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

<u>Note:</u> Picture has been omitted to save space

A New OEP Feature

OEP: A common battleground developed by Boeing.

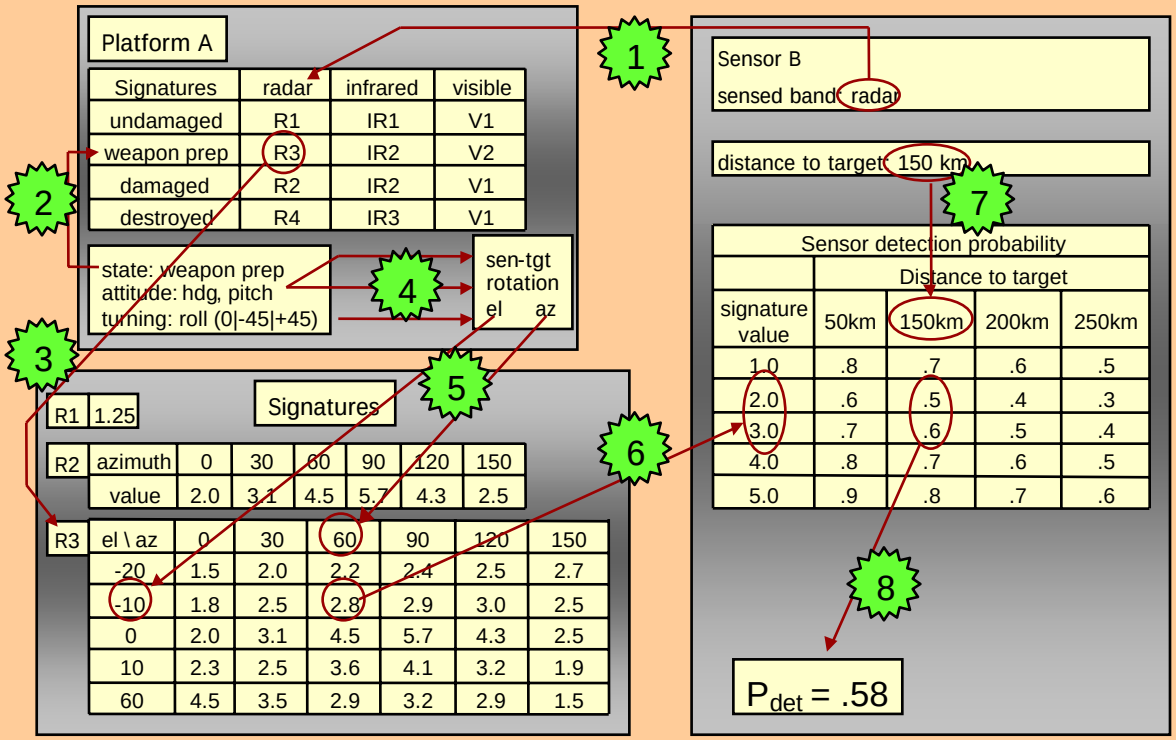
<u>Note:</u> Picture has been omitted to save space

