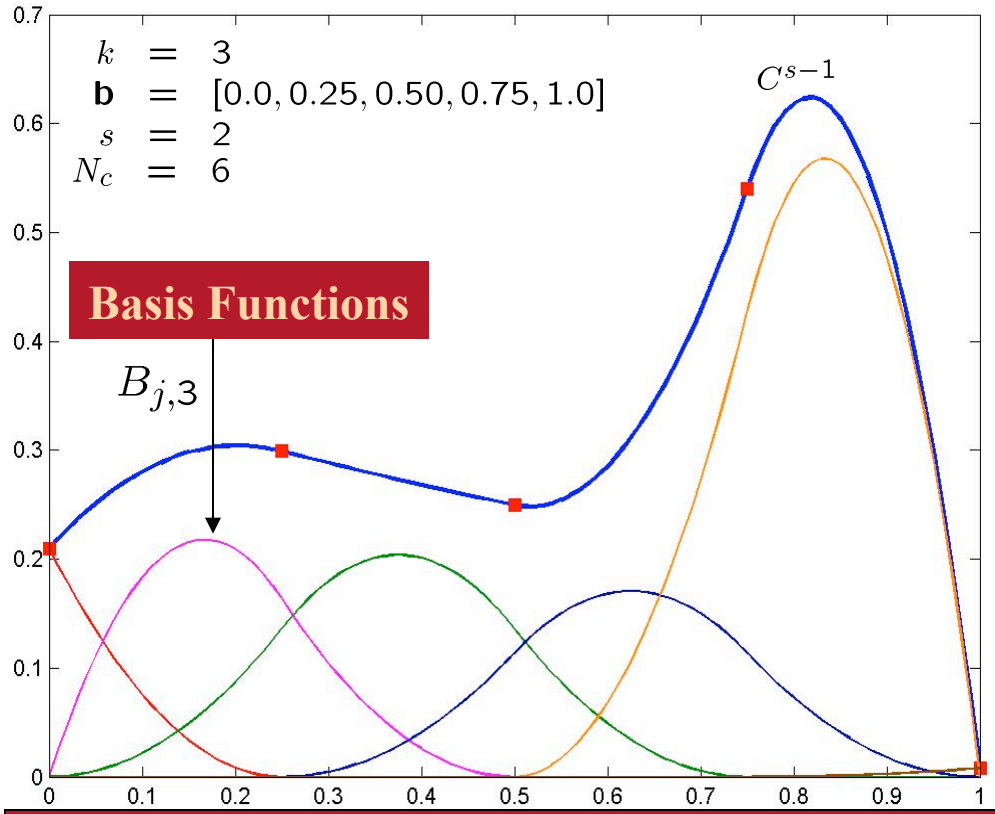
$s \equiv$ Smoothness

$$x(t) \in \mathcal{PP}_{k,\mathbf{b},s}$$

PP functions can efficiently be written as a linear combination of normalized B-Splines.

B-Spline Parameterization

Piecewise polynomial function made up of 4 polynomials of 3rd order



$$x(t) \in \mathcal{PP}_{k,\mathbf{b},s}$$

$$x(t) = \sum_{j=1}^{N_c} B_{j,k}(t)\ c_j$$

$$N_c = (N_b - 1)(k - s) + s$$

$$N_c = (5 - 1)(3 - 2) + 2 = 6$$

$$x(t) = B_{1,3}(t)\ c_1 + \cdots + B_{6,3}(t)\ c_6$$

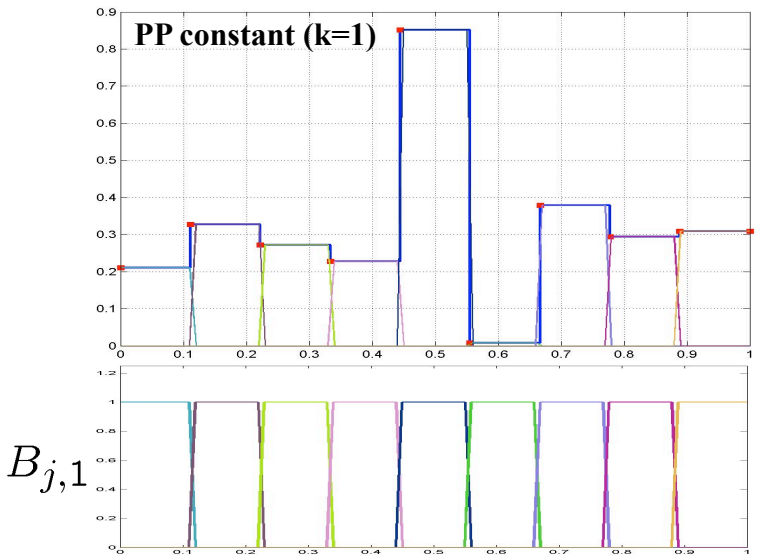
$B_{j,k} \equiv$ Basis Functions

$c_j \equiv$ Coefficients

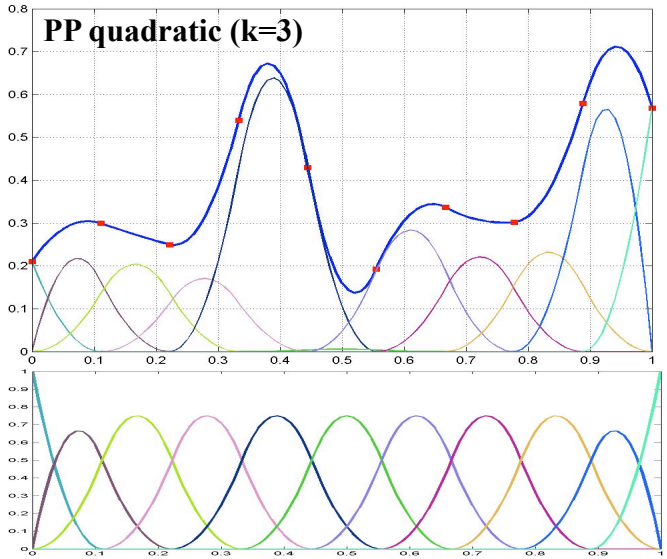
$N_c \equiv$ # of Coeffs

- **Finite number of basis functions required**
- **Fast and stable numerical computation of the basis functions and their derivatives**
- **Ease of enforcing continuity at breakpoints**
- **Basis functions have local support**
- **At most k basis functions are active per interval**

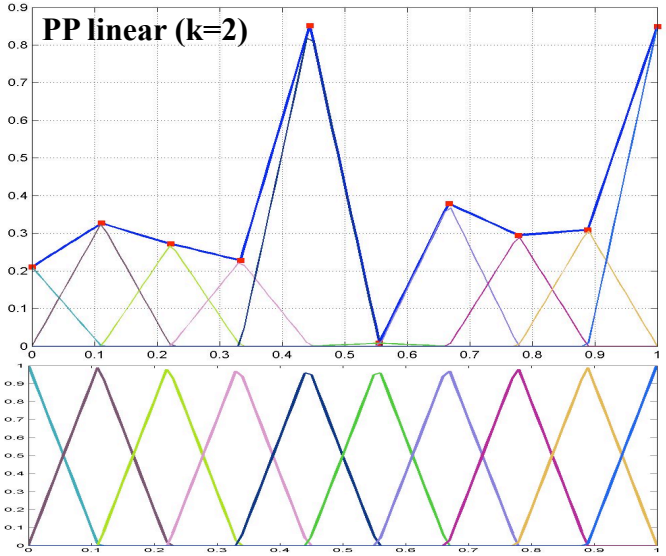
Basis functions and PP functions



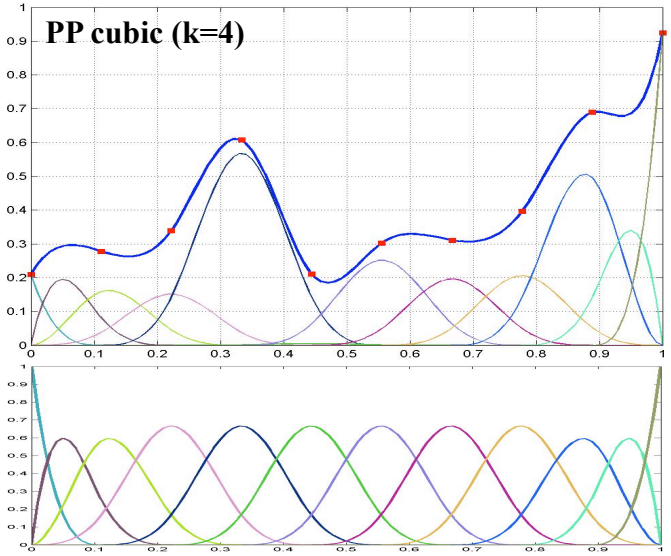
$B_{j,1}$



$B_{j,3}$



$B_{j,2}$



$B_{j,4}$

Goal: To Generate Optimal Trajectories for Constrained Dynamical Systems in Real Time.

OC Problem

$$\min_{\mathbf{x},\mathbf{u}} \; \mathcal{F}_i(t_0,\mathbf{x}(t_0),\mathbf{u}(t_0)) + \int_{t_0}^{t_f} \mathcal{F}_t(s,\mathbf{x}(s),\mathbf{u}(s))\, ds + \mathcal{F}_f(t_f,\mathbf{x}(t_f),\mathbf{u}(t_f))$$

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t,\mathbf{x}(t),\mathbf{u}(t))$$

$$\begin{aligned} L_i \le \; & C_i(t_0,\mathbf{x}(t_0),\mathbf{u}(t_0)) \; \le U_i \\ L_t \le \; & C_t(t,\mathbf{x}(t),\mathbf{u}(t)) \; \le U_t \\ L_f \le \; & C_f(t_f,\mathbf{x}(t_f),\mathbf{u}(t_f)) \; \le U_f \end{aligned}$$

where $\mathbf{x} : [t_0,t_f] \to \mathbb{R}^n$ and $\mathbf{u} : [t_0,t_f] \to \mathbb{R}^m$.

$$\mathbf{x},\mathbf{u} \in \mathcal{PP}_{k,\mathbf{b},s}$$

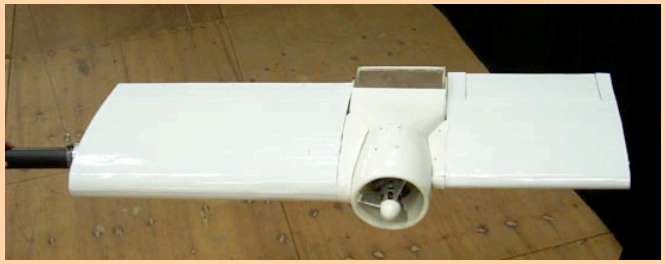
NLP Problem

$$\min_{\mathbf{c}} f(\mathbf{c})$$

$$\begin{aligned} C_E(\mathbf{c}) &= 0 \\ C_I(\mathbf{c}) &\le 0 \end{aligned}$$

- B-Spline Parameterization of unknown functions
- Reduction or Elimination of Dynamic Constraints
- Discretization of time (Collocation)

Planar Ducted Fan



$$\min_{\mathbf{x},\mathbf{u}} \ \frac{1}{2}\hat{\mathbf{x}}(t_f)^T \mathbf{P}\ \hat{\mathbf{x}}(t_f) + \frac{1}{2}\int_{t_0}^{t_f} \left\{\hat{\mathbf{x}}(s)^T \mathbf{Q}\ \hat{\mathbf{x}}(s) + \hat{\mathbf{u}}(s)^T \mathbf{R}\ \hat{\mathbf{u}}(s)\right\} ds$$

subject to:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{1}{m}\ cos(x_5)\ u_1 + \frac{1}{m}\ sin(x_5)\ u_2 \\ x_4 \\ g - \frac{1}{m}\ sin(x_5)\ u_1 + \frac{1}{m}\ cos(x_5)\ u_2 \\ x_6 \\ \frac{R_o}{J}\ u_2 \end{bmatrix}; \quad \mathbf{x}(t_0) = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ \frac{\pi}{2} \\ 0 \end{bmatrix}$$

$$\begin{aligned} 0 \leq \ \ u_1 \ \ &\leq u_1^{Max} \\ -\frac{1}{2}u_1^{Max} \leq \ \ u_2 \ \ &\leq \frac{1}{2}u_1^{Max} \end{aligned}$$

$\mathbf{x} \in \mathbb{R}^6$ and $\mathbf{u} \in \mathbb{R}^2$

$\mathbf{x} = (x, \dot{x}, y, \dot{y}, \theta, \dot{\theta})$; $\mathbf{u} = (F_{xb}, F_{yb})$

$\mathbf{P} \geq 0$, $\mathbf{Q} \geq 0$ and $\mathbf{R} > 0$

where $\hat{\mathbf{x}}(t) = (\mathbf{x}(t) - \mathbf{x}_e)$ and $\hat{\mathbf{u}}(t) = (\mathbf{u}(t) - \mathbf{u}_e)$
with $\mathbf{x}_e = (0, 0, 0, 0, \frac{\pi}{2}, 0)$ and $\mathbf{u}_e = (mg, 0)$.

Reduction or Elimination of Dynamic Constraints

- **Full Dynamics:**
  - –B-Spline parameterization of both the states and inputs.
- **Inverse Dynamic Optimization:**
  - –B-Spline parameterization of the states
  - –Write the inputs in terms of the states and their derivatives
- **Differentially Flat Outputs:**
  - –Determine a set of outputs such as that both the states and inputs can be written in terms of these outputs and their derivatives
  - –B-Spline parameterization of these outputs

Full Dynamics

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{1}{m}\, cos(x_5)\; u_1 + \frac{1}{m}\, sin(x_5)\; u_2 \\ x_4 \\ g - \frac{1}{m}\, sin(x_5)\; u_1 + \frac{1}{m}\, cos(x_5)\; u_2 \\ x_6 \\ \frac{R_o}{J}\; u_2 \end{bmatrix}$$

$$\begin{bmatrix} z_1 \\ \dot{z}_1 \\ \ddot{z}_1 \\ z_2 \\ \dot{z}_2 \\ \ddot{z}_2 \\ z_3 \\ \dot{z}_3 \\ \ddot{z}_3 \\ z_4 \\ z_5 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \\ y \\ \dot{y} \\ \ddot{y} \\ \theta \\ \dot{\theta} \\ \ddot{\theta} \\ u_1 \\ u_2 \end{bmatrix}$$

B-Spline Parameterization

$$z_i^{(r)}(t) = \sum_{j=1}^{N_i^c} B_{j,k_i}^{(r)}(t)\; c_j^i$$

Variables = 11
Eq. Constraints = 3

$$\begin{aligned} \ddot{x} &= \frac{1}{m}\, cos(\theta)\; u_1 + \frac{1}{m}\, sin(\theta)\; u_2 \\ \ddot{y} &= g - \frac{1}{m}\, sin(\theta)\; u_1 + \frac{1}{m}\, cos(\theta)\; u_2 \\ \ddot{\theta} &= \frac{R_o}{J}\; u_2 \end{aligned}$$

$$\begin{aligned} \ddot{z}_1 - \frac{1}{m}\, cos(z_3)\; z_4 - \frac{1}{m}\, sin(z_3)\; z_5 &= 0 \\ \ddot{z}_2 - g + \frac{1}{m}\, sin(z_3)\; z_4 - \frac{1}{m}\, cos(z_3)\; z_5 &= 0 \\ \ddot{z}_3 - \frac{R_o}{J}\; z_5 &= 0 \end{aligned}$$

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \\ u_1 \\ u_2 \end{bmatrix}$$

Choose the outputs to be the states and inputs

Inverse Dynamic Optimization

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{1}{m}\ cos(x_5)\ u_1 + \frac{1}{m}\ sin(x_5)\ u_2 \\ x_4 \\ g - \frac{1}{m}\ sin(x_5)\ u_1 + \frac{1}{m}\ cos(x_5)\ u_2 \\ x_6 \\ \frac{R_o}{J}\ u_2 \end{bmatrix}$$

$$\begin{bmatrix} z_1 \\ \dot{z}_1 \\ \ddot{z}_1 \\ z_2 \\ \dot{z}_2 \\ \ddot{z}_2 \\ z_3 \\ \dot{z}_3 \\ \ddot{z}_3 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \\ y \\ \dot{y} \\ \ddot{y} \\ \theta \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix}$$

B-Spline Parameterization

$$z_i^{(r)}(t) = \sum_{j=1}^{N_i^c} B_{j,k_i}^{(r)}(t)\ c_j^i$$

Variables = 9
Eq. Constraints = 1

$$\begin{aligned} \ddot{x} &= \frac{1}{m}\ cos(\theta)\ u_1 + \frac{1}{m}\ sin(\theta)\ u_2 \\ \ddot{y} &= g - \frac{1}{m}\ sin(\theta)\ u_1 + \frac{1}{m}\ cos(\theta)\ u_2 \\ \ddot{\theta} &= \frac{R_o}{J}\ u_2 \end{aligned}$$

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

$$\begin{aligned} u_1 &= m\ \ddot{z}_1\ cos(z_3) - (m\ \ddot{z}_2 - m\ g)\ sin(z_3) \\ u_2 &= m\ \ddot{z}_1\ sin(z_3) - (m\ \ddot{z}_2 - m\ g)\ cos(z_3) \end{aligned}$$

$$\ddot{z}_3 - \frac{R_o}{J}\ z_5 = 0$$

Choose the outputs to be the states

Write the inputs in terms of the outputs and their derivatives

Differentially Flat Outputs

Flat Outputs:

$$(z_1, \ldots, z_m) = h(\mathbf{x}, \mathbf{u}^{(0)}, \mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(r)})$$

$$(z_1^{(0)}, \ldots, z_1^{(d_1)}, \ldots, z_m^{(0)}, \ldots, z_m^{(d_m)}) = \Phi(\mathbf{x}, \mathbf{u}^{(0)}, \mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(r)})$$

$$\mathbf{x} = \omega_1(\tilde{\mathbf{z}}) \qquad \mathbf{u} = \omega_2(\tilde{\mathbf{z}}) \qquad \tilde{\mathbf{z}} = z_1^{(0)}, \ldots, z_1^{(d_1)}, \ldots, z_m^{(0)}, \ldots, z_m^{(d_m)})$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{1}{m}\, cos(x_5)\; u_1 + \frac{1}{m}\, sin(x_5)\; u_2 \\ x_4 \\ g - \frac{1}{m}\, sin(x_5)\; u_1 + \frac{1}{m}\, cos(x_5)\; u_2 \\ x_6 \\ \frac{R_o}{J}\; u_2 \end{bmatrix}$$

$$\begin{bmatrix} z_1 \\ \dot{z}_1 \\ \ddot{z}_1 \\ \dddot{z}_1 \\ z_2 \\ \dot{z}_2 \\ \ddot{z}_2 \\ \dddot{z}_2 \end{bmatrix}$$

B-Spline Parameterization

$$z_i^{(r)}(t) = \sum_{j=1}^{N_i^c} B_{j,k_i}^{(r)}(t)\; c_j^i$$

Variables = 8

Eq. Constraints = 0

$$\ddot{x} = \frac{1}{m}\, cos(\theta)\; u_1 + \frac{1}{m}\, sin(\theta)\; u_2$$

$$\ddot{y} = g - \frac{1}{m}\, sin(\theta)\; u_1 + \frac{1}{m}\, cos(\theta)\; u_2$$

$$\ddot{\theta} = \frac{R_o}{J}\, u_2$$

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} x + \frac{J}{R_o\; m}\, cos\; (\theta) \\ y - \frac{J}{R_o\; m}\, sin\; (\theta) \end{bmatrix}$$

Choose the flat outputs

The Modified OC Problem

$$\min_{\mathbf{x},\mathbf{u}} \; \mathcal{F}_i(t_0,\mathbf{x}(t_0),\mathbf{u}(t_0)) + \int_{t_0}^{t_f} \mathcal{F}_t(s,\mathbf{x}(s),\mathbf{u}(s))\, ds + \mathcal{F}_f(t_f,\mathbf{x}(t_f),\mathbf{u}(t_f))$$

$$\dot{\mathbf{x}}(t) = \mathsf{F}(t,\mathbf{x}(t),\mathbf{u}(t))$$

$$\begin{aligned} L_i \le \; & C_i(t_0,\mathbf{x}(t_0),\mathbf{u}(t_0)) \; \le U_i \\ L_t \le \; & C_t(t,\mathbf{x}(t),\mathbf{u}(t)) \; \le U_t \\ L_f \le \; & C_f(t_f,\mathbf{x}(t_f),\mathbf{u}(t_f)) \; \le U_f \end{aligned}$$

where $\mathbf{x} : [t_0,t_f] \to \mathbb{R}^n$ and $\mathbf{u} : [t_0,t_f] \to \mathbb{R}^m$.