- 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

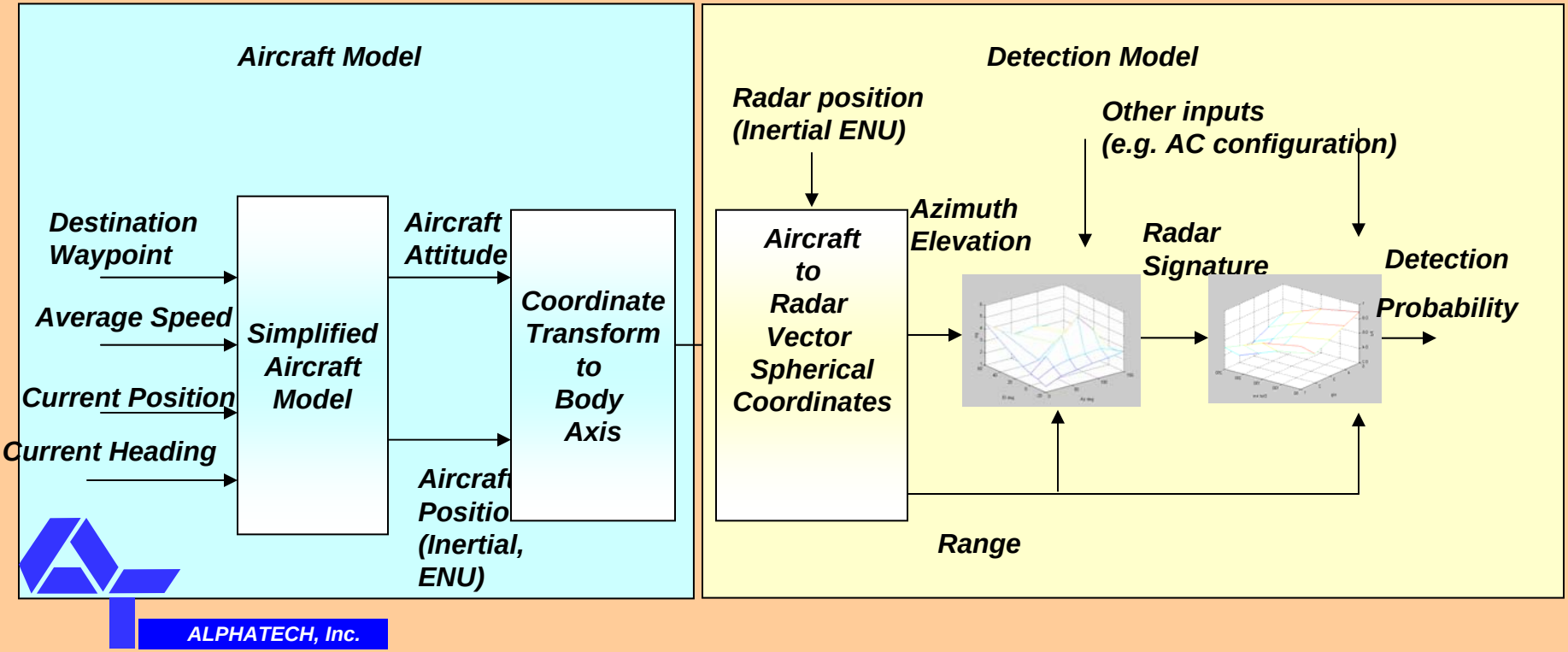


The steps to compute probability of detection:

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

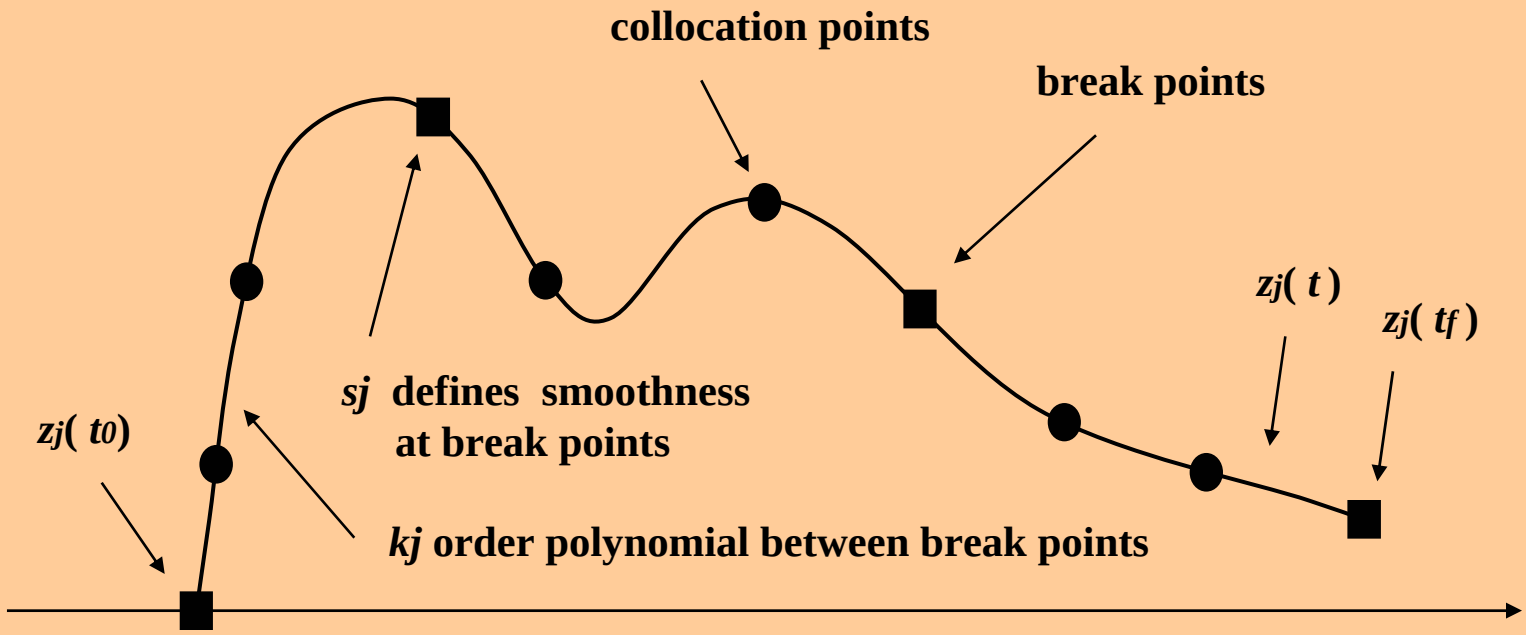
- 1-D functions are defined with B-Splines as:

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

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

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:

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

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

- 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

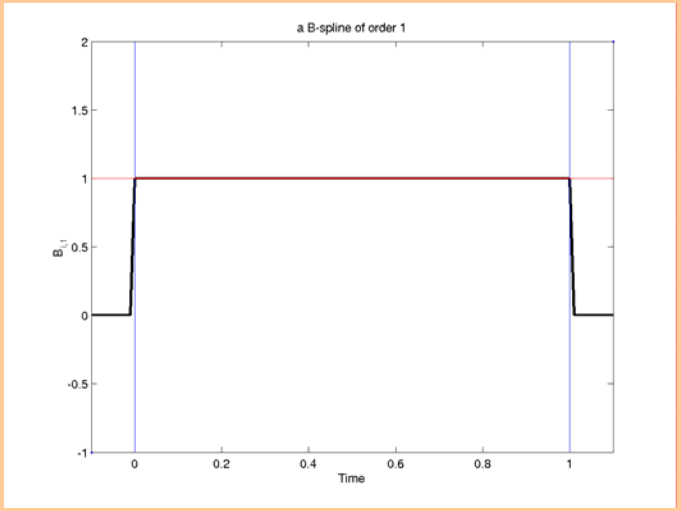
knot multiplicty (m_j) + *smoothness condition* (s_j) = *order* (k_j)