- **Full Dynamics**
- **Inverse Dynamic Optimization**
- **Differentially Flat Outputs**

$$\tilde{\mathbf{z}} = \begin{bmatrix} z_1^{(0)} \\ \vdots \\ z_1^{(d_1)} \\ \vdots \\ z_{N_o}^{(0)} \\ \vdots \\ z_{N_o}^{(d_{N_o})} \end{bmatrix}$$

$$z_i^{(r)}(t) = \sum_{j=1}^{N_i^c} B_{j,k_i}^{(r)}(t)\; c_j^i$$

$$\min_{\tilde{\mathbf{z}}} \; \mathcal{G}_i(t_0,\tilde{\mathbf{z}}(t_0)) + \int_{t_0}^{t_f} \mathcal{G}_t(s,\tilde{\mathbf{z}}(s))\, ds + \mathcal{G}_f(t_f,\tilde{\mathbf{z}}(t_f))$$

$$\Gamma(t,\tilde{\mathbf{z}}(t)) = 0$$

$$\begin{aligned} l_i \le \; & \mathcal{A}_i\; \tilde{\mathbf{z}}(t_0) \; \le u_i \\ l_t \le \; & \mathcal{A}_t\; \tilde{\mathbf{z}}(t) \; \le u_t \\ l_f \le \; & \mathcal{A}_f\; \tilde{\mathbf{z}}(t_f) \; \le u_f \\ L_i \le \; & \mathcal{C}_i(t_0,\tilde{\mathbf{z}}(t_0)) \; \le U_i \\ L_t \le \; & \mathcal{C}_t(t,\tilde{\mathbf{z}}(t)) \; \le U_t \\ L_f \le \; & \mathcal{C}_f(t_f,\tilde{\mathbf{z}}(t_f)) \; \le U_f \end{aligned}$$

where $\tilde{\mathbf{z}} : [t_0,t_f] \to \mathbb{R}^{N_v}$.

Goal: To Generate Optimal Trajectories for Constrained Dynamical Systems in Real Time.

OC Problem

$$\min_{\tilde{\mathbf{z}}} \; \mathcal{G}_i(t_0, \tilde{\mathbf{z}}(t_0)) + \int_{t_0}^{t_f} \mathcal{G}_t(s, \tilde{\mathbf{z}}(s))\, ds + \mathcal{G}_f(t_f, \tilde{\mathbf{z}}(t_f))$$

$$\begin{aligned}
l_i \leq \quad & \mathcal{A}_i\ \tilde{\mathbf{z}}(t_0) && \leq u_i \\
l_t \leq \quad & \mathcal{A}_t\ \tilde{\mathbf{z}}(t) && \leq u_t \\
l_f \leq \quad & \mathcal{A}_f\ \tilde{\mathbf{z}}(t_f) && \leq u_f \\
L_i \leq \quad & \mathcal{C}_i(t_0, \tilde{\mathbf{z}}(t_0)) && \leq U_i \\
L_t \leq \quad & \mathcal{C}_t(t, \tilde{\mathbf{z}}(t)) && \leq U_t \\
L_f \leq \quad & \mathcal{C}_f(t_f, \tilde{\mathbf{z}}(t_f)) && \leq U_f
\end{aligned}$$

where $\tilde{\mathbf{z}} : [t_0, t_f] \rightarrow \mathbb{R}^{N_v}$.

$$z_i^{(r)}(t) = \sum_{j=1}^{N_i^c} B_{j,k_i}^{(r)}(t)\ c_j^i$$

NLP Problem

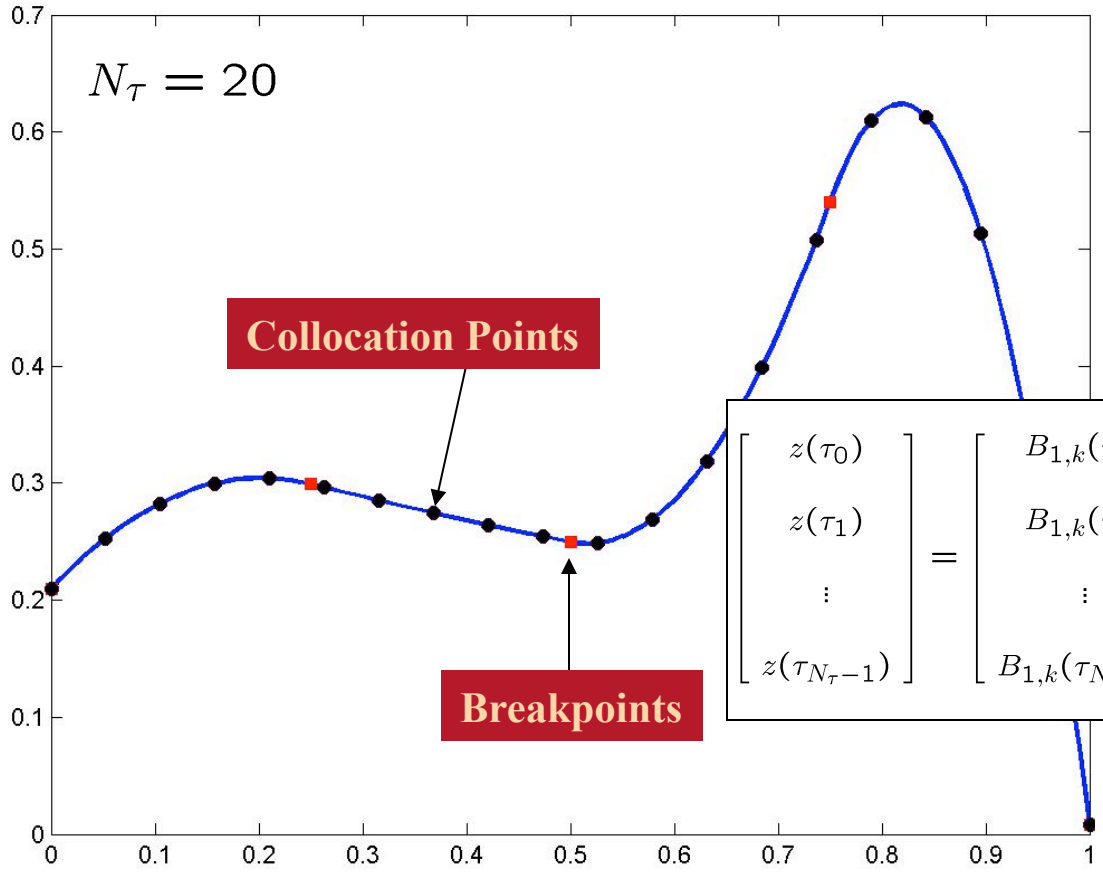
NPSOL

$$\min_{\tilde{\mathbf{C}}} f(\tilde{\mathbf{C}})$$

$$\mathbf{L} \leq \begin{bmatrix} \tilde{\mathbf{C}} \\ \mathbf{A}\tilde{\mathbf{C}} \\ \mathbf{c}(\tilde{\mathbf{C}}) \end{bmatrix} \leq \mathbf{U}$$

- B-Spline Parameterization of unknown functions
- Reduction or Elimination of Dynamic Constraints
- Discretization of time (Collocation)

Collocation



$$\tau = [\tau_0, \ldots, \tau_{N_\tau - 1}]$$
$$\tau_0 < \tau_0 < \cdots < \tau_{N_\tau - 1}$$
$$N_\tau \equiv \#\ \text{of Colloc Points}$$
$$z(t) = \sum_{i=1}^{N_c} B_{i,k}(t)\ c_i$$

$$\begin{bmatrix} z(\tau_0) \\ z(\tau_1) \\ \vdots \\ z(\tau_{N_\tau - 1}) \end{bmatrix} = \begin{bmatrix} B_{1,k}(\tau_0) & B_{2,k}(\tau_0) & \ldots & B_{N_c,k}(\tau_0) \\ B_{1,k}(\tau_1) & B_{2,k}(\tau_1) & \ldots & B_{N_c,k}(t) \\ \vdots & & & \\ B_{1,k}(\tau_{N_\tau - 1}) & B_{2,k}(\tau_{N_\tau - 1}) & \ldots & B_{N_c,k}(\tau_{N_\tau - 1}) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{N_c} \end{bmatrix}$$

The collocation points are the sites at which the constraints will be enforced and the Integral evaluated

Collocation

$$\tilde{\mathbf{z}} = \begin{bmatrix} z_1^{(0)} \\ \vdots \\ z_1^{(d_1)} \\ \vdots \\ z_{N_o}^{(0)} \\ \vdots \\ z_{N_o}^{(d_{N_o})} \end{bmatrix}$$

$$\tilde{\mathbf{z}}(\tau_{N_\tau - 1}) = \begin{bmatrix} \tilde{\mathbf{z}}_1(\tau_{N_\tau-1}) \\ \tilde{\mathbf{z}}_2(\tau_{N_\tau-1}) \\ \vdots \\ \tilde{\mathbf{z}}_{N_o}(\tau_{N_\tau-1}) \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{B}}_1(\tau_{N_\tau-1}) & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{B}}_2(\tau_{N_\tau-1}) & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \tilde{\mathbf{B}}_{N_o}(\tau_{N_\tau-1}) \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{C}}_1 \\ \tilde{\mathbf{C}}_2 \\ \vdots \\ \tilde{\mathbf{C}}_{N^o} \end{bmatrix}$$

$$\tilde{\mathbf{z}}(\tau_0) = \begin{bmatrix} \tilde{\mathbf{z}}_1(\tau_0) \\ \tilde{\mathbf{z}}_2(\tau_0) \\ \vdots \\ \tilde{\mathbf{z}}_{N_o}(\tau_0) \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{B}}_1(\tau_0) & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{B}}_2(\tau_0) & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \tilde{\mathbf{B}}_{N_o}(\tau_0) \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{C}}_1 \\ \tilde{\mathbf{C}}_2 \\ \vdots \\ \tilde{\mathbf{C}}_{N^o} \end{bmatrix}$$

The NLP Problem

$$\min_{\tilde{\mathbf{z}}} \; \mathcal{G}_i(t_0, \tilde{\mathbf{z}}(t_0)) + \int_{t_0}^{t_f} \mathcal{G}_t(s, \tilde{\mathbf{z}}(s))\, ds + \mathcal{G}_f(t_f, \tilde{\mathbf{z}}(t_f))$$

$$\begin{aligned}
l_i \leq \quad & \mathcal{A}_i\ \tilde{\mathbf{z}}(t_0) && \leq u_i \\
l_t \leq \quad & \mathcal{A}_t\ \tilde{\mathbf{z}}(t) && \leq u_t \\
l_f \leq \quad & \mathcal{A}_f\ \tilde{\mathbf{z}}(t_f) && \leq u_f \\
L_i \leq \quad & \mathcal{C}_i(t_0, \tilde{\mathbf{z}}(t_0)) && \leq U_i \\
L_t \leq \quad & \mathcal{C}_t(t, \tilde{\mathbf{z}}(t)) && \leq U_t \\
L_f \leq \quad & \mathcal{C}_f(t_f, \tilde{\mathbf{z}}(t_f)) && \leq U_f
\end{aligned}$$

where $\tilde{\mathbf{z}} : [t_0, t_f] \to \mathbb{R}^{N_v}$.

$$\tilde{\mathbf{z}}(t) = \tilde{\mathbf{B}}(t)\ \tilde{\mathbf{C}}$$
$$t \in \tau = [\tau_0, \ldots, \tau_{N_\tau - 1}]$$

$$z \in \mathcal{PP}_{k,\mathbf{b},s}$$

$$\min_{\tilde{\mathbf{C}}} f(\tilde{\mathbf{C}})$$

$$\mathbf{L} \leq \begin{bmatrix} \tilde{\mathbf{C}} \\ \mathbf{A}\tilde{\mathbf{C}} \\ \mathbf{c}(\tilde{\mathbf{C}}) \end{bmatrix} \leq \mathbf{U}$$

$$f(\tilde{\mathbf{C}}) = \mathcal{G}_i(\tau_0, \tilde{\mathbf{z}}(\tau_0)) + \sum_{i=1}^{N_\tau} \mathcal{G}_u(\tau_i, \tilde{\mathbf{z}}(\tau_i))\, h_i + \mathcal{G}_f(\tau_{N_\tau - 1}, \tilde{\mathbf{z}}(\tau_{N_\tau - 1}))$$

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_c \\ l_i \\ (l_t)^1 \\ \vdots \\ (l_t)^{N_\tau} \\ l_f \\ L_i \\ (L_t)^1 \\ \vdots \\ (L_t)^{N_\tau} \\ L_f \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} \mathcal{A}_i\ \tilde{\mathbf{B}}(\tau_0^c) \\ \mathcal{A}_t\ \tilde{\mathbf{B}}(\tau_0) \\ \vdots \\ \mathcal{A}_t\ \tilde{\mathbf{B}}(\tau_{N_\tau - 1}) \\ \mathcal{A}_f\ \tilde{\mathbf{B}}(\tau_{N_\tau - 1}) \end{bmatrix} \quad \mathbf{U} = \begin{bmatrix} \mathbf{U}_c \\ u_i \\ (u_t)^1 \\ \vdots \\ (u_t)^{N_\tau} \\ u_f \\ U_i \\ (U_t)^1 \\ \vdots \\ (U_t)^{N_\tau} \\ U_f \end{bmatrix}$$

$$\mathbf{c}(\tilde{\mathbf{C}}) = \begin{bmatrix} \mathcal{C}_i(\tilde{\mathbf{z}}(\tau_0)) \\ \mathcal{C}_t(\tilde{\mathbf{z}}(\tau_0)) \\ \vdots \\ \mathcal{C}_t(\tilde{\mathbf{z}}(\tau_{N_\tau - 1})) \\ \mathcal{C}_f(\tilde{\mathbf{z}}(\tau_{N_\tau - 1})) \end{bmatrix}$$

Calling NTG

$$\min_{\tilde{\mathbf{z}}} \; \mathcal{G}_i(t_0, \tilde{\mathbf{z}}(t_0)) + \int_{t_0}^{t_f} \mathcal{G}_t(s, \tilde{\mathbf{z}}(s))\, ds + \mathcal{G}_f(t_f, \tilde{\mathbf{z}}(t_f))$$

$$\begin{aligned}
l_i \le \quad & \mathcal{A}_i\ \tilde{\mathbf{z}}(t_0) && \le u_i \\
l_t \le \quad & \mathcal{A}_t\ \tilde{\mathbf{z}}(t) && \le u_t \\
l_f \le \quad & \mathcal{A}_f\ \tilde{\mathbf{z}}(t_f) && \le u_f \\
L_i \le \quad & \mathcal{C}_i(t_0, \tilde{\mathbf{z}}(t_0)) && \le U_i \\
L_t \le \quad & \mathcal{C}_t(t, \tilde{\mathbf{z}}(t)) && \le U_t \\
L_f \le \quad & \mathcal{C}_f(t_f, \tilde{\mathbf{z}}(t_f)) && \le U_f
\end{aligned}$$

where $\tilde{\mathbf{z}} : [t_0, t_f] \to \mathbb{R}^{N_v}$.

$$\tilde{\mathbf{z}}(t) = \tilde{\mathbf{B}}(t)\ \tilde{\mathbf{C}}$$
$$t \in \tau = [\tau_0, \ldots, \tau_{N_\tau - 1}]$$

$$z \in \mathcal{PP}_{k,\mathbf{b},s}$$

$$\tilde{\mathbf{z}} = \begin{bmatrix} z_1^{(0)} \\ \vdots \\ z_1^{(d_1)} \\ \vdots \\ z_{N_o}^{(0)} \\ \vdots \\ z_{N_o}^{(d_{N_o})} \end{bmatrix}$$

$$z_i^{(r)}(t) = \sum_{j=1}^{N_i^c} B_{j,k_i}^{(r)}(t)\ c_j^i$$

```
ntg(
      NofOutputs,
      CollocPoints,
      NofCollocPoints,
      NofBreaksPoints-1,
      BreakPoints,
      Order,
      Smoothness,
      MaxDerivatives+1,
      Coefficients,
      NLIC, Ai,
      NLTC, At,
      NLFC, Af,
      NNLIC, Ci,
      NNLTC, Ct,
      NNLFC, Cf,
      NCAVi, CAVi,
      NCAVt, CAVt,
      NCAVf, CAVf,
      LowerBounds,
      UpperBounds,
      NICF, Gi,
      NUCF, Gt,
      NFCF, Gf,
      NOAVi, OAVi,
      NOAVt, OAVi,
      NOAVf, OAVi,
      istate,clambda,R,
     &inform,
     &OptimalCost
);
```

References:

- Mark B. Milam. Real-time optimal trajectory generation for constrained dynamical systems (Caltech Thesis 2003)
- C. De Boor. A Practical Guide to Splines. AMS 27, Springer-Verlag, 1978.
- N. Petit, M. B. Milam and R. M. Murray, Inversion based constrained trajectory optimization, . In 5th IFAC symposium on nonlinear control systems, 2001
- M. Milam, K. Mushambi, and R. R. Murray, A new computational approach to real-time trajectory generation for constrained mechanical systems. CDC2000
- M. Fliess, J. Levine, P. Martin, and P. Rouchon. Flatness and defect of nonlinear systems: introductory theory and examples. International Journal of control 61(6):1327-1360, 1995
- P. E. Gill, W. Murray, M. A. Saunders, and M. Wright. User's Guide for NPSOL 5.0: A Fortran package for nonlinear programming. Systems Optimization Lab, Stanford University, CA 94305.
- J. Nocedal and S. J. Wright, Numerical Optimization. Springer-Verlag, 1999.
- Splines Toolbox in MATLAB

Optimal control problem with spatial constraints

Sérgio Loureiro Fraga

slfraga@fe.up.pt / slfraga@cds.caltech.edu

Visitor student from:

Faculty of Engineering of University of Porto – Portugal

Department of Electrical and Computing Engineering

www.fe.up.pt/dcegwww

Control and Dynamical Systems

The California Institute of Technology

September 15, 2003

Outline

- Mathematical formulation
- Problem specification
- Examples

Mathematical formulation

Solving optimal control problems with NTG

Optimal control problem formulation

NTG formulation

$$\min_{x,u} \; J[x,u] = g(t_0, x_0, u_0, t_f, x_{t_f}, u_{t_f}) + \int_{t_0}^{t_f} g_t(s,x,u)ds$$

Such that

$\dot{x} = F(t,x,u)$

$x_0 \in X_0$

$x_{t_f} \in X_f$

$x(t) \in S$

$u(t) \in U$

$$\min_{z(t)} \; J = F_i(t_0, z_0) + \int_{t_0}^{t_f} F_t(s,z)ds + F_f(t_f, z_f)$$

Such that

Linear constraints:

$l_i \le A_i z \le u_i$

$l_f \le A_f z \le u_f$

$l_f \le A_f z \le u_f$

Nonlinear constraints:

$L_i \le C_i(t_0, z_{t_0}) \le U_i$

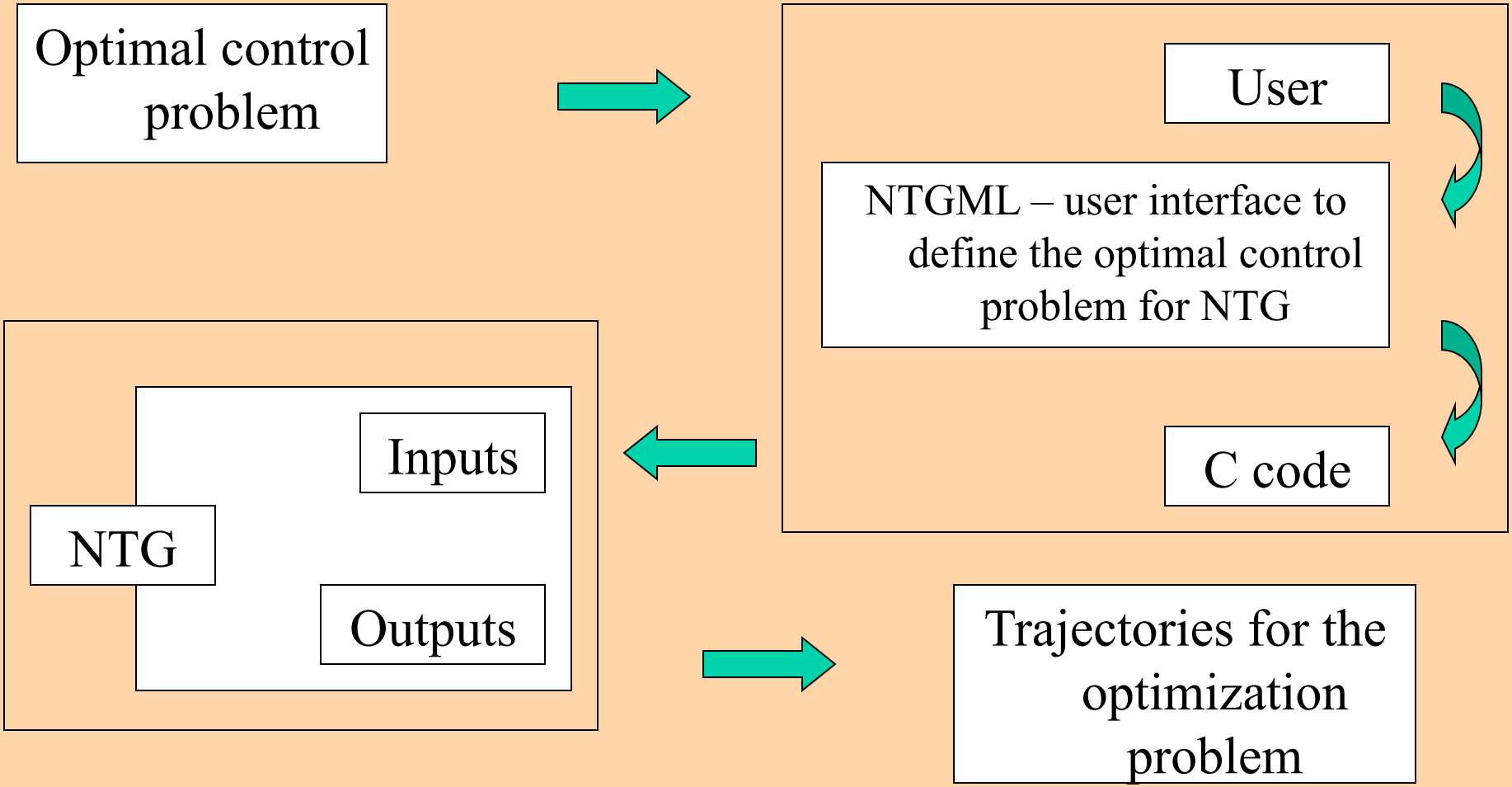
$L_f \le C_t(t,z) \le U_f$

$L_f \le C_f(t_f, z_{t_f}) \le U_f$

Where:

$$z(t) = \begin{bmatrix} z_1 & \dot{z}_1 & \dots & z_n & \dot{z}_n & \dots \end{bmatrix}^T$$

NTG components



NTGML – user interface to define the optimal control problem (OCP)

User interface → Problem definition

Using a markup language: XML

- Structured data
- Grammar for problem definition
- Easy to parse the data for the applications

Features:

- Text file where the data is stored
  - accepted in any OS and any application
  - Simple text editor to create/modify a document
- Structured data with user defined tags
- Data repository for the application
  - Standards enabling creation of valid documents
  - Many parsers developed based in accepted standards
- Definition of the structure and the contents of a document

XML file example

```
<ntgml ...>
  <output> ... </output>
  <lc>     ... </lc>
  <nlc>    ... </nlc>
  <cost>   ... </cost>
  <horizon>...</horizon>
  <misc>   ...</misc>
</ntgml>
```

NTGML - features

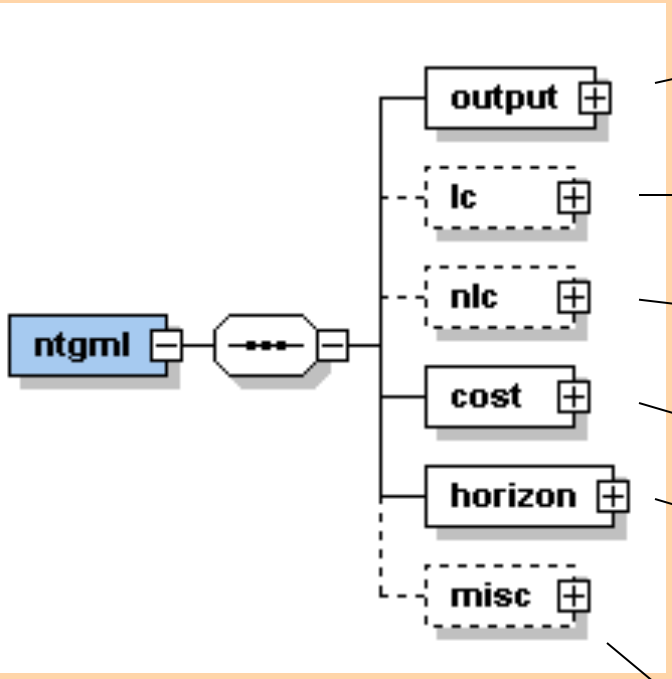
Which features do we use from XML?

Store the problem data in a way that is easy for the NTG applications to access it.	→	Usage of well developed parsers, which implements the standards of XML
Allow the user to specify the OC problem in an easier way	→	Access to a grammar to create a document containing the OCP data
Multi-platform requirements	→	XML is a portable language

NTGML – OCP definition

Grammar to construct the problem specification

Creation of a XML file according to the defined grammar to specify the OC problem.



$$z(t) = [z_1(t) \dots z^{(d_1)}{}_1(t) \dots z_n(t) \dots z^{(d_n)}{}_n(t)]^T$$

$$LB \le Az \le UB$$

$$LB \le c(z) \le UB$$

$$\min_{z(t)} J = F_i(z) + \int_0^T F_t(z)dt + F_f(z)$$

Schema

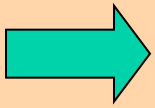
- Constants
- Options
- …

- Definition of the time horizon
- Number of collocation points

Example

Objective: how to use NTGML to solve an OC problem

Optimal control problem



Generate a XML file for this problem

Results and charts

OCP specification – XML file

Van Der Pol Oscillator

$$\min_{u(t),x(t)} \frac{1}{2}\int_0^5 (x_1^2 + x_2^2 + u^2)dt$$

Subject to:

$$x_1(0) = 1$$

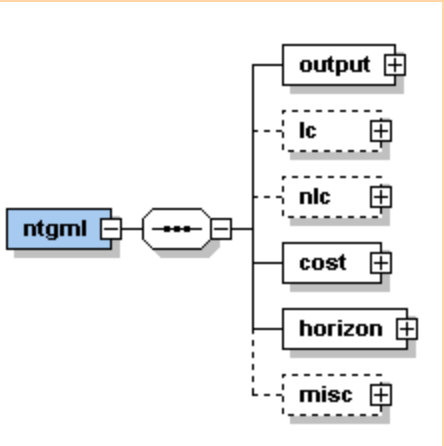
$$x_2(0) = 0$$

$$-x_1(5) + x_2(5) = 1$$

where

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -x_1 + (1 - x_1)^2 x_2 + u$$



Schema rules the way the user should specify the NTGML file

```
<ntgml ...>
  <output> ... </output>
  <lc>     ... </lc>
  <nlc>    ... </nlc>
  <cost>   ... </cost>
  <horizon>...</horizon>
  <misc>   ...</misc>
</ntgml>
```

Problem specification

OCP specification – variables' definition

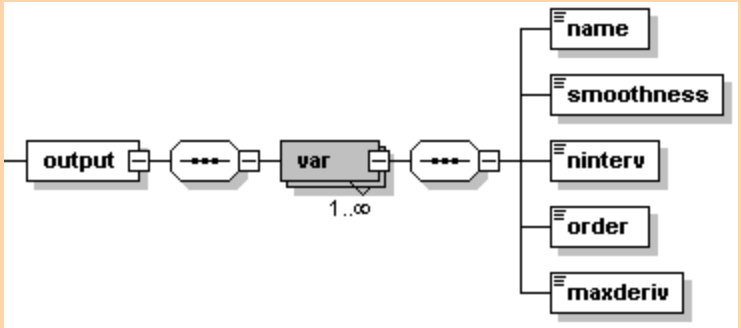
```
<ntgml ...>
  <output> ... </output>
  <lc>     ... </lc>
  <nlc>    ... </nlc>
  <cost>   ... </cost>
  <horizon>...</horizon>
  <misc>   ...</misc>
</ntgml>
```

Output

x_1 – state variable

x_2 – state variable

u – input



$$z(t) = [x_1(t)\ \dot{x}_1(t)\ x_2(t)\ \dot{x}_2(t)\ u(t)]^T$$

```
<output>
  <var>
    <name>       x1</name>
    <smoothness>3</smoothness>
    <ninterv>    5</ninterv>
    <order>      5</order>
    <maxderiv>   1</maxderiv>
  </var>
  <var>...</var>
  <var>...</var>
</output>
```

Required smoothness for this variable - C^{s-1}

Number of intervals

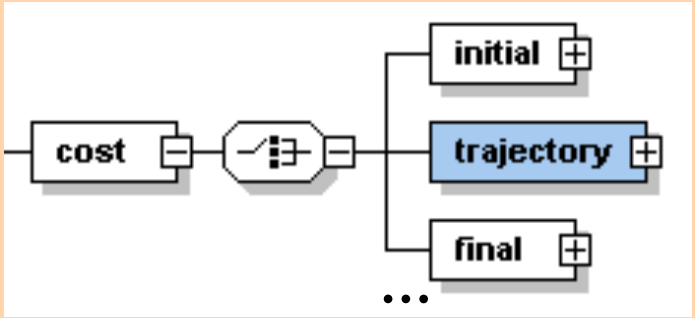
Order of the B-Spline functions

Maximum derivatives used for this variable

OCP specification – cost function

```
<ntgml ...>
  <output> ... </output>
  <lc>     ... </lc>
  <nlc>    ... </nlc>
  <cost>   ... </cost>
  <horizon>...</horizon>
  <misc>   ...</misc>
</ntgml>
```

Cost



$$\min_{u(t),x(t)} \frac{1}{2}\int_0^5 (x_1^2 + x_2^2 + u^2)dt$$

Integrand of the trajectory function

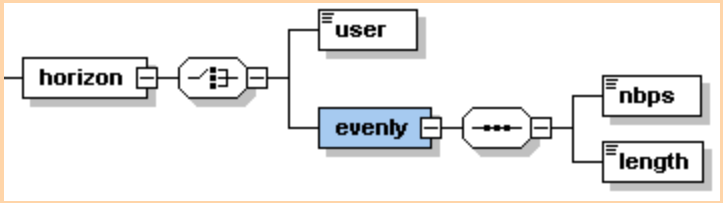
```
<cost>
  <trajectory>
    <func>
        ...
    </func>
  </trajectory>
</cost>
```

```
<func>
  <userfunc>
    <func>0.5*(x1*x1+x2*x2+u*u)</func>
    <grad>
      <name>x1</name>
      <func>x1</func>
    </grad>
    ...
    <grad>...</grad>
    ...
  </userfunc>
</func>
```

OCP specification – horizon time

```
<ntgml ...>
  <output> ... </output>
  <lc>     ... </lc>
  <nlc>    ... </nlc>
  <cost>   ... </cost>
  <horizon>...</horizon>
  <misc>   ...</misc>
</ntgml>
```

Horizon



NBPS – number of collocation points
(where the constraints are specified)

```
<horizon>
  <evenly>
    <nbps> 20</nbps>
    <length>5</length>
  </evenly>
</horizon>
```

$NBPS$ – number of collocation points

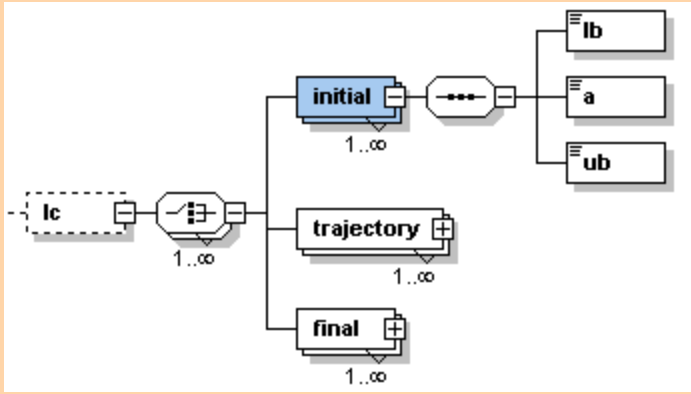
T – length of the horizon in seconds

OCP specification – linear constraints

```
<ntgml ...>
  <output> ... </output>
  <lc>     ... </lc>
  <nlc>    ... </nlc>
  <cost>   ... </cost>
  <horizon>...</horizon>
  <misc>   ...</misc>
</ntgml>
```

lc

$$x_1(0) = 1$$
$$x_2(0) = 0$$
$$-x_1(5) + x_2(5) = 1$$



Use this vector to define the linear constraint

$$z(t) = [x_1(t)\ \dot{x}_1(t)\ x_2(t)\ \dot{x}_2(t)\ u(t)]^T$$

Initial constraints

```
<lc>
  <initial>
    <lb>1</lb>
    <a>1 0 0 0 0</a>
    <ub>1</ub>
  </initial>
  <initial>
    <lb>0</lb>
    <a>0 0 1 0 0</a>
    <ub>0</ub>
  </initial>
...
```

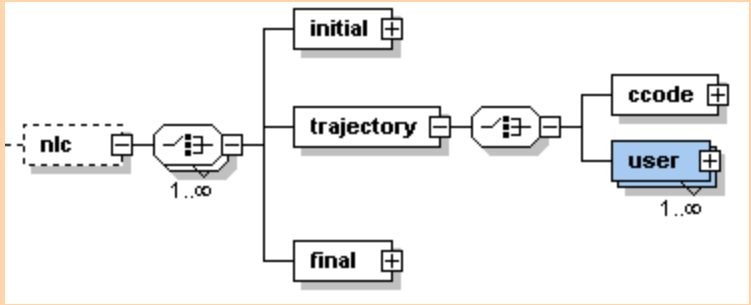
Final constraint

```
...
  <final>
    <lb>1</lb>
    <a>-1 0 1 0 0 </a>
    <ub>1</ub>
  </final>
</lc>
```

OCP specification – nonlinear constraints

```
<ntgml ...>
  <output> ... </output>
  <lc>     ... </lc>
  <nlc>    ... </nlc>
  <cost>   ... </cost>
  <horizon>...</horizon>
  <misc>   ...</misc>
</ntgml>
```

nlc



Trajectory constraints

$LB \le c(z) \le UB$

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -x_1 + (1 - x_1)^2 . x_2 + u$$

$$\dot{x}_1 - x_2 = 0$$
$$\dot{x}_2 + x_1 - (1 - x_1)^2 . x_2 - u = 0$$

Computation of the function gradients

```
<nlc>
  <trajectory>
...
    <user>
      <lb>0</lb>
      <ub>0</ub>
      <userfunc>
        <func>x2d+x1-(1-x1*x1)*x2-u</func>
        <grad>
          <name>x1</name>
          <func>1+2*x2*(1-x1)</func>
        </grad>
        ...
      </userfunc>
    </user>
  </trajectory>
</nlc>
```

Results

$$\min_{u(t),x(t)} \frac{1}{2}\int_0^5 (x_1^2 + x_2^2 + u^2)dt$$

Subject to:

$x_1(0) = 1$

$x_2(0) = 0$

$-x_1(5) + x_2(5) = 1$

where

$\dot{x}_1 = x_2$

$\dot{x}_2 = -x_1 + (1 - x_1)^2 . x_2 + u$

User specifies the problem in a xml file with the help of a grammar

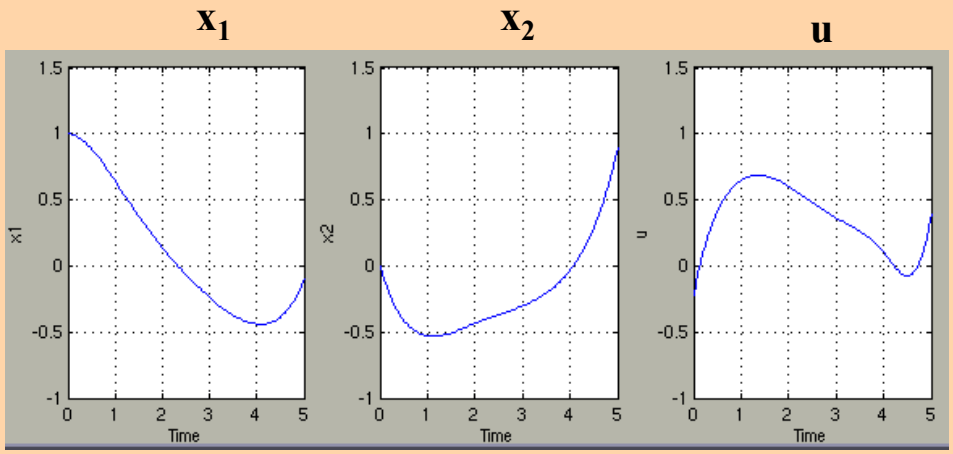
```
<ntgml ...>
  <output> ... <output>
  <lc> ... </lc>
  <nlc> ... </nlc>
  <cost> ... </cost>
  <horizon> ... </horizon>
  <misc> ... </misc>
</ntgml>
```

NTG

	Number of intervals (l)	Smoothness (s)	Order (k)
x_1	5	3	4
x_2	5	3	4
u	5	3	4

Total number of coefficients

$$\sum l_i(k_i - s_i) + s_i = 24$$



Cost = 1.47

Computation time = 0.03s (Pentium III @ Linux OS)

Results

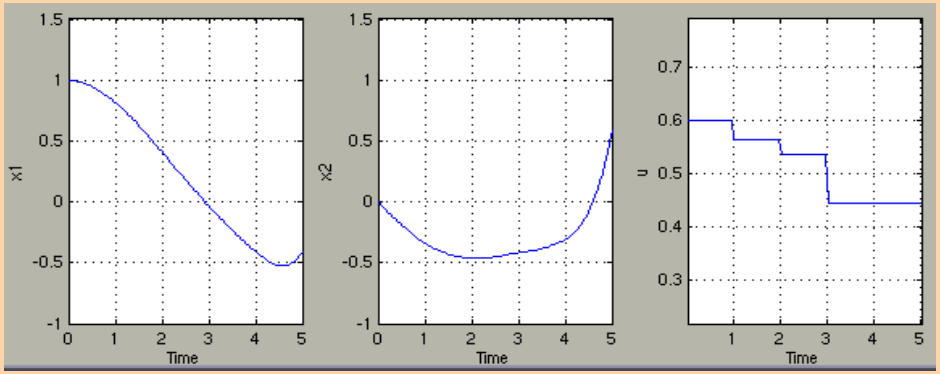
Variables

	Number of intervals	Smoothness	Order
x_1	5	3	4
x_2	5	3	4
u	5	0	1

For this problem it is possible to control the behavior of each variable independently.