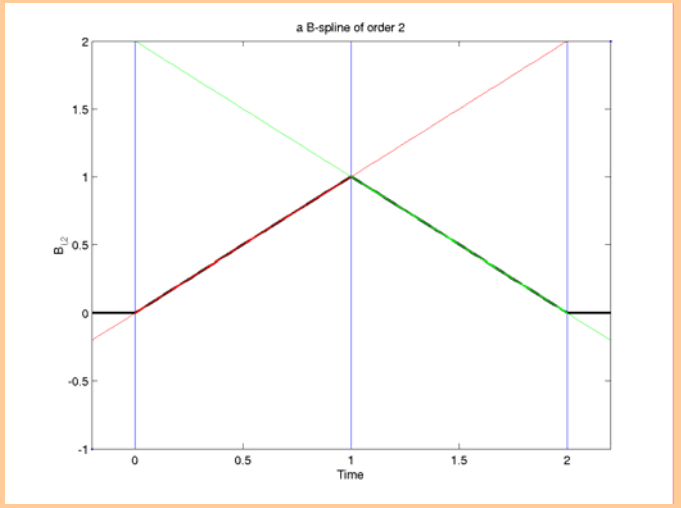
The Recurrence Relation

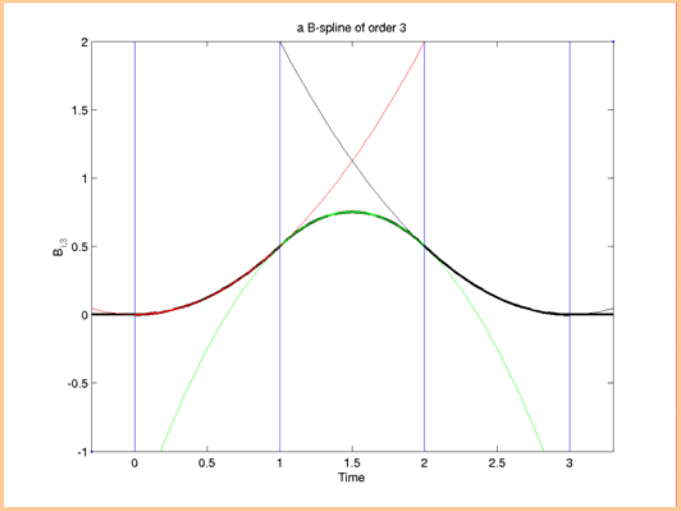
$$
\begin{array}{ccccc}
 & & & & 0 \\
 & & & 0 & \\
 & & . & & B_{i-k+1,k} \\
 & 0 & & B_{i-k+2,k-1} & \\
 0 & & . & & B_{i-k+2,k} \\
 & B_{i-2,3} & & B_{i-k+3,k-1} & \\
B_{i-1,2} & & . & & B_{i-k+3,k} \\
 & B_{i-1,3} & . & & . \\
B_{i,2} & & & . & . \\
 & B_{i,3} & & B_{i-1,k-1} & \\
0 & & . & & B_{i-1,k}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

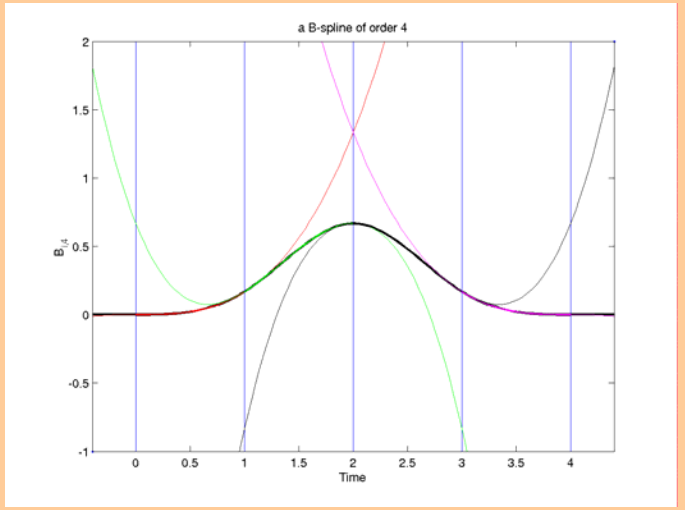
- The *triangular* array of B-Splines of order $\leq k$, nonzero on $[t_i, \cdots, t_{i+1}]$