Example: Allow the control to be piecewise constant

x_1 **x_2** **u**



Cost = 1.81

The cost is bigger because the space of the functions is smaller

Computational time = 0.02s
(Pentium III @ Linux OS)

The number of coefficients to compute is smaller than before so the computational time decreases

$$\sum l_i(k_i - s_i) + s_i = 21$$

Variables' parameters

Cost ⟺ Trade-off ⟺ Time

Improving computation time

Improve optimization

Parameterization of the state and input with one variable – z(t)

$$\min_{u(t),x(t)} \frac{1}{2}\int_0^5 (x_1^2 + x_2^2 + u^2)dt$$

Subject to:

$x_1(0) = 1$

$x_2(0) = 0$

$-x_1(5) + x_2(5) = 1$

where

$\dot{x}_1 = x_2$

$\dot{x}_2 = -x_1 + (1-x_1)^2 x_2 + u$

$$x_1(t) = z(t)$$

$$x_2(t) = \dot{x}_1(t) = \dot{z}(t)$$

$$u(t) = \dot{x}_2 + x_1 - (1-x_1)^2 x_2 = \ddot{z} + z - (1-z)^2 \dot{z}$$

$$\min_{z(t)} \frac{1}{2}\int_0^5 (z^2 + \dot{z}^2 + (\ddot{z} + z - (1-z^2)\dot{z})^2)dt$$

Subject to:

z(0)=1

ż(0)=0

-z(5)+ż(5)=1

Use differential flatness properties to find variables that parameterize the state and input of the system

No differential constraints

Performance comparison

	Variables	Total number of coefficients
Before	$x_1(t), x_2(t), \mathrm{u}(t)$	24
With parameterization	$z(t)$	$l(k-s)+s=6$

	Number of intervals (l)	Smoothness (s)	Order (k)
z	5	3	4

→ Same values as before

Computational savings

- Less trajectories to compute => less coefficients to compute =>better computational times
- No nonlinear constraints to consider

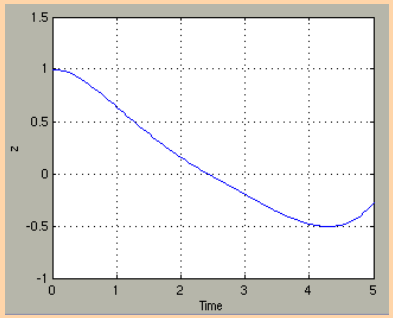
Disadvantages

- One more derivative to compute => increase order k
- Compute the state and input from the parameterization
- No individual control of the behavior of each state variable and input

Results with parameterization

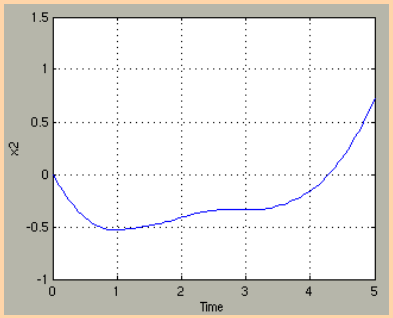
Computation of z(t)

z(t)

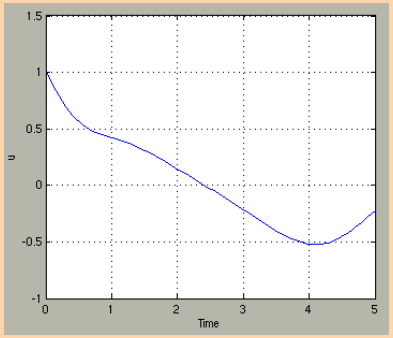


$$x_1(t) = z(t)$$
$$x_2(t) = \dot{z}(t)$$
$$u(t) = \ddot{z} + z - (1 - z)^2 \dot{z}$$

$x_2(t)$



u



Cost = 1.71

Computational time = 0.01s
(Pentium III @ Linux OS)

Computational more efficient

Difficulty: find variables that parameterize the state and input

Differential flat systems

End

Solving Open Final Time Problems Using NTG/NTGML

Raktim Bhattacharya

raktim@cds.caltech.edu

Open Final Time Problem - Vanderpol Oscillator

Original Optimal Control Problem

Cost Function

$$\min_{z(t)} \int_0^5 \left[z^2 + \dot{z}^2 + \{\ddot{z} + z - (1 - z^2)\dot{z}\}^2 \right] dt$$

Constraints

$$\begin{aligned} z(0) &= 1 \\ \dot{z}(0) &= 0 \\ -z(5) + \dot{z}(5) &= 1 \end{aligned}$$

Optimisation variable is $z(t)$.

Open Final Time Control Problem

Cost Function

$$\min_{z(t), t_f} \int_0^{t_f} \left[z^2 + \dot{z}^2 + \{\ddot{z} + z - (1 - z^2)\dot{z}\}^2 \right] dt$$

Constraints

$$\begin{aligned} z(0) &= 1 \\ \dot{z}(0) &= 0 \\ -z(t_f) + \dot{z}(t_f) &= 1 \end{aligned}$$

Optimisation variable is $z(t)$ and t_f.

Open Final Time Problem - Vanderpol Oscillator

Open Final Time Optimal Control Problem

Cost Function

$$\min_{z(t),t_f} \int_0^{t_f} \left[z^2 + \dot{z}^2 + \{\ddot{z} + z - (1 - z^2)\dot{z}\}^2 \right] dt$$

Constraints

$$\begin{aligned} z(0) &= 1 \\ \dot{z}(0) &= 0 \\ -z(t_f) + \dot{z}(t_f) &= 1 \end{aligned}$$

Optimisation variables are $z(t)$ and t_f

Normalise Time

$$\tau = t/t_f, \quad \tau \in [0, 1]$$

$$\frac{d}{dt} = \frac{1}{t_f}\frac{d}{d\tau}, \quad \frac{d^2}{dt^2} = \frac{1}{t_f^2}\frac{d^2}{d\tau^2}, \ldots$$

Define

$$z(t) = z(t_f\tau) = \xi(\tau)$$

$$\Rightarrow \quad \begin{aligned} \dot{z} &= \frac{\dot{\xi}}{t_f} \\ \ddot{z} &= \frac{\ddot{\xi}}{t_f^2} \end{aligned}$$

Open Final Time Problem - Vanderpol Oscillator

Open Final Time Optimal Control Problem

Cost Function

$$\min_{z(t),t_f} \int_0^{t_f} \left[z^2 + \dot{z}^2 + \{\ddot{z} + z - (1 - z^2)\dot{z}\}^2 \right] dt$$

Constraints

$$\begin{aligned} z(0) &= 1 \\ \dot{z}(0) &= 0 \\ -z(t_f) + \dot{z}(t_f) &= 1 \end{aligned}$$

Optimisation variables are $z(t)$ and t_f

Modified Control Problem

Cost Function

$$\min_{\xi(\tau),t_f} \int_0^1 \left[\xi^2 + \left(\frac{\dot{\xi}}{t_f} \right)^2 + \left\{ \frac{\ddot{\xi}}{t_f^2} + \xi - (1 - \xi^2)\frac{\dot{\xi}}{t_f} \right\}^2 \right] t_f d\tau$$

Constraints

$$\begin{aligned} \xi(0) &= 1 \\ \frac{\dot{\xi}(0)}{t_f} &= 0 \quad \Rightarrow \dot{\xi}(0) = 0 \\ -\xi(1) + \frac{\dot{\xi}(1)}{t_f} &= 1 \quad \Rightarrow -t_f\xi(1) + \dot{\xi}(1) - t_f = 0 \end{aligned}$$

Optimisation variables are $\xi(\tau)$ and t_f

Time Optimal Control Problem – Brachistochrone Problem

Problem:
A bead slides on a frictionless wire between points
A(0,0) and B(x_f,y_f) in a constant gravity field. Initial velocity = V_0.

What is the minimum-time path between the two points?

$$\min_{\theta(t)} t_f^2$$

such that

$$\begin{aligned} \dot{x} &= V(y)cos(\theta) \\ \dot{y} &= V(y)sin(\theta) \end{aligned}$$

$$\begin{aligned} x(0) &= 0 \\ y(0) &= 0 \\ x(t_f) &= x_{t_f} \\ y(t_f) &= y_{t_f} \end{aligned}$$

where

$$V(y) = \sqrt{V_0^2 + 2gy}$$

$$\min_{z_1(\tau),z_2(\tau),z_3} z_3^2$$

such that

$$\begin{aligned} z_1(0) &= 0 \\ z_2(0) &= 0 \\ z_1(1) &= x_{t_f} \\ z_2(1) &= y_{t_f} \\ z_3 &> 0 \\ \dot{z}_1^2 + \dot{z}_2^2 - z_3^2\left[V_0^2 + 2gz_2(\tau)\right] &= 0 \ \forall \tau \in [0,1] \end{aligned}$$

Choose $z_1 \equiv x$, $z_2 \equiv y$ and $z_3 \equiv t_f$.
Normalise Time.

Formulation Using NTGML - <OUTPUT>, <HORIZON>

```
<NTGML>
        <OUTPUT>
                <VAR>
                        <NAME>     z1  </NAME>
                        <NINTERV>  6   </NINTERV>
                        <MULT>     2   </MULT>
                        <ORDER>    4   </ORDER>
                        <MAXDERIV> 2   </MAXDERIV>
                </VAR>
                <VAR>
                        <NAME>     z2  </NAME>
                        <NINTERV>  6   </NINTERV>
                        <MULT>     2   </MULT>
                        <ORDER>    4   </ORDER>
                        <MAXDERIV> 2   </MAXDERIV>
                </VAR>
                <VAR>
                        <NAME>     z3  </NAME>
                        <NINTERV>  1   </NINTERV>
                        <MULT>     0   </MULT>
                        <ORDER>    1   </ORDER>
                        <MAXDERIV> 1   </MAXDERIV>
                </VAR>
        </OUTPUT>
        <HORIZON>
                <NBPS>   15  </NBPS>
                <LENGTH> 1   </LENGTH>
        </HORIZON>
        <LC>   …           </LC>
        <NLC>  …           </NLC>
        <COST> …           </COST>
</NTGML>
```

Formulation Using NTGML – Linear Constraints <LC>

```
</NTGML>. . .
          <LC>
                    <INITIAL>
                              <LB> 0   </LB>
                              <!-- z1   z1d z2  z2d z3 -->
                              <A>   1   0   0   0   0    </A>
                              <UB> 0 </UB>
                    </INITIAL>
                    <INITIAL>
                              <LB> 0   </LB>
                              <A>   0   0   1   0   0    </A>
                              <UB> 0 </UB>
                    </INITIAL>
                    <INITIAL>
                              <LB> 0.01 </LB>
                              <A>  0   0   0   0   1    </A>
                              <UB> 10000000 </UB>
                    </INITIAL>
                    <FINAL>
                              <LB> 1 </LB>
                              <A>  1   0   0   0   0    </A>
                              <UB> 1 </UB>
                    </FINAL>
                    <FINAL>
                              <LB> 1 </LB>
                              <A>  0   0   1   0   0    </A>
                              <UB> 1 </UB>
                    </FINAL>
          </LC>
. . .
</NTGML>
```

Formulation Using NTGML – Nonlinear Constraints <NLC>

```
<NTGML>  . . .
   <NLC>
     <TRAJECTORY> <USER>
              <LB> 0 </LB>
              <UB> 0 </UB>
              <USERFUNC>
                   <FUNC>
                      z1d*z1d + z2d*z2d - z3*z3*(0.1*0.1 + 2*9.806*z2)
                   </FUNC>
                   <GRAD>
                             <NAME> z1   </NAME>
                             <FUNC> 0    </FUNC>
                   </GRAD>
                   <GRAD>
                             <NAME> z1d    </NAME>
                             <FUNC> 2*z1d  </FUNC>
                   </GRAD>
                   <GRAD>
                             <NAME> z2               </NAME>
                             <FUNC> -2*9.806*z3*z3   </FUNC>
                   </GRAD>
                   <GRAD>
                             <NAME> z2d   </NAME>
                             <FUNC> 2*z2d </FUNC>
                   </GRAD>
                   <GRAD>
                             <NAME> z3   </NAME>
                             <FUNC> -2*z3*(0.1*0.1 + 2*9.806*z2) </FUNC>
                   </GRAD>
               </USERFUNC>
          </USER> </TRAJECTORY>
</NLC>
          . . .
</NTGML>
```

Order of gradients

z_1 z_1d z2 z_2d z3

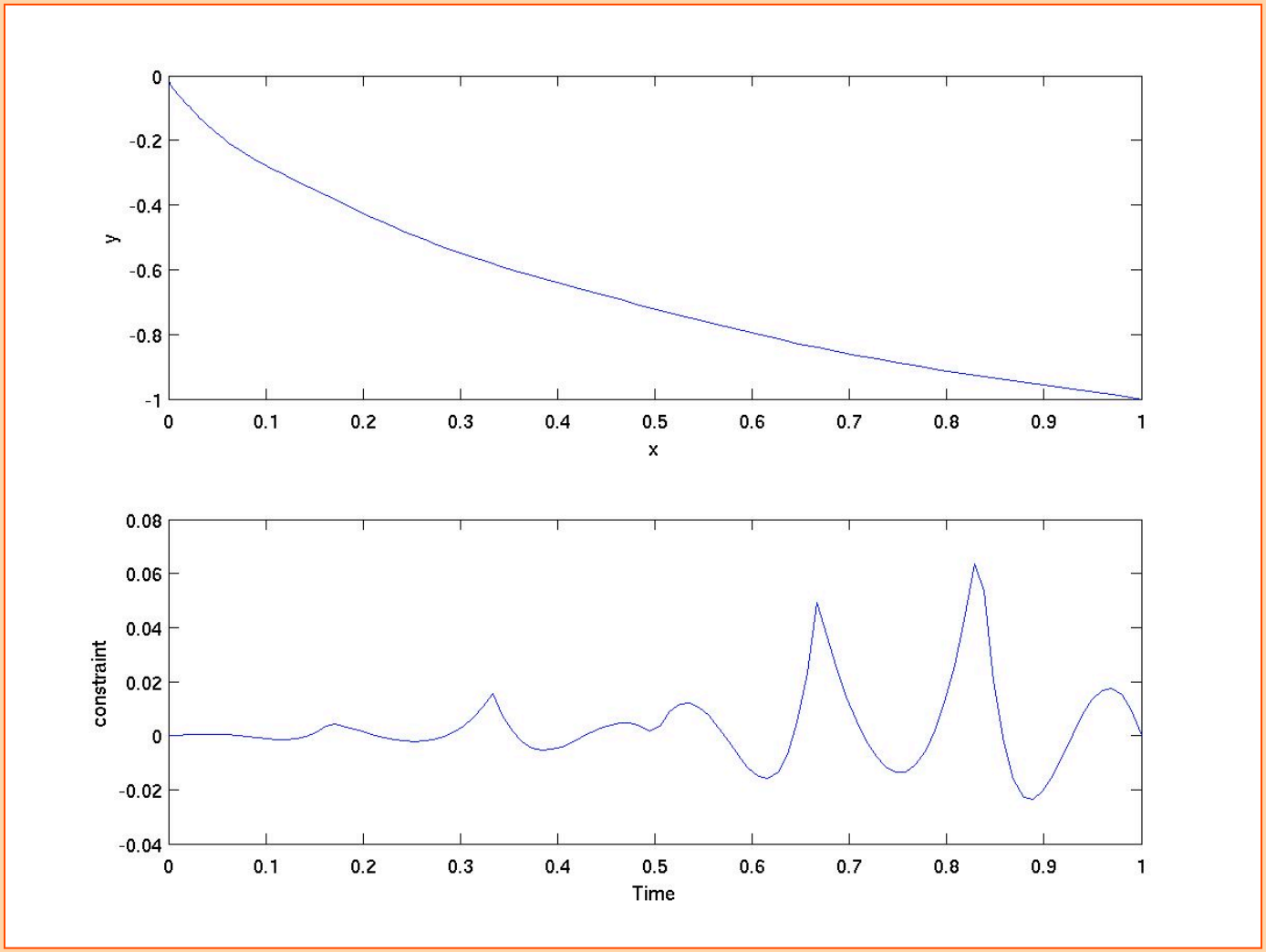
Formulation Using NTGML – COST <COST>

```
<NTGML>  . . .
   <COST>
     <FINAL> <USERFUNC>
              <LB> 0 </LB>
              <UB> 0 </UB>
                   <FUNC>
                      z3*z3
                   </FUNC>
                   <GRAD>
                              <NAME> z1   </NAME>
                              <FUNC> 0    </FUNC>
                   </GRAD>
                   <GRAD>
                              <NAME> z1d    </NAME>
                              <FUNC> 0     </FUNC>
                   </GRAD>
                   <GRAD>
                              <NAME> z2   </NAME>
                              <FUNC> 0    </FUNC>
                   </GRAD>
                   <GRAD>
                              <NAME> z2d   </NAME>
                              <FUNC> 0     </FUNC>
                   </GRAD>
                   <GRAD>
                              <NAME> z3   </NAME>
                              <FUNC> 2*z3     </FUNC>
                   </GRAD>
          </USERFUNC> </FINAL>
</NLC>
          . . .
</NTG>
```

Order of gradients

z_1 z_1d z2 z_2d z3

Code Generation and Plots



Applications of NTG: Vanderpol Oscillator Example

Tamer Inanc

Control and Dynamical Systems

The California Institute of Technology

tinanc@cds.caltech.edu

September 16, 2003

Vanderpol Oscillator

Optimal Control Problem

$$\min_{u(t)} \quad \frac{1}{2}\int_0^5 (x_1^2 + x_2^2 + u^2)dt$$

subject to:

$$\begin{aligned} x_1(0) &= 1 \\ x_2(0) &= 0 \\ -x_1(5) + x_2(5) &= 1 \\ \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -x_1 + (1 - x_1^2)x_2 + u \end{aligned}$$

Choosing Flat Outputs:

$$z(t) \equiv x_1(t)$$

problem becomes:

$$\min_{z(t)} \quad \frac{1}{2}\int_0^5 \left[z^2 + \dot{z}^2 + \{\ddot{z} + z - (1 - z^2)\dot{z}\}^2\right] dt$$

subject to:

$$\begin{aligned} z(0) &= 1 \\ \dot{z}(0) &= 0 \\ -z(5) + \dot{z}(5) &= 1 \end{aligned}$$

Let's Start Programming with NTG

- `#include "ntg.h"`

Define Global Variables:

- `#define NOUT 1` → # of flat outputs, $z = x_1$
- `#define NINTERV 2` → # of intervals, l_j
- `#define SMTH 3` → smoothness of splines, s_j
- `#define ORDER 5` → degree of spline polynomial, k_j
- `#define MAXDERIV 2` → highest derived required
- `#define NCOEF 7` → total # of coeffs, $\sum_j l_j \cdot (k_j - s_j) + s_j$

Define outputs and derivatives:

- `#define z zp[0][0]`
- `#define zd zp[0][1]`
- `#define zdd zp[0][2]`

format: zp [output #] [derivative #]

Define the # of constraints:

$$\left.\begin{array}{l} z(0) = 1 \\ \dot{z}(0) = 0 \end{array}\right\} \rightarrow \boldsymbol{NLIC}$$

$$\left.-z(5) + \dot{z}(5) = 1\right\} \rightarrow \boldsymbol{NLFC}$$

Linear constraints:

- **`#define NLIC`** **`2`** → Number of Linear Initial Constraints
- **`#define NLTC`** **`0`** → # of linear trajectory constraints
- **`#define NLFC`** **`1`** → # of linear final constraints

Nonlinear constraints:

- **`#define NNLIC`** **`0`** → Number of NonLinear Initial Constraints
- **`#define NNLTC`** **`0`** → # of nonlinear trajectory constraints
- **`#define NNLFC`** **`0`** → # of nonlinear final constraints

Define nonlinear constraint active variables:

of active variables (AV) that show up in the *nonlinear* constraints

- **`#define NINITIALCONSTRAV`** `0` →
- **`#define NTRAJECTORYCONSTRAV`** `0` →
- **`#define NFINALCONSTRAV`** `0` →

→(# of initial constraint active variables)
→(# of trajectory constraint active variables)
→(# of final constraint active variables)

In this example we do not have any nonlinear constraints

Define cost functions and their active variables:

$$\min_{z(t)} \quad \frac{1}{2}\int_0^5 \left[z^2 + \dot{z}^2 + \{\ddot{z} + z - (1 - z^2)\dot{z}\}^2\right] dt$$

- **`#define NICF 0`** → Number of Initial Cost Function
- **`#define NTCF 1`** → # of trajectory cost function
- **`#define NFCF 0`** → # of final cost function

- **`#define NINITIALCOSTAV 0`** → # of initial cost AV
- **`#define NTRAJECTORYCOSTAV 3`** → # of trajectory cost AV
- **`#define NFINALCOSTAV 0`** → #of final cost AV

Define all active variables for nonlinear constraint and cost functions:

- **`static AV trajectorycostav[NTRAJECTORYCOSTAV]=`** $\{\underbrace{\{0,0\}}_{z}, \underbrace{\{0,1\}}_{z_d}, \underbrace{\{0,2\}}_{z_{dd}}\};$

Function Declarations and main( ) function:

- `void tcf(int *mode, int *nstate, int *i, double *f, double *df, double **zp);`

main( )

- define NTG parameters listed previously

Initialize →

the break points *(for this problem, regularly spaced from t=0 to t=5)*:

- `linspace(breaks[0], 0, 5, ninterv[0]+1);`

coefficients:

- `linspace(coefficients,1,1,ncoef);`

and collocation points:

- `linspace(cps, 0, 5, ncps);`

Define the constraints

format: lic [constraint #][active variable #]

Linear Initial Constraint:

- `lic[0][0] = 1.0;`
- `lic[1][1] = 1.0;`

$\leftarrow \begin{cases} z(0) = 1 \\ \dot{z}(0) = 0 \end{cases}$

Linear Trajectory Constraint:

- `// None`

Linear Final Constraints:

- `lfc[0][0] = -1.0;`
- `lfc[0][1] = 1.0;`

$\leftarrow \{ -z(5) + \dot{z}(5) = 1$

Define the bounds

- `lowerb[0] = upperb[0] = 1.0;` → *linear initial constraint*

  `lowerb[1] = upperb[1] = 0.0;`

  `lowerb[2] = upperb[2] = 1.0;` → *linear final constraint*

format: lowerb [increasing order starting from 0]

Define the subroutines

In this example we do not have any nonlinear constraint function

tcf( ): $\min_{z(t)} \ \frac{1}{2}\int_0^5 \left[z^2 + \dot{z}^2 + \{\ddot{z} + z - (1 - z^2)\dot{z}\}^2\right] dt$

- ```
  if (*mode = 0 || *mode = 2)
    {
     *f = 0.5*(z*z + zd*zd + pow(zdd + z - (1.0 - z*z)*zd , 2.0));
    }

  if (*mode  = 1 || *mode = 2)
    {
     df[0] = z + (zdd + z - (1.0 - z*z)*zd )*(1.0 + 2.0*z*zd);
     df[1] = zd - (zdd + z - (1.0 - z*z)*zd )*(1.0 - z*z);
     df[2] = zdd + z - (1.0 - z*z)*zd;
    }
  ```

format: **df [in the order of z, zd, zdd]**
  ```

Call ntg( )

NPSOL options:

- npsoloption("warm start")
- ….“major iteration limit”
- ….“minor iteration limit”
- ….“line search tolerance”
- ….“hessian”

*User's Guide For NPSOL 5.0, Philip E. Gill, Walter Murray, Michael A. Saunders, Technical Report SOL 86-1, 1998

- ntg(NOUT, cps, ncps, ninterv, breaks, order, smth, maxderiv, coefficients,
  NLIC, lic,
  NLTC, NULL,
  NLFC, lfc,
  NNLIC, NULL,
  NNLTC, NULL,
  NNLFC, NULL,
  NINITIALCONSTRAV, NULL,
  NTRAJECTORYCONSTRAV, NULL,
  NFINALCONSTRAV, NULL,
  lowerb, upperb,
  NICF, NULL,
  NUCF, tcf,
  NFCF, NULL,
  NINITIALCOSTAV, NULL,
  NTRAJECTORYCOSTAV, trajectorycostav,
  NFINALCOSTAV, NULL,
  istate, clambda, R, &inform, &objective);

NPSOL Parameters

- **`istate:`** An integer array. istate(j) indicates whether constraint

  $r_j(x) \geq lb_j$ **or** $r_j(x) \leq ub_j$ is expected to be active at a solution of NLP

- **`clamda:`** An integer array. The components of clamda corresponding to nonlinear constraints must contain multiplier estimate as follows:

inactive $r_j(x)$	$istate(j) = 0$	$clamda(j) = 0$
$r_j(x) \geq lb_j$	$istate(j) = 1$	$clamda(j) > 0$
$r_j(x) \leq ub_j$	$istate(j) = 2$	$clamda(j) < 0$

- **`R:`** For a Cold Start, it is taken as the identity. For a Warm Start, it provides the upper-triangular Cholesky factor of the initial approx. of the Hessian of the Lagrangian.

- **`inform:`** reports the result of the call to NPSOL. The possible values:

0	*The iterates have converged to a point x that satisfies the optimality conditions*
1	*The final iterate x satisfies the optimality conditions to the accuracy requested, but the sequence of iterates has not converged yet*
2	*The linear constraints and bounds could not be satisfied. No feasible solution*
3	*The nonlinear constraints and bounds could not be satisfied. No feasible solution*
⋮	⋮

Solution?

- NTG gives the coefficients of the *B-Splines*, i.e., gives in: C_i

$$z = \sum_{i=1}^{ncoef} B_{i,k} \cdot C_i$$

- Can use a MATLAB program to calculate the output *OR*

- ```
  for (i=0; i<ncps; i++)
   {
    time[i] = 5.0*(double)(i)/(ncps -1);
    Splineinterp(fz,time[i],breaks,ninterv,coefficients,ncoef,order,smth,3)
    z_result[i]   = fz[0];
    zd_result[i]  = fz[1];
    zdd_result[i] = fz[2];
   }
  ```

$$\begin{aligned} x_1 &= z & &= z_{result} \\ x_2 &= \dot{x}_1 & &= zd_{rezult} \\ u &= \dot{x}_2 + x_1 - (1 - x_1^2)x_2 & &= zdd_{result} + z_{result} - (1 - z_{result}^2)zd_{result} \end{aligned}$$
  ```

Open Final Time Problem with

- ***A* nonlinear trajectory constraint (NLTCF):**

`nltcf( )` → $0 \le x_1^2(t) + x_2^2(t) \le 10 \;\Rightarrow$

$0 \le z^2(t) + \dot{z}^2(t) \le 10 \qquad (\dot{x}_1 = x_2)$

$$\min_{z(t)} \quad \frac{1}{2}\int_0^{t_f} \left[z^2 + \dot{z}^2 + \{\ddot{z} + z - (1 - z^2)\dot{z}\}^2\right] dt$$

subject to:

$$\begin{aligned} z(0) &= 1 \\ \dot{z}(0) &= 0 \\ -z(tf) + \dot{z}(tf) &= 1 \\ 0 \le z^2(t) + \dot{z}^2(t) &\le 10 \end{aligned}$$

Open Final Time Problem with a NLTCF

temporal constraint:

$\tau = t/t_f, \quad \tau \in [0,1]$

τ : ***scaled time***, t_f : ***unknown final time***

$$\frac{d}{dt} = \frac{1}{t_f}\frac{d}{d\tau}, \quad \frac{d^2}{dt^2} = \frac{1}{t_f^2}\frac{d^2}{d\tau^2}, \cdots$$

$$z(t) = z(t_f\tau) = \xi(\tau) \quad \Rightarrow \quad \dot{z} = \frac{\xi'}{t_f} \longrightarrow \xi' = \frac{d\xi}{d\tau}$$

$$\ddot{z} = \frac{\xi''}{t_f^2} \longrightarrow \xi'' = \frac{d^2\xi}{d\tau^2}$$

$$\min_{\xi(\tau),t_f} \int_0^1 \left[\xi^2 + \left(\frac{\xi'}{t_f}\right)^2 + \left\{\frac{\xi''}{t_f^2} + \xi - (1-\xi^2)\frac{\xi'}{t_f}\right\}^2\right] t_f d\tau$$

subject to:

$$\xi(0) = 1$$

$$\frac{\xi'(0)}{t_f} = 0 \quad \Rightarrow \xi'(0) = 0$$

$$-\xi(1) + \frac{\xi'(1)}{t_f} = 1 \quad \Rightarrow -t_f\xi(1) + \xi'(1) - t_f = 0$$

$$0 \le \xi^2(\tau) + \left(\frac{\xi'(\tau)}{t_f}\right)^2 \le 10$$

Open Final Time Problem with a NLTCF

$$\min_{\xi(\tau),t_f} \int_0^1 \left[\xi^2 + \left(\frac{\xi'}{t_f}\right)^2 + \left\{\frac{\xi''}{t_f^2} + \xi - (1-\xi^2)\frac{\xi'}{t_f}\right\}^2\right] t_f d\tau$$

subject to:

$$\begin{aligned} \xi(0) &= 1 \\ \frac{\xi'(0)}{t_f} &= 0 \;\Rightarrow \xi'(0) = 0 \\ -\xi(1) + \frac{\xi'(1)}{t_f} &= 1 \;\Rightarrow -t_f\xi(1) + \xi'(1) - t_f = 0 \\ 0 \le \xi^2(\tau) + \left(\frac{\xi'(\tau)}{t_f}\right)^2 &\le 10 \end{aligned}$$

Choose:

$$\begin{aligned} z_1 &= \xi(\tau) \\ z_2 &= t_f \end{aligned}$$

$$\min_{z_1(\tau),z_2} \int_0^1 \left[z_1^2 + \left(\frac{z_1'}{z_2}\right)^2 + \left\{\frac{z_1''}{z_2^2} + z_1 - (1-z_1^2)\frac{z_1'}{z_2}\right\}^2\right] z_2 d\tau$$

subject to:

$$\left.\begin{aligned} z_1(0) &= 1 \\ z_1'(0) &= 0 \end{aligned}\right\} \longrightarrow \text{Linear Initial Constraints}$$

$$\left.-z_2 z_1(1) + z_1'(1) - z_2 = 0\right\} \longrightarrow \text{NonLinear Final Constraint}$$

$$\left.0 \le z_1^2(\tau) + \left(\frac{z_1'(\tau)}{z_2}\right)^2 \le 10\right\} \longrightarrow \text{NonLinear Trajectory Constraint}$$

Programming with NTG - Open Final Time Problem with a NLTCF

- `#include "ntg.h"`

Define Global Variables:

- `#define NOUT 2` → # of flat outputs, $z_1 = \xi,\ z_2 = \tau$
- `#define NINTERV 2,1` → # of intervals, l_j
- `#define SMTH 3,1` → smoothness of splines, s_j
- `#define ORDER 5,1` → degree of spline polynomial, k_j
- `#define MAXDERIV 2,0` → highest derived required
- `#define NCOEF 8` → total # of coeffs, $\sum_j l_j \cdot (k_j - s_j) + s_j$

Define outputs and derivatives:

- `#define z1 zp[0][0]`
- `#define z1d zp[0][1]`
- `#define z1dd zp[0][2]`
- `#define z2 zp[1][0]`

format: zp [output #] [derivative #]

Define the # of constraints:

Previously:

$$\left.\begin{array}{l} z(0) = 1 \\ \dot{z}(0) = 0 \end{array}\right\} \rightarrow NLIC$$

$$-z(5) + \dot{z}(5) = 1 \Big\} \rightarrow NLFC$$

$$\left.\begin{array}{rcl} z_1(0) & = & 1 \\ z_1'(0) & = & 0 \end{array}\right\} \longrightarrow \text{Linear Initial Constraints (NLIC)}$$

$$-z_2 z_1(1) + z_1'(1) - z_2 \;=\; 0 \;\Big\} \longrightarrow \text{NonLinear Final Constraint (NNLFC)}$$

$$0 \le z_1^2(\tau) + \left(\frac{z_1'(\tau)}{z_2}\right)^2 \le 10 \;\Big\} \longrightarrow \text{NonLinear Trajectory Constraint (NNLTC)}$$

Linear constraints:

- **`#define NLIC 2`** → Number of Linear Initial Constraints
- **`#define NLTC 0`** → # of linear trajectory constraints
- **`#define NLFC 0`** → # of linear final constraints

Nonlinear constraints:

- **`#define NNLIC 0`** → Number of NonLinear Initial Constraints
- **`#define NNLTC 1`** → # of nonlinear trajectory constraints
- **`#define NNLFC 1`** → # of nonlinear final constraints

Define nonlinear constraint active variables:

of active variables (AV) that show up in the *nonlinear* constraints

$$0 \le z_1^2(\tau) + \left(\frac{z_1'(\tau)}{z_2}\right)^2 \le 10 \quad \Big\} \longrightarrow \textit{NonLinear Trajectory Constraint (NNLTC)}$$

$$-z_2 z_1(1) + z_1'(1) - z_2 \;=\; 0 \quad \Big\} \longrightarrow \textit{NonLinear Final Constraint (NNLFC)}$$

- **`#define NINITIALCONSTRAV`** 0 →
- **`#define NTRAJECTORYCONSTRAV`** 3
  →
- **`#define NFINALCONSTRAV`** 3 →

→(# of initial constraint active variables)

→(# of trajectory constraint active variables)

→(# of final constraint active variables)

Define cost functions and their active variables:

$$\min_{z_1(\tau),z_2} \quad z_2 + \int_0^1 \left[z_1^2 + \left(\frac{z_1'}{z_2}\right)^2 + \left\{ \frac{z_1''}{z_2^2} + z_1 - (1 - z_1^2)\frac{z_1'}{z_2} \right\}^2 \right] z_2 d\tau$$

- **`#define NICF`** **`1`** → Number of Initial Cost Function
- **`#define NTCF`** **`1`** → # of trajectory cost function
- **`#define NFCF`** **`0`** → # of final cost function

- **`#define NINITIALCOSTAV`** **`1`** → # of initial cost AV
- **`#define NTRAJECTORYCOSTAV`** **`4`** → # of trajectory cost AV
- **`#define NFINALCOSTAV`** **`0`** → #of final cost AV

Define all active variables for nonlinear constraint and cost functions:

$$\min_{z_1(\tau),z_2} \; z_2 + \int_0^1 \left[z_1^2 + \left(\frac{z_1'}{z_2}\right)^2 + \left\{ \frac{z_1''}{z_2^2} + z_1 - (1-z_1^2)\frac{z_1'}{z_2} \right\}^2 \right] z_2 d\tau$$

- `#define NINITIALCONSTRAV 0`
- `#define NTRAJECTORYCONSTRAV 3`
- `#define NFINALCONSTRAV 3`
- `#define NINITIALCOSTAV 1`
- `#define NTRAJECTORYCOSTAV 4`
- `#define NFINALCOSTAV 0`

$$0 \le z_1^2(\tau) + \left(\frac{z_1'(\tau)}{z_2}\right)^2 \le 10 \;\Big\} \longrightarrow \textit{NonLinear Trajectory Constraint (NNLTC)}$$

$$-z_2 z_1(1) + z_1'(1) - z_2 \;=\; 0 \;\Big\} \longrightarrow \textit{NonLinear Final Constraint (NNLFC)}$$

- `//static AV initialconstrav[NINITIALCONSTRAV]={};`
- `static AV trajectoryconstrav[NTRAJECTORYCONSTRAV]=` $\{\underbrace{\{0,0\}}_{z_1},\underbrace{\{0,1\}}_{z_{1d}},\underbrace{\{1,0\}}_{z_2}\};$
- `static AV finalconstrav[NFINALCONSTRAV]=` $\{\{0,0\},\{0,1\},\{1,0\}\};$
- `static AV initialcostav[NINITIALCOSTAV]=` $\{\{1,0\}\};$
- `static AV trajectorycostav[NTRAJECTORYCOSTAV]=` $\{\underbrace{\{0,0\}}_{z_1},\underbrace{\{0,1\}}_{z_{1d}},\underbrace{\{0,2\}}_{z_{1dd}},\underbrace{\{1,0\}}_{z_2}\};$
- `//static AV finalcostav[NFINALCOSTAV]={};`

Function Declarations and main( ) function:

- `//void nlicf(int *mode, int *nstate, int *i, double *f, double **df, double**zp);`
- `void nltcf(int *mode, int *nstate, int *i, double *f, double **df, double**zp);`
- `void nlfcf(int *mode, int *nstate, int *i, double *f, double **df, double**zp);`
- `void icf(int *mode, int *nstate, int *i, double *f, double *df, double**zp);`
- `void tcf(int *mode, int *nstate, int *i, double *f, double *df, double**zp);`
- `//void fcf(int *mode, int *nstate, int *i, double *f, double *df, double**zp);`

main( )

- define NTG parameters listed previously

Initialize →

the break points *(for this problem, regularly spaced from* $\tau = 0 \ \ to \ \ 1$ *)*

- `linspace(breaks[0], 0, 1.0, ninterv[0]+1);`
- `breaks[1] ={0.0, 1.0};` → For $z_2 = t_f$

coefficients:

- `linspace(coefficients,1,1,ncoef);`
- `coefficients[ncoef] = 1.0;` → For $z_2 = t_f$

and collocation points:

- `linspace(cps, 0, 1.0, ncps);`

Define the constraints

format: **lic [constraint #] [active variable #]**

Linear Initial Constraint:

- `lic[0][0] = 1.0;`
- `lic[1][1] = 1.0;`

$$\leftarrow \begin{cases} z_1(0) = 1 \\ z_1'(0) = 0 \end{cases}$$

Linear Trajectory Constraint:

- // None

Linear Final Constraints:

- //None

$$0 \le z_1^2(\tau) + \left(\frac{z_1'(\tau)}{z_2}\right)^2 \le 10 \quad \Big\} \longrightarrow \textit{NonLinear Trajectory Constraint (NNLTC)}$$

$$-z_2 z_1(1) + z_1'(1) - z_2 = 0 \quad \Big\} \longrightarrow \textit{NonLinear Final Constraint (NNLFC)}$$

Define the bounds

- `lowerb[0] = upperb[0] = 1.0;` → *linear initial constraints*

  `lowerb[1] = upperb[1] = 0.0;`

  `lowerb[2] = 0.0; upperb[2] = 10.0;` → *nonlinear trajectory constraint*

  `lowerb[3] = upperb[3] = 0.0;` → *nonlinear final constraint*

format: **lowerb [increasing order starting from 0 for all constraints]**

Define the cost subroutines

$$\min_{z_1(\tau),z_2} \quad z_2 + \int_0^1 \left[z_1^2 + \left(\frac{z_1'}{z_2}\right)^2 + \left\{ \frac{z_1''}{z_2^2} + z_1 - (1 - z_1^2)\frac{z_1'}{z_2} \right\}^2 \right] z_2 d\tau$$

icf( ):

- ```
  if (*mode = 0 || *mode = 2)
   {
       *f =  z2;
   }

  if (*mode  = 1 || *mode = 2)
   {
       df[0] = 0;      → w.r.t. z1
       df[1] = 0;      → w.r.t. z1d
       df[2] = 0;      → w.r.t. z1dd
       df[3] = 1.0;    → w.r.t. z2
   }
  ```

format: **df [in the order of z1, z1d, z1dd, z2]**
  ```

Define the cost subroutines

$$\min_{z_1(\tau),z_2} \quad z_2 + \frac{1}{2}\int_0^1 \left[z_1^2 + \left(\frac{z_1'}{z_2}\right)^2 + \left\{ \frac{z_1''}{z_2^2} + z_1 - (1-z_1^2)\frac{z_1'}{z_2} \right\}^2 \right] z_2 d\tau$$

tcf( ):

- ```
  if (*mode = 0 || *mode = 2)
    {
    *f = 0.5*((z1*z1 + z1d*z1d/(z2*z2) +
         pow(z1dd/(z2*z2) + z1 - (1.0 - z1*z1)*z1d/z2 , 2.0)))z2;
    }
  ```

```
if (*mode  = 1 || *mode = 2)
   {
   df[0] = (z1 +  (z1dd/z2*z2 + z1 - (1.0 - z1*z1)*z1d/z2 )*(1.0 + 2.0*z1*z1d/z2))*z2;

   df[1] = (z1d/z2*z2 - (z1dd/z2*z2 + z1 - (1.0 - z1*z1)*z1d/z2 )*(1.0 - z1*z1)/z2)*z2;

   df[2] = (z1dd/z2*z2 + z1 - (1.0 - z1*z1)*z1d/z2}*1/z2*z2)*z2;

   df[3] = -z1d*z1d/z2*z2 + (z1dd/z2 + z1*z2 - (1 - z1*z1)*z1d)*
          (-2*z1dd/z2*z2 + (1 - z1*z1)*z1d/z2) + (z1*z1 + z1d*z1d/(z2*z2) +
          pow(z1dd/(z2*z2) + z1 - (1.0 - z1*z1)*z1d/z2 , 2.0));

   }
```

format: df [in the order of z1, z1d, z1dd, z2]
```

Define the nonlinear constraint subroutines

$$0 \le z_1^2(\tau) + \left(\frac{z_1'(\tau)}{z_2}\right)^2 \le 10 \quad \Big\} \longrightarrow \textit{NonLinear Trajectory Constraint (NNLTC)}$$

nltcf( ):

```
• if (*mode = 0 || *mode = 2)
    {
        f[0] = z1*z1 + z1d*z1d/(z2*z2);
    }

  if (*mode  = 1 || *mode = 2)
    {
        df[0][0] = 2*z1;                    → w.r.t. z1

        df[0][1] = 2*(z1d/(z2*z2);          → w.r.t. z1d

        df[0][2] = 0.0;                     → w.r.t. z1dd

        df[0][3] = -2*(z1d/z2*z1d/(z2*z2); → w.r.t. z2
    }
```

format: **df [in the order of z1, z1d, z1dd, z2]**

Define the nonlinear constraint subroutines

$$-z_2 z_1(1) + z_1'(1) - z_2 \;=\; 0 \;\Big\} \longrightarrow$$ *NonLinear Final Constraint (NNLFC)*

nlfcf( ):

```
if (*mode = 0 || *mode = 2)
  {
       f[0] = -z2*z1 + z1d – z2;
  }

if (*mode  = 1 || *mode = 2)
  {
       df[0][0] = -z2;          → w.r.t. z1

       df[0][1] = 1.0;          → w.r.t. z1d

       df[0][2] = 0.0;          → w.r.t. z1dd

       df[0][3] = -z1 -1.0;     → w.r.t. z2
  }
```

format: **df [in the order of z1, z1d, z1dd, z2]**

Call ntg( )

- ```
  ntg(NOUT, cps, ncps, ninterv, breaks, order, smth,
      maxderiv, coefficients,
   NLIC,                                              lic,
      NLTC,                                             NULL,
      NLFC,                                             NULL,
      NNLIC,                                            NULL,
      NNLTC,                                            nltcf,
      NNLFC,                                            nlfcf,
      NINITIALCONSTRAV,                          NULL,
      NTRAJECTORYCONSTRAV,                       trajectoryconstrav,
      NFINALCONSTRAV,                    finalconstrav,
   lowerb, upperb,
   NICF,                                              icf,
      NUCF,                                             tcf,
      NFCF,                                             NULL,
      NINITIALCOSTAV,                            initialcostav,
      NTRAJECTORYCOSTAV,
      trajectorycostav,
   NFINALCOSTAV,                                      NULL,
      istate, clambda, R, &inform, &objective);
  ```
  ```

Solution?

- NTG gives the coefficients of the *B-Splines*, i.e., gives in: C_i

$$z = \sum_{i=1}^{ncoef} B_{i,k} \cdot C_i$$

- ```
  for (i=0; i<ncps; i++)
    {
       time[i] = 1.0*(double)(i)/(ncps -1);
       Splineinterp(fz, time[i], breaks, ninterv, coefficients,
                    ncoef, order, smth, 3)
       z1_result[i] = fz[0];
       z1d_result[i] = fz[1];
       z1dd_result[i] = fz[2];
       }
  ```

$$\left.\begin{array}{lllll} t_f & = & z2 & & \\ x_1 & = & z1 & = & z1_{result} \\ x_2 & = & \dot{x}_1 & = & z1d_{result} \cdot \frac{1}{t_f} \\ \dot{x}_2 & = & \ddot{x}_1 & = & z1dd_{result} \cdot \frac{1}{t_f^2} \end{array}\right\} \Rightarrow \; u = \dot{x}_2 + x_1 - (1 - x_1^2)x_2$$
  ```

Using Tables in NTG
&
B-Splines Revisited

Tamer Inanc

Control and Dynamical Systems

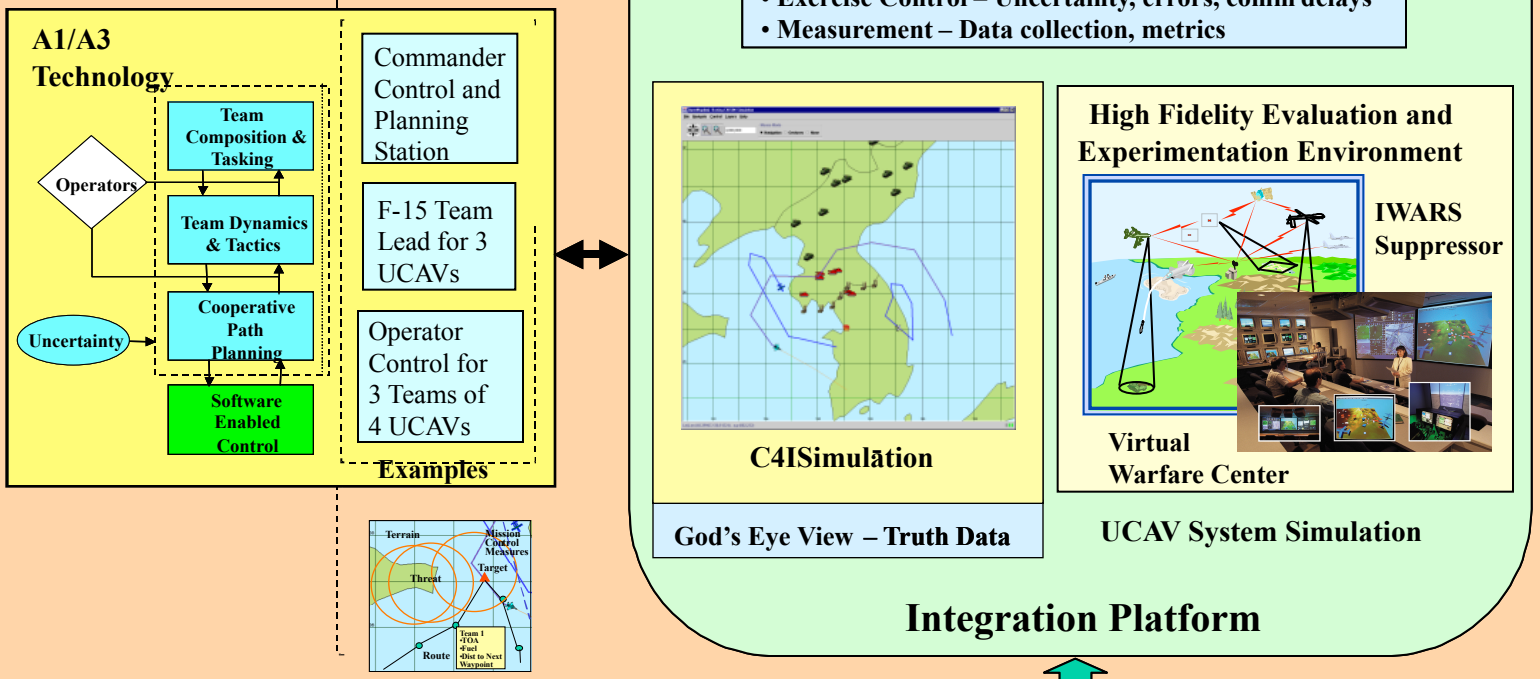
The California Institute of Technology

tinanc@cds.caltech.edu

September 16, 2003

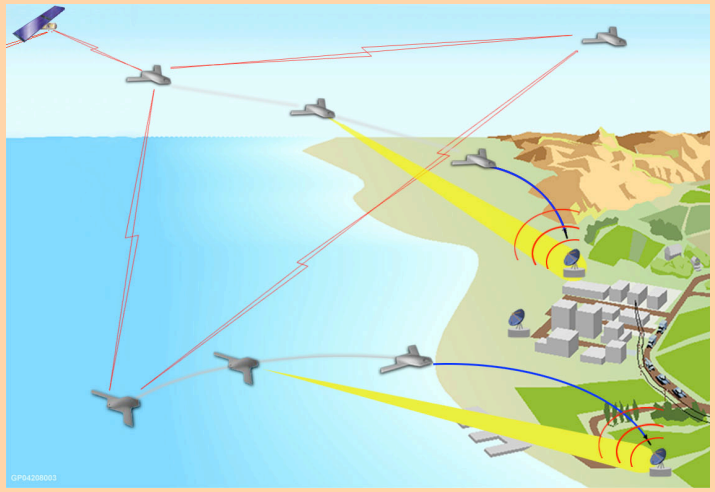
Motivation:
Example - NTG Temporal Method for Low Observability (LO)

Open Experimental Platform (OEP):
An integration platform to evaluate and validate MICA technologies



A New OEP Feature

OEP: A common battleground developed by Boeing.



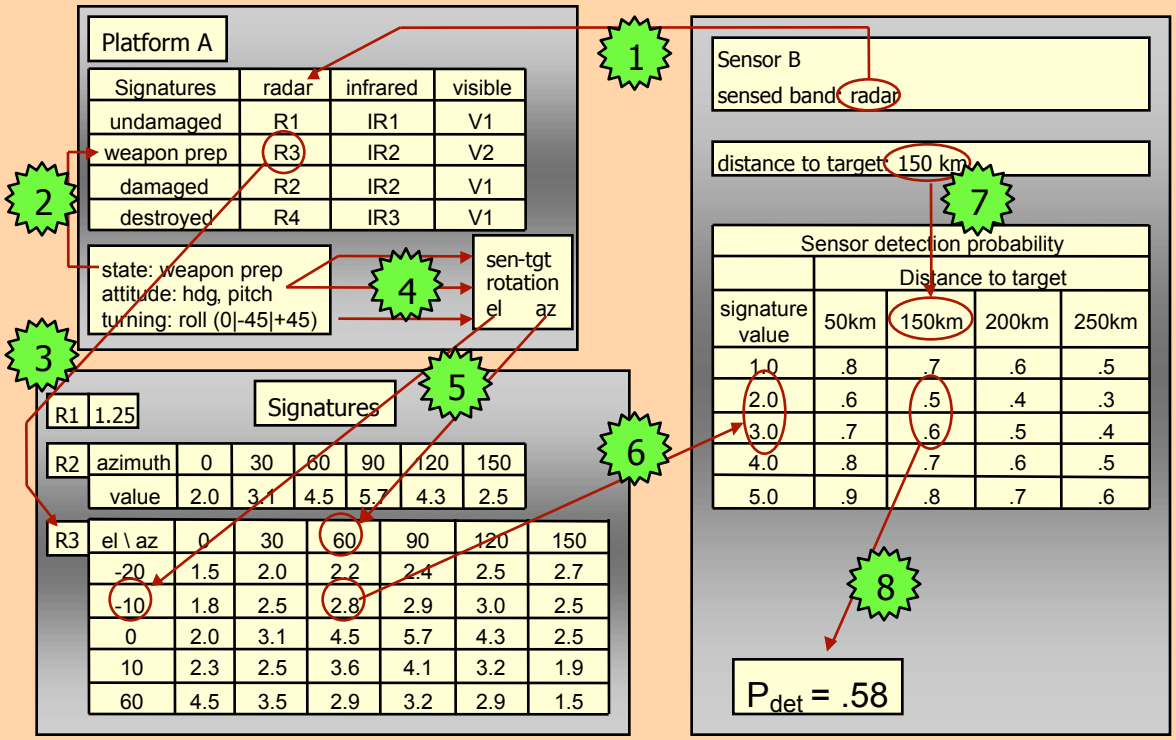
BOEING

- The probability of detection model is a new OEP feature that depends on
  - range to radars
  - aircraft configuration (weapon doors open or closed)
  - aircraft attitude
  - platform and sensor types
- Other model features and factors
  - ***lock loss*** (radar must detect for given number or sample periods or track is lost)
  - time for weapon strike vs. retreat time

The newest MICA control design challenge is to accomplish the mission in the presence of radars which can detect and shoot at the aircraft.

Detection Model

3D Signature

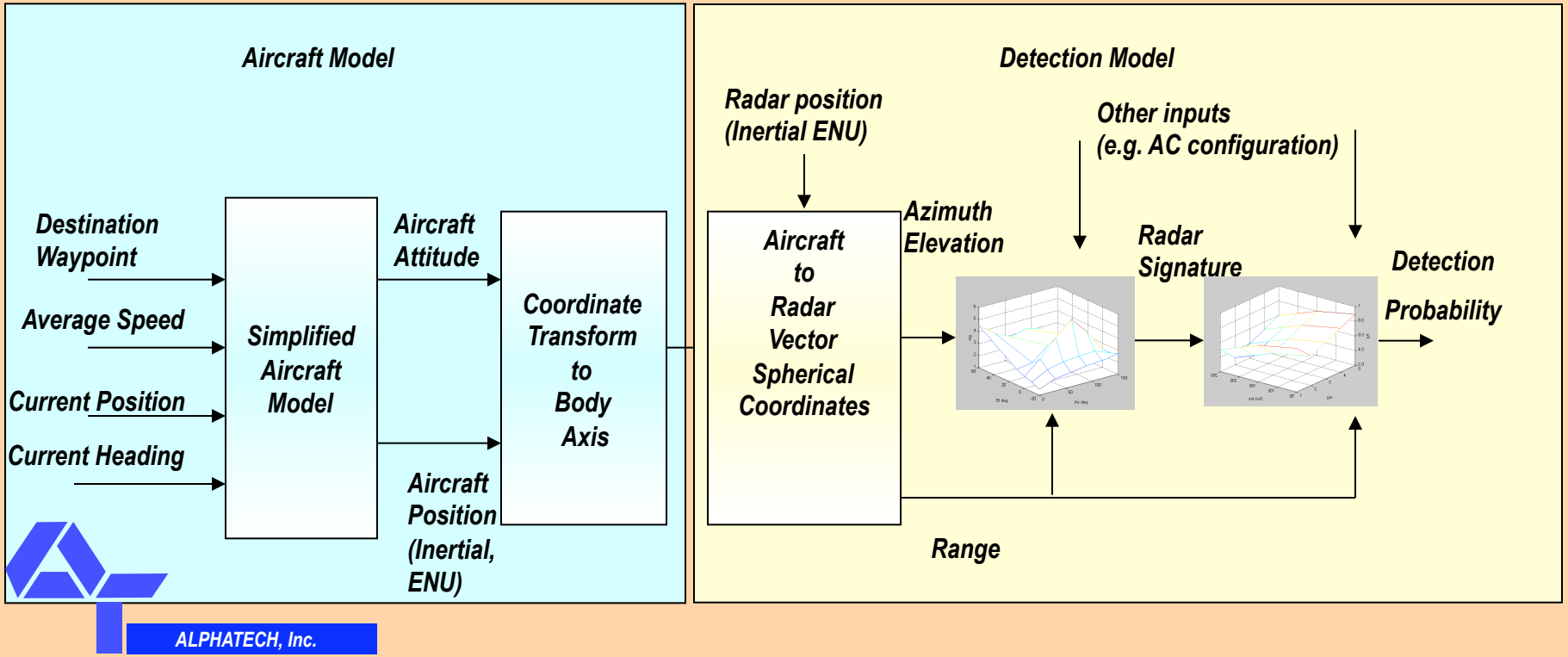


The steps to compute probability of detection:

1. Identify sensor type
2. Identify platform and configuration
3. Identify which signature table to use
4. Compute elevation.
5. Compute azimuth
6. Get signature from table
7. Compute range
8. Get prob. of detection from table

The probability of detection model depends on azimuth, elevation and range, as well as aircraft configuration. These models are based on table look-ups.