Examples of B-Spline Curves





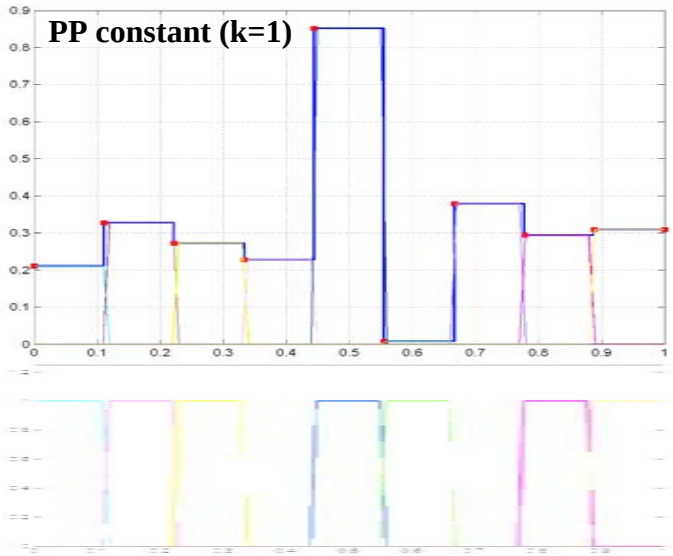




Basis functions and PP functions

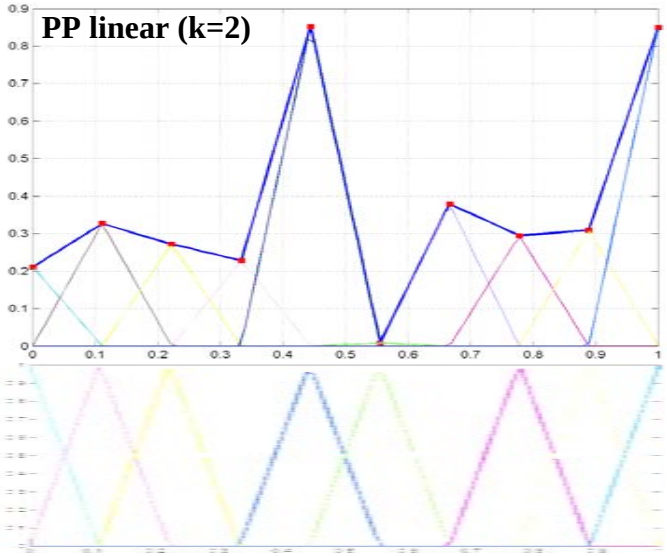
by Melvin E. Flores

PP constant (k=1)



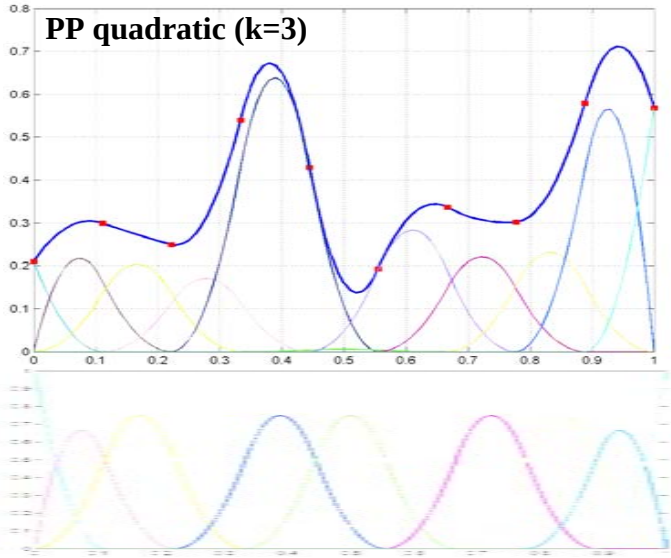
$B_{j,1}$

PP linear (k=2)



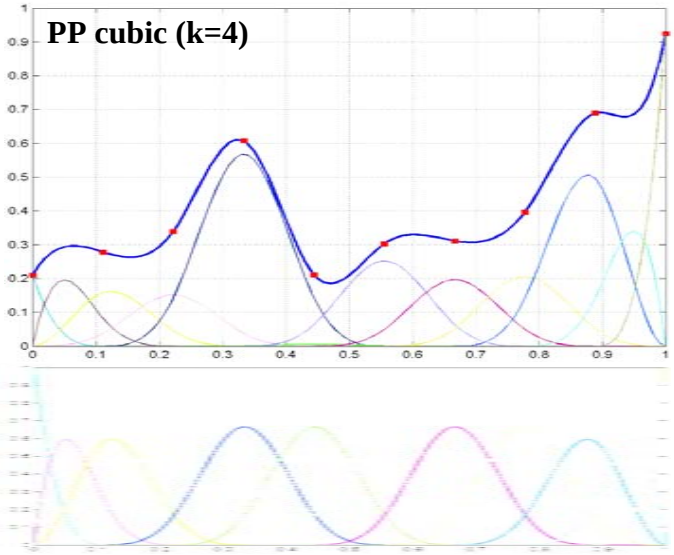
$B_{j,2}$

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 $[t_r \cdots t_s]$**

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

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

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

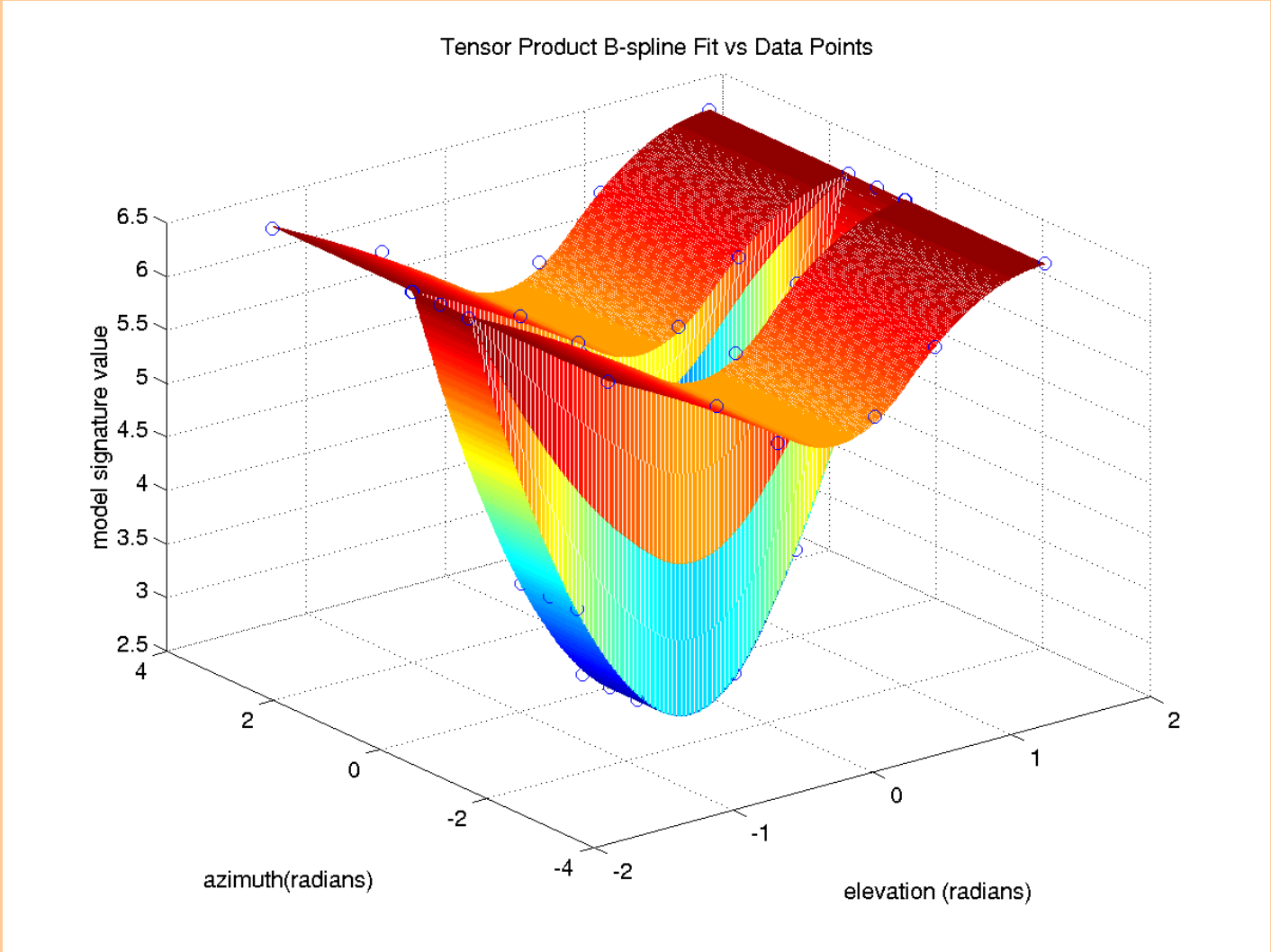
- **`>> sgcheck = spval(sgfit,{elref,azref});`**

  `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:

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

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

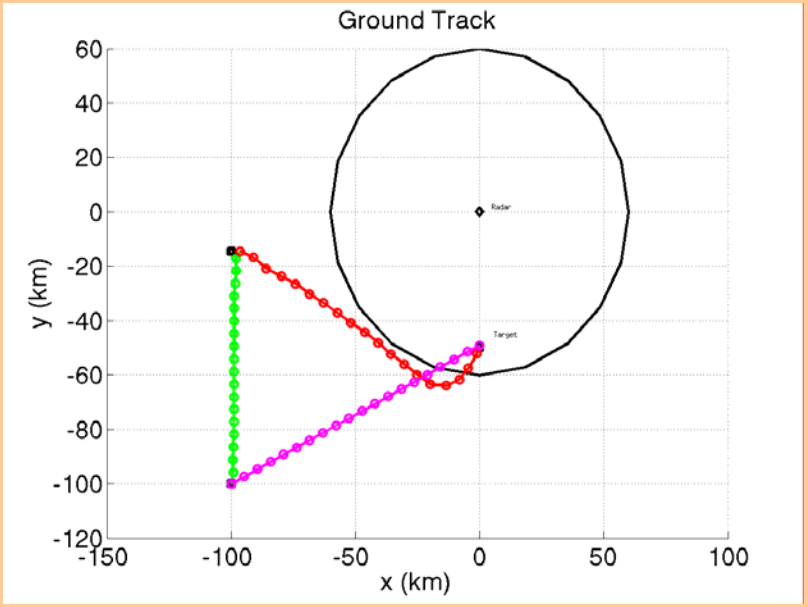
$$\left(\frac{dsig}{del}, \frac{dsig}{daz}\right) \longrightarrow D\left(\sum_i B_{i,k} C_i\right) = \sum_{i=r-k+2}^{s-1} (k-1) \frac{C_i - C_{i-1}}{t_{i+k-1} - t_i} B_{i,k-1} \quad \textit{on } [t_r \cdots t_s]$$