The Aircraft and Detection Models



This block diagram shows how two major components of the OEP model-the aircraft and detection models- are connected.

*This research is based upon work supported by the United States Air Force Research Laboratory under Contract No. F33615-01-C-3149

Solution with NTG → Need Analytical Models for the Detection Tables

- Need an analytical model for the signature/probability of detection tables
  - Use ***B-Spline Tensor product functions***

el/az	0	+/-30	+/-31	+/-180
0	1.5	1.5	5.5	5.5
+/-20	2.5	2.5	5.5	5.5
+/-45	3.5	3.5	6.0	6.0
+/-90	6.5	6.5	6.5	6.5
el/az	0	+/-30	+/-31	+/-180
0	2.0	2.0	6.5	6.5
+/-20	3.0	3.0	6.5	6.5
+/-45	4.0	4.0	6.5	6.5
+/-90	6.5	6.5	6.5	6.5

BOEING

Signature table for a small UAV, long SAM side

$$
\begin{aligned}
sig &= f(el, az) \\
el &= f_{el}(\psi(\dot{x}, \dot{y}), x, y) \\
az &= f_{az}(\psi(\dot{x}, \dot{y}), x, y)
\end{aligned}
$$

Intro to B-Splines

- Polynomials are suitable for function approximations because they can be evaluated, differentiated, and integrated easily.

- Limitation of Polynomial Approximation:
  - approximating polynomial may be too large
  - global dependence on local properties

- Solution:
  - subdivide the interval of approximation into small segments → *breaks*
  - use *piecewise* polynomial functions

- The polynomial pieces can be blend smoothly so that the resulting (patched) function can have several continuous derivatives

- Any such smooth piecewice polynomial function is called a spline

B-Spline Curves:

- **B-Splines:** basic splines (in Schoenberg's terminology)
- **B-Splines are useful for their:**
  - higher degree of freedom for curve design
  - ease of enforcing continuity between knot points
  - more control flexibility than Bezier curves
  - local modification property
  - ease of calculating their derivatives
- ***References***:
  - A Practical Guide to Splines, Revised Edition, Carl de Boor, 2001.
  - http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/B-spline (Dr. Ching-Kuang Shene, Dept. of Comp. Science, Michigan Technology Unv.)
  - MATLAB Spline Toolbox, User's Guide version2, Carl de Boor, 1999

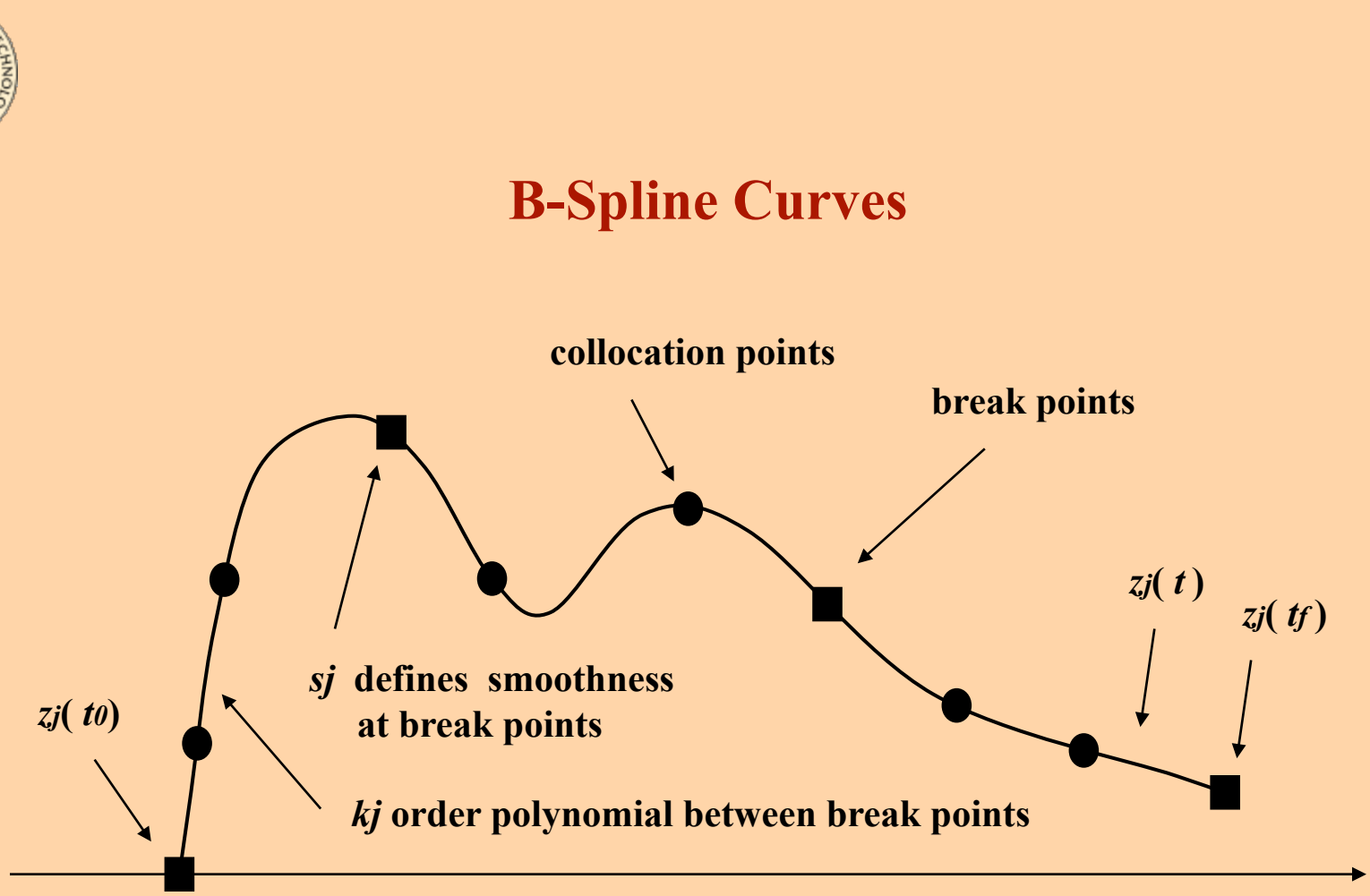
B-Spline Curves

- 1-D functions are defined with B-Splines as:

$$z_j(x) = \sum_{i=1}^{P_j} B_{i,k_j} C_i^j$$

$$p_j = l_j(k_j - s_j) + s_j$$

B_{i,k_j}	:	*B-Spline Basis functions*
C_i^j	:	*the coefficients of the B-splines (control points)*
k_j	:	*order of spline polynomial for output* z_j
l_j	:	*number of intervals*
s_j	:	*number of smoothness conditions at the break points*
p_j	:	*number of coefficients*



B-Spline Curves

B-Spline Basis Functions

- First order Basis Functions:

$$B_{i,1}(x) = N_{i,0}(x) \;=\; \begin{cases} 1 & : \; \textbf{\textit{if}} \quad t_i \le x \le t_{i+1} \\ 0 & : \; \textbf{\textit{otherwise}} \end{cases}$$

$$\textbf{\textit{if}} \quad t_i = t_{i+1} \rightarrow B_{i,1} = 0$$

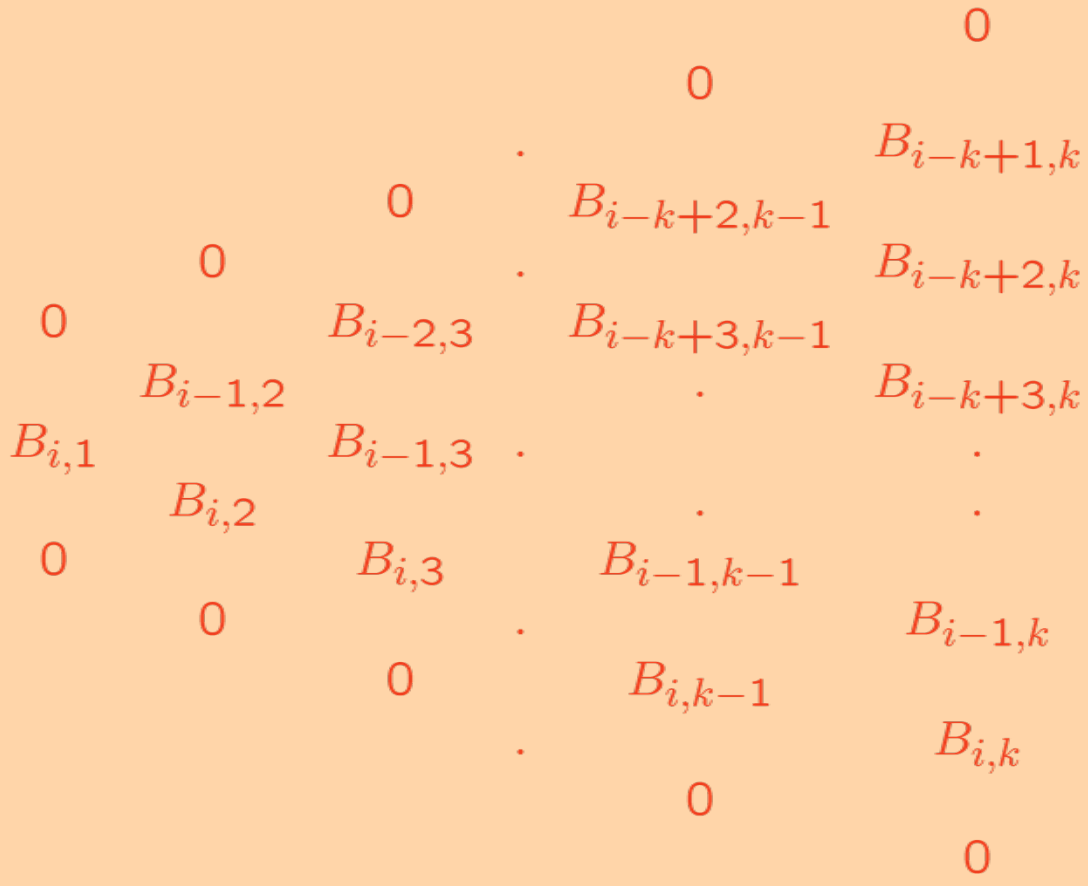
- Higher orders, *Cox-de Boor recursion formula*:

$$B_{i,k}(x) \;=\; \frac{x - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(x) + \frac{t_{i+k} - x}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(x)$$

$t[t_1, t_2, \cdots, t_{p+k}]$: knot points vector (non decreasing)

- knot multiplicities control the smoothness of the spline

knot multiplicty (m_j) + *smoothness condition* (s_j) = *order* (k_j)

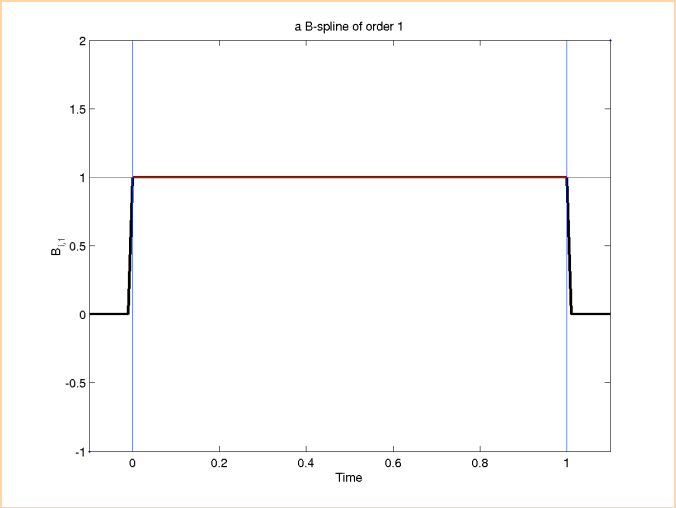


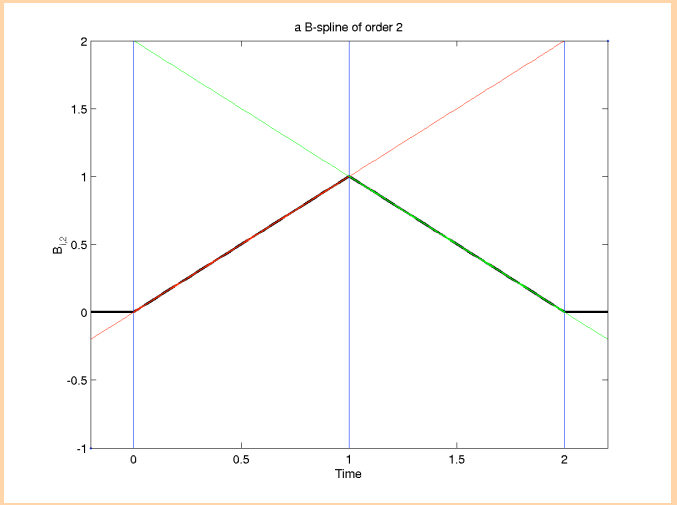
The Recurrence Relation

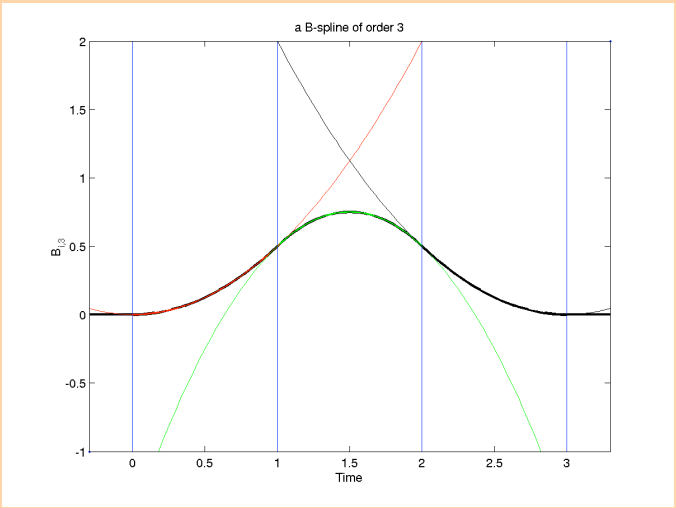
$$
\begin{array}{ccccc}
 & & & & 0 \\
 & & & 0 & \\
 & & \cdot & & B_{i-k+1,k} \\
 & 0 & & B_{i-k+2,k-1} & \\
 0 & & \cdot & & B_{i-k+2,k} \\
 & B_{i-2,3} & & B_{i-k+3,k-1} & \\
B_{i-1,2} & & & \cdot & B_{i-k+3,k} \\
 & B_{i-1,3} & \cdot & & \cdot \\
B_{i,2} & & & \cdot & \cdot \\
 & B_{i,3} & & B_{i-1,k-1} & \\
 0 & & \cdot & & B_{i-1,k} \\
 & 0 & & B_{i,k-1} & \\
 & & \cdot & & B_{i,k} \\
 & & & 0 & \\
 & & & & 0
\end{array}
$$

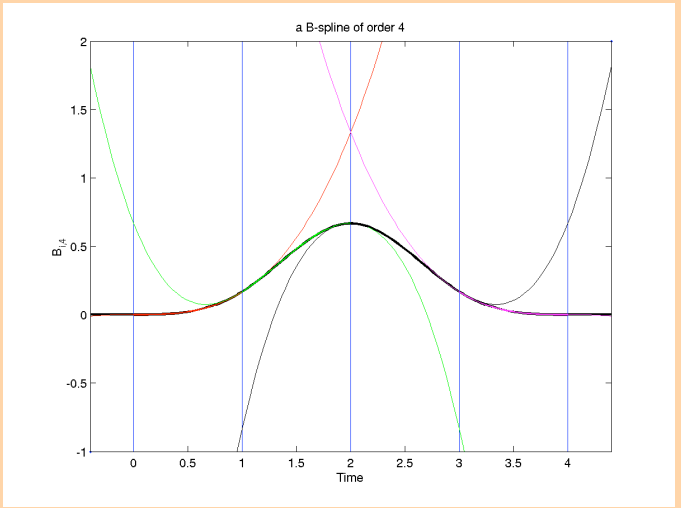
- The *triangular* array of B-Splines of order $\leq k$, nonzero on $[t_i, \cdots, t_{i+1}]$

Examples of B-Spline Curves





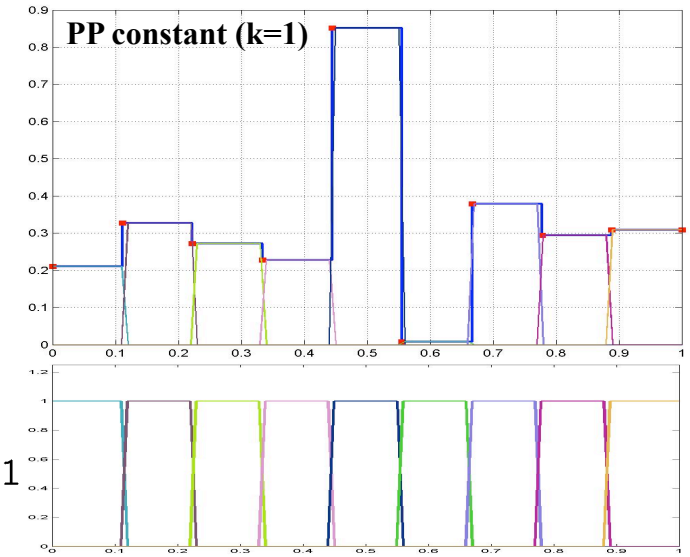




Basis functions and PP functions

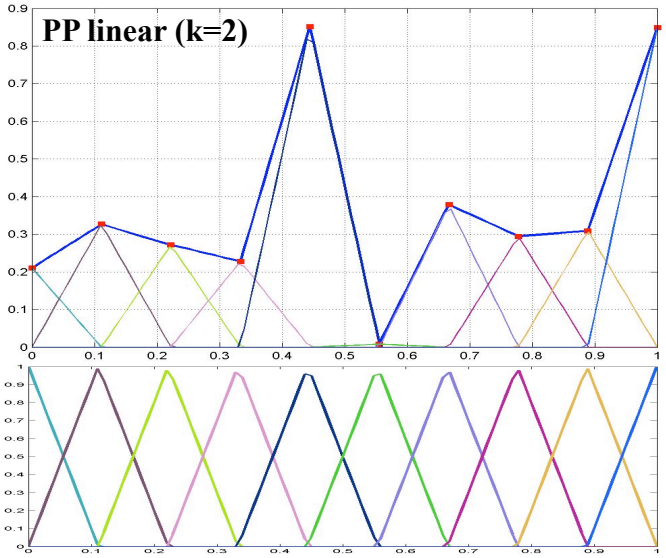
by Melvin E. Flores

PP constant (k=1)