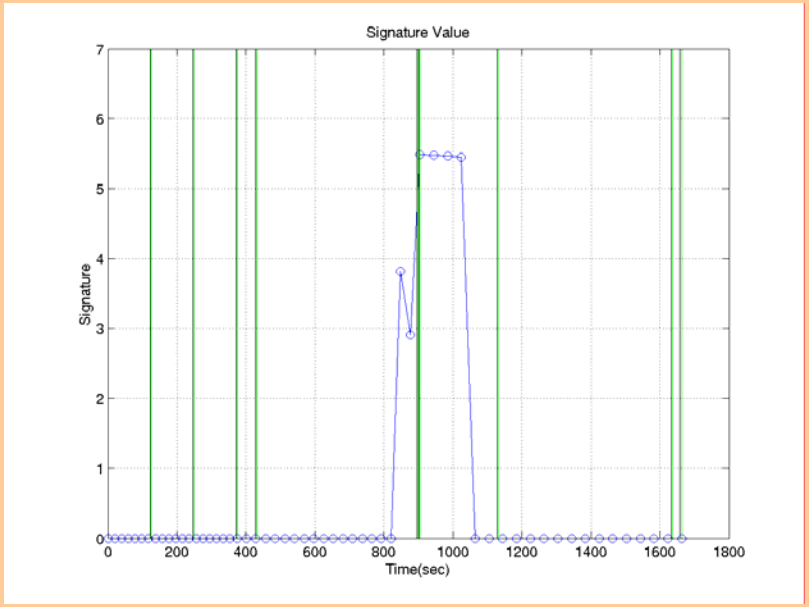
- Derivatives respect to NTG outputs?

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

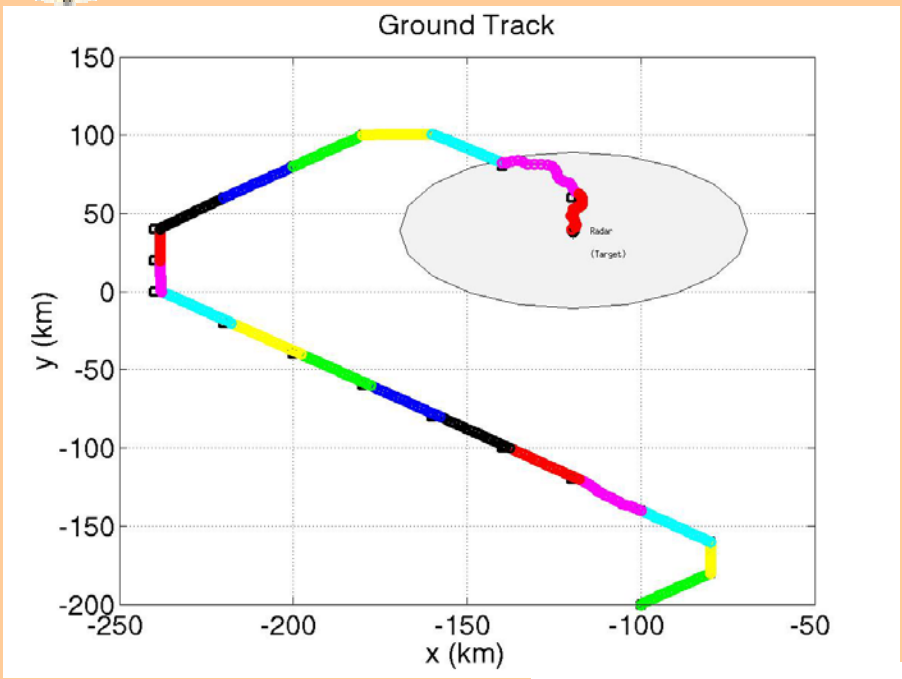
$$\begin{aligned} \frac{dsig}{dx} &= \frac{dsig}{del} \cdot \frac{del}{dx} + \frac{dsig}{daz} \cdot \frac{daz}{dx} \\ &\vdots \quad \vdots \qquad\qquad \vdots \\ \frac{dsig}{d\dot{y}} &= \frac{dsig}{del} \cdot \frac{del}{d\dot{y}} + \frac{dsig}{daz} \cdot \frac{daz}{d\dot{y}} \end{aligned}$$