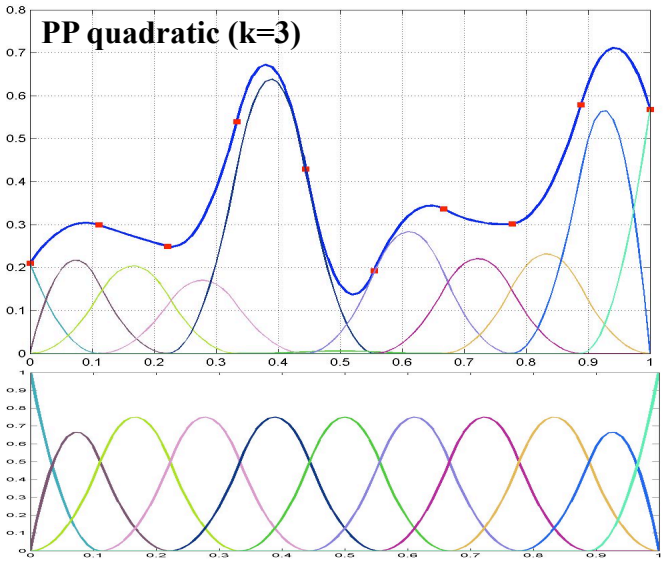
$B_{j,1}$

PP linear (k=2)



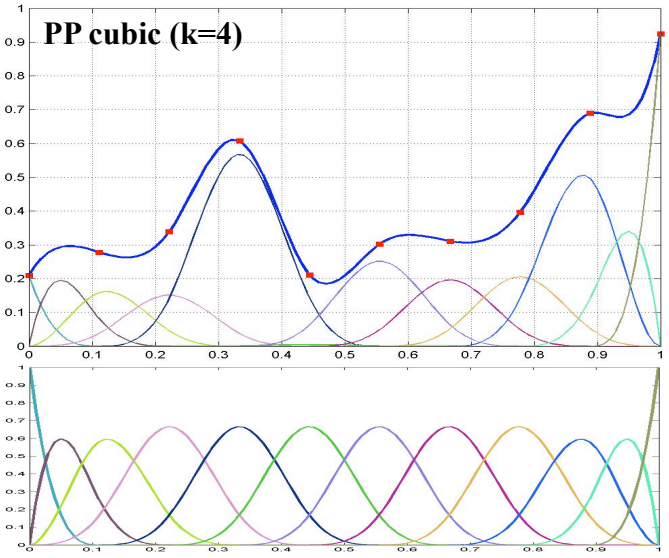
$B_{j,2}$

PP quadratic (k=3)



$B_{j,3}$

PP cubic (k=4)



$B_{j,4}$

Derivatives of B-Splines

- **The first derivative of a k^{th} order spline is a $(k-1)^{th}$ order spline:**
- **If we are interested in the fixed interval $[t_r \cdots t_s]$**

$$\sum_i B_{i,k} C_i = \sum_{i=r-k+1}^{s-1} B_{i,k} C_i \quad \textbf{on} \quad [t_r \cdots t_s]$$

$$D\left(\sum_i B_{i,k} C_i\right) = \sum_{i=r-k+2}^{s-1} (k-1) \frac{C_i - C_{i-1}}{t_{i+k-1} - t_i} B_{i,k-1} \quad \textbf{on} \quad [t_r \cdots t_s]$$

2-D Tensor Product B-Splines

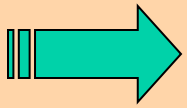
$$z(x,y) = \sum_{i=1}^{P_x} \sum_{j=1}^{P_y} B_{i,k_x}(x) B_{j,k_y}(y) A_{i,j}$$

$B_{i,k_x}(x)$	:	*B-Spline Basis functions for x-direction*
$A_{i,j}$	:	*Coefficient matrix,* $P_x \times P_y$
k_x	:	*order of spline polynomial for x-direction*
p_x	:	*number of coefficients for x-direction*

Using MATLAB's Spline Toolbox:

el/az	0	+/-0.5236	+/- 0.5411	3.1416
0	1.5	1.5	5.5	5.5
0.3491	2.5	2.5	5.5	5.5
0.7854	3.5	3.5	6.0	6.0
1.5708	6.5	6.5	6.5	6.5

Signature table for a small UAV, long SAM side



$$
\begin{aligned}
sig &= f(el, az) \\
el &= f_{el}(\psi(\dot{x}, \dot{y}), x, y) \\
az &= f_{az}(\psi(\dot{x}, \dot{y}), x, y)
\end{aligned}
$$

- `>> sgfit = spap2({3,5},[3,2],{elref,azref},sigref,{wel,waz});`

  `spap2` constructs a B-Spline that best approximates the data in the Least-Squares sense

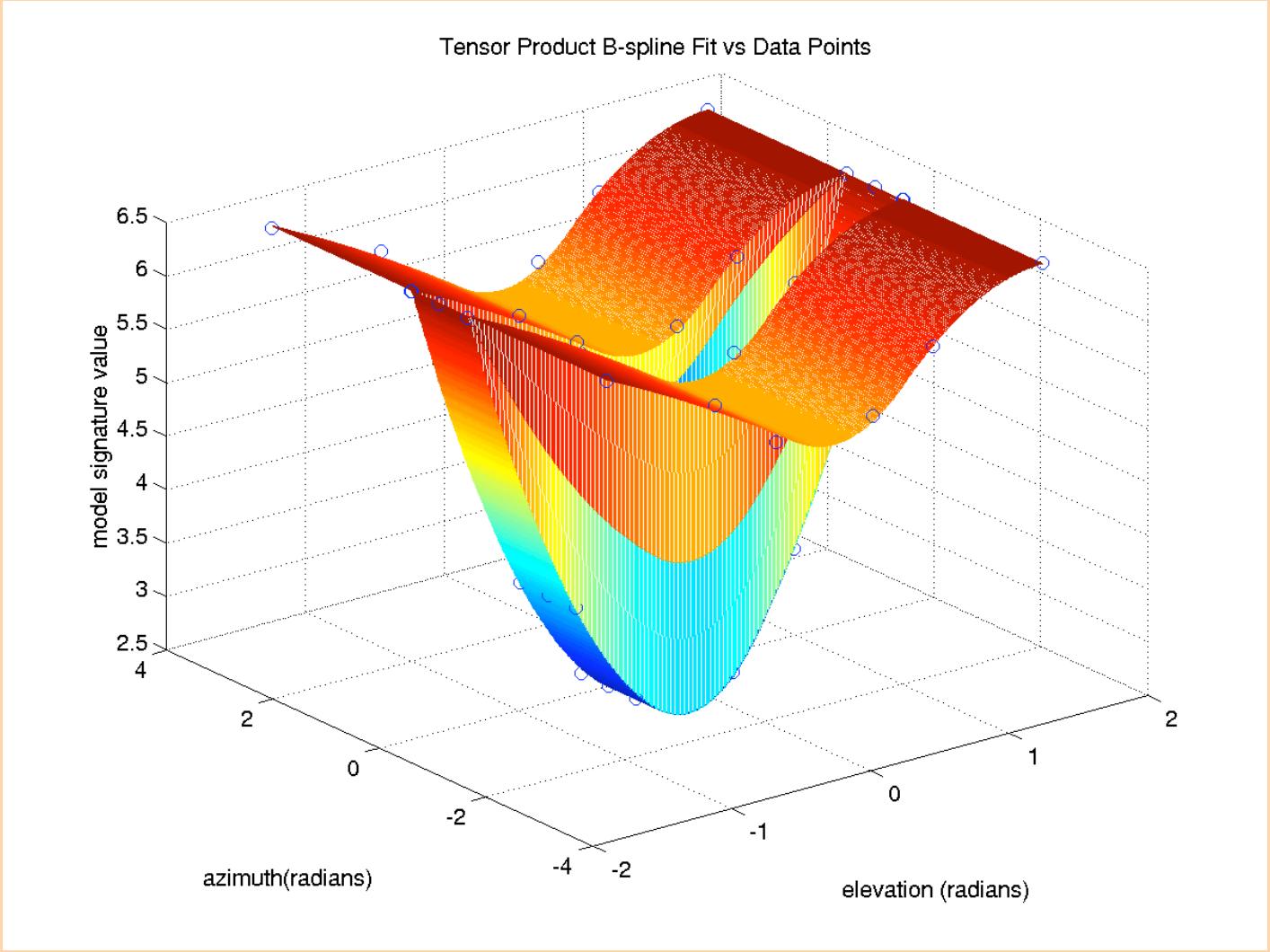
- `>> sgcheck = spval(sgfit,{elref,azref});`

  `spval` returns the value at {elref,azref} of the function whose B-form is in `sgfit`

- `>> elder = fnder(sgfit,[1,0]); azder = fnder(sgfit,[0,1])`

  `fnder` returns the (representation of the) first derivative

Analytical Model for the Signature Table



Using B-Spline Model in NTG

- Need to extract B-Spline info from **`sgfit`**
- **`>> [knots, coefs, n, k] = spbrk(sgfit);`**

  `spbrk` breaks the B-form in `sgfit into parts`

```
knots{1} = [-1.5708 -1.5708 -1.5708 -0.2182 0.5672 1.5708 1.5708 1.5708]

knots{2} = [-3.1416 -3.1416 -0.5411 -0.5236 0.5236 0.5411 3.1416 3.1416]

n(1) = 5
n(2) = 6      (# of coefficients)

k(1) = 3
k(2) = 2        (order of splines)

coefs → 5x6

n(1) + k(1) = length(t(1))
```

Using B-Spline Model in NTG

- Assume that we have a nonlinear constraint on signature:

  $1.5 \leq sig = f(el, az) \leq 4.5$

- Need to implement spline fit, **`sig = f(el,az)`**, in c-code, use equation:

  $z(x,y) = \sum_{i=1}^{P_x} \sum_{j=1}^{P_y} B_{i,k_x}(x) B_{j,k_y}(y) A_{i,j}$ *and*

- Derivatives of the signature respect to **`(el,az)`**, use equation:

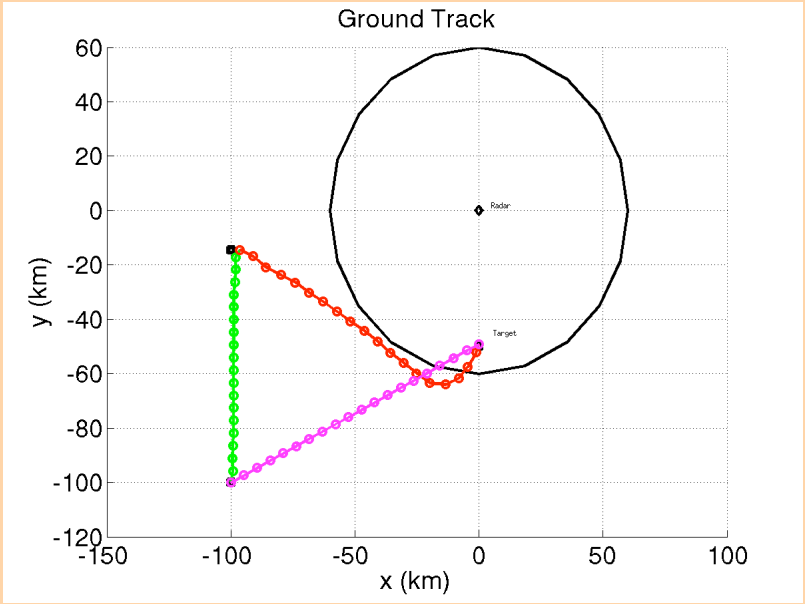
  $$\left(\frac{dsig}{del}, \frac{dsig}{daz}\right) \longrightarrow D\left(\sum_i B_{i,k} C_i\right) = \sum_{i=r-k+2}^{s-1} (k-1) \frac{C_i - C_{i-1}}{t_{i+k-1} - t_i} B_{i,k-1} \quad \textbf{on } [t_r \cdots t_s]$$

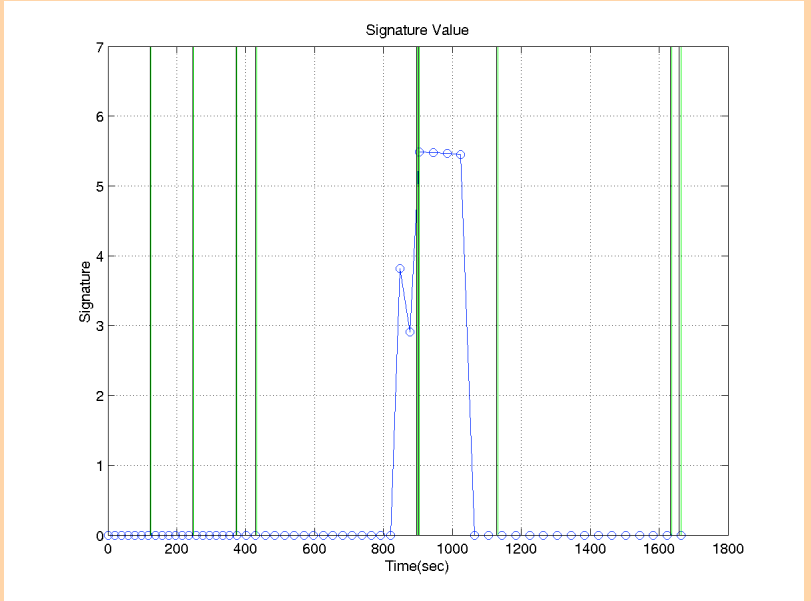
- Derivatives respect to NTG outputs?

$$\left.\begin{aligned} el &= f_{el}(\psi(\dot{x}, \dot{y}), x, y) \\ az &= f_{az}(\psi(\dot{x}, \dot{y}), x, y) \end{aligned}\right\} \Rightarrow$$

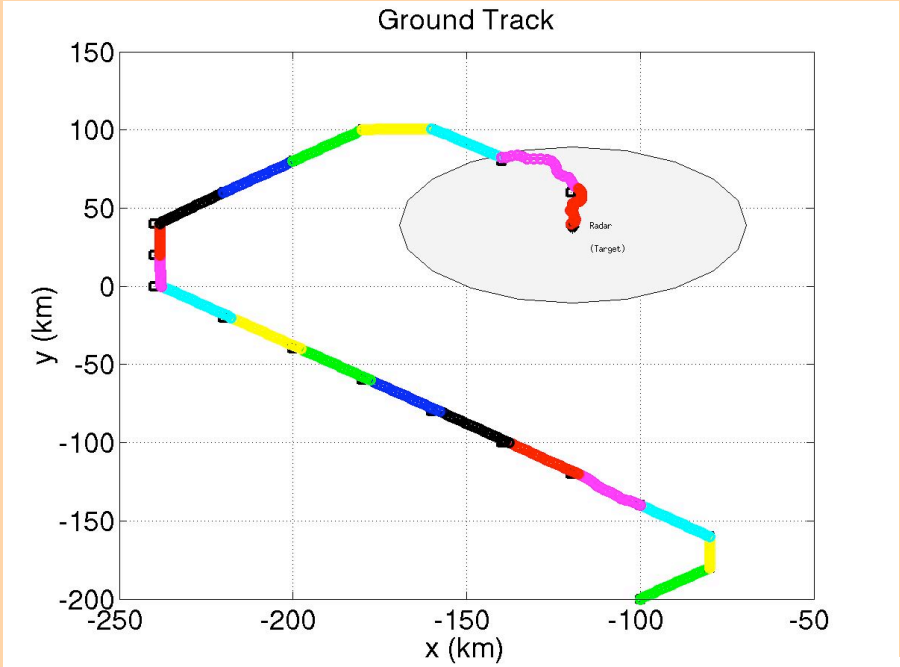
$$\begin{aligned} \frac{dsig}{dx} &= \frac{dsig}{del} \cdot \frac{del}{dx} + \frac{dsig}{daz} \cdot \frac{daz}{dx} \\ &\vdots \quad \vdots \qquad\qquad \vdots \\ \frac{dsig}{d\dot{y}} &= \frac{dsig}{del} \cdot \frac{del}{d\dot{y}} + \frac{dsig}{daz} \cdot \frac{daz}{d\dot{y}} \end{aligned}$$

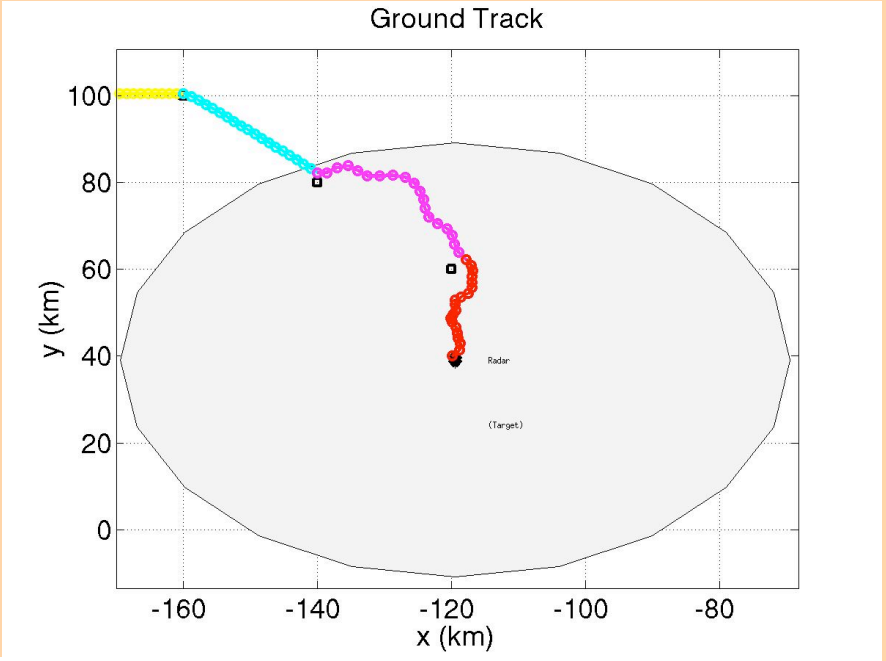
NTG Temporal LO Examples

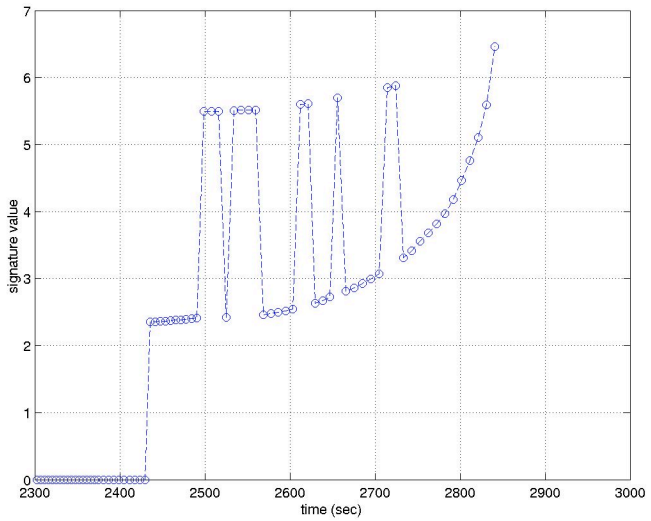




NTG Temporal LO Examples







Thank you for your attention.