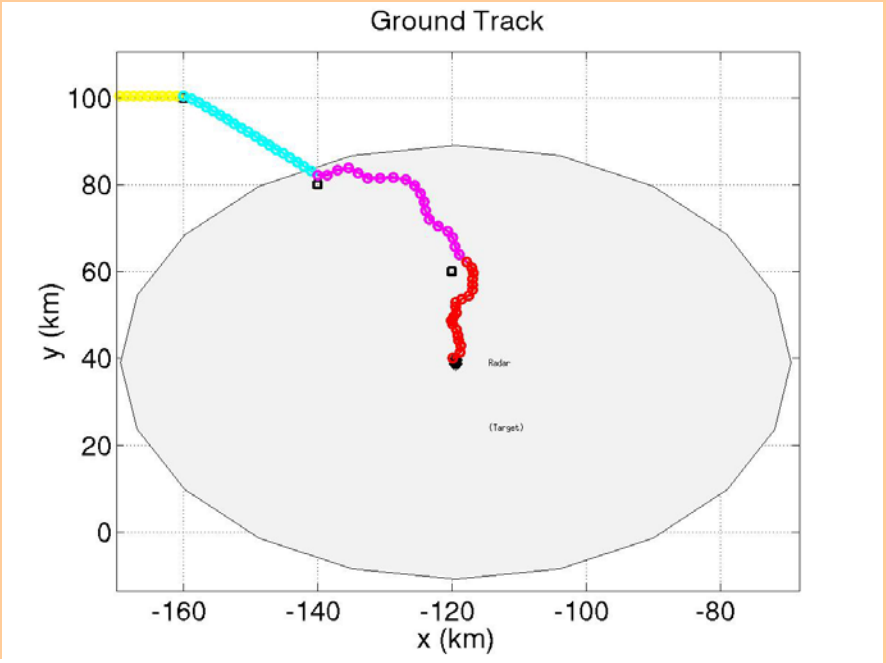
NTG Temporal LO Examples

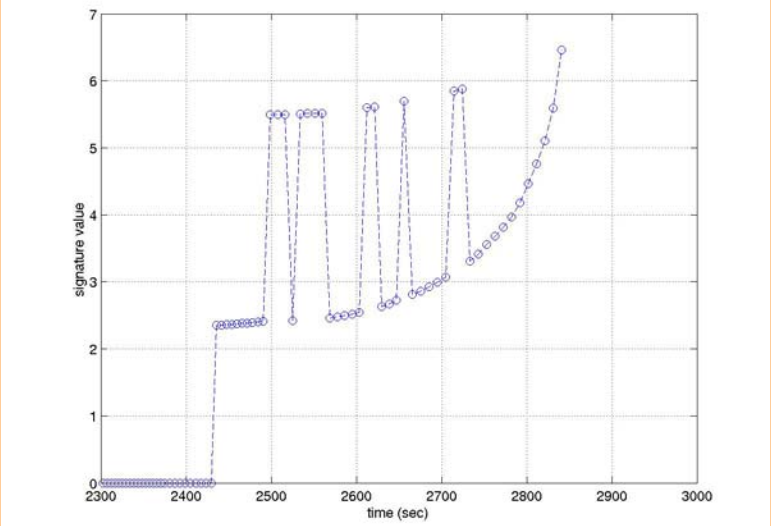




NTG Temporal LO Examples







Thank you for your attention.