Introduction to the Optimal Control Software $\mathbb{GPOPS}-\mathbb{II}$

Anil V. Rao

Department of Mechanical and Aerospace Engineering

University of Florida

Gainesville, FL 32611-6250

Tutorial on $\mathbb{GPOPS}-\mathbb{II}$

NSF CBMS Workshop

23 July 2018

Overview of Presentation

- Mathematical Problem Being Solved by $\mathbb{GPOPS}-\mathbb{II}$
  - ▷ Continuous optimal control problem
  - ▷ Gaussian quadrature approximation of optimal control problem
  - ▷ Structure of NLP arising from Gauss quadrature approximation
- Layout of $\mathbb{GPOPS}-\mathbb{II}$ Software
  - ▷ Main driver function
    - ◦ Specification of bounds on variables and constraints
    - ◦ Providing an initial guess
  - ▷ Form and syntax of user-defined continuous and endpoint functions
- Use of Various Options in $\mathbb{GPOPS}-\mathbb{II}$
  - ▷ Choice of NLP solvers
  - ▷ NLP solver options
  - ▷ Derivative options

Optimal Control Problem Solved by $\mathbb{GPOPS}-\mathbb{II}$

$$J = \phi\left(\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(P)}, \mathbf{s}\right),$$

$$\dot{\mathbf{y}}^{(p)} = \mathbf{a}^{(p)}(\mathbf{y}^{(p)}, \mathbf{u}^{(p)}, t^{(p)}, \mathbf{s}), \quad (p = 1, \ldots, P),$$

$$\mathbf{b}_{\min} \leq \mathbf{b}\left(\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(P)}, \mathbf{s}\right) \leq \mathbf{b}_{\max},$$

$$\mathbf{e}^{(p)} = \left[\mathbf{y}^{(p)}(t_0^{(p)}), t_0^{(p)}, \mathbf{y}^{(p)}(t_f^{(p)}), t_f^{(p)}, \mathbf{q}^{(p)}\right], \ (p = 1, \ldots, P)$$

$$\mathbf{c}_{\min}^{(p)} \leq \mathbf{c}^{(p)}(\mathbf{y}^{(p)}, \mathbf{u}^{(p)}, t^{(p)}, \mathbf{s}) \leq \mathbf{c}_{\max}^{(p)}, \quad (p = 1, \ldots, P),$$

$$\mathbf{s}_{\min} \leq \mathbf{s} \leq \mathbf{s}_{\max},$$

$$\mathbf{q}_{\min}^{(p)} \leq \mathbf{q}^{(p)} \leq \mathbf{q}_{\max}^{(p)}, \quad (p = 1, \ldots, P),$$

$$q_i^{(p)} = \int_{t_0^{(p)}}^{t_f^{(p)}} g_i^{(p)}(\mathbf{y}^{(p)}, \mathbf{u}^{(p)}, t^{(p)}, \mathbf{s})dt, \quad (i = 1, \ldots n_q^{(p)}), (p = 1, \ldots, P).$$

- Form of Cost Functional
  - ▷ Written purely in Mayer Form
  - ▷ Integrals are computed separately and treated as variables
  - ▷ Allows for more general formulation

- Generality of Constraints
  - ▷ Event constraints utilize information from endpoints of any phase
  - ▷ Can place constraints directly on integrals
  - ▷ Path constraints can be a function of state, control, static parameters, and time

Radau Collocation for a One-Phase Optimal Control Problem

- State Approximation Using Lagrange Polynomials

$$\mathbf{y}^{(k)}(\tau) \approx \mathbf{Y}^{(k)}(\tau) = \sum_{j=1}^{N_k+1} \mathbf{Y}_j^{(k)} \ell_j^{(k)}(\tau), \quad \ell_j^{(k)}(\tau) = \prod_{\substack{l=1 \\ l \neq j}}^{N_k+1} \frac{\tau - \tau_l^{(k)}}{\tau_j^{(k)} - \tau_l^{(k)}},$$

- Derivative of State Approximation

$$\frac{d\mathbf{Y}^{(k)}(\tau)}{d\tau} = \sum_{j=1}^{N_k+1} \mathbf{Y}_j^{(k)} \frac{d\ell_j^{(k)}(\tau)}{d\tau}.$$

- Approximation of Cost Functional

$$\mathcal{J} = \phi(\mathbf{Y}_1^{(1)}, t_0, \mathbf{Y}_{N_K+1}^{(K)}, t_f, \mathbf{q}),$$

- Discretized Dynamic Constraints

$$\sum_{j=1}^{N_k+1} D_{ij}^{(k)} \mathbf{Y}_j^{(k)} - \frac{t_f - t_0}{2} \mathbf{a}(\mathbf{Y}_i^{(k)}, \mathbf{U}_i^{(k)}, \tau_i^{(k)}; t_0, t_f) = \mathbf{0}, \quad (i = 1, \ldots, N_k).$$

$$D_{ij}^{(k)} = \left[\frac{d\ell_j^{(k)}(\tau)}{d\tau}\right]_{\tau_i^{(k)}}, \quad (i = 1, \ldots, N_k, \quad j = 1, \ldots, N_k+1, \quad k = 1, \ldots, K),$$

- Equivalent Integral Form of Dynamic Constraints

$$\mathbf{Y}_{i+1}^{(k)} - \mathbf{Y}_1^{(k)} - \frac{t_f - t_0}{2} \sum_{j=1}^{N_k} I_{ij}^{(k)} \mathbf{a}(\mathbf{Y}_i^{(k)}, \mathbf{U}_i^{(k)}, \tau_i^{(k)}; t_0, t_f) = \mathbf{0}, \quad (i = 1, \ldots, N_k),$$

$$\mathbf{I}^{(k)} \equiv \left[\mathbf{D}_{2:N_k+1}^{(k)}\right]^{-1},$$

- Discretized Path Constraints

$$\mathbf{c}_{\min} \leq \mathbf{c}(\mathbf{Y}_i^{(k)}, \mathbf{U}_i^{(k)}, \tau_i^{(k)}; t_0, t_f) \leq \mathbf{c}_{\max}, \quad (i = 1, \ldots, N_k),$$

- Discretized Integrals

$$q_j \approx \sum_{k=1}^{K} \sum_{i=1}^{N_k} \frac{t_f - t_0}{2} w_i^{(k)} g_j(\mathbf{Y}_i^{(k)}, \mathbf{U}_i^{(k)}, \tau_i^{(k)}; t_0, t_f), \quad (i = 1, \ldots, N_k, j = 1, \ldots, n_q$$

- Discretized Boundary Conditions

$$\mathbf{b}_{\min} \leq \mathbf{b}(\mathbf{Y}_1^{(1)}, t_0, \mathbf{Y}_{N_K+1}^{(K)}, t_f, \mathbf{q}) \leq \mathbf{b}_{\max}.$$

$$\mathbf{Y}_{N_k+1}^{(k)} = \mathbf{Y}_1^{(k+1)}, \quad (k = 1, \ldots, K-1),$$

NLP Arising from Multiple-Phase Radau Collocation

- Objective Function

$$\Phi(\mathbf{Z})$$

- Constraints

$$\mathbf{F}_{\min} \leq \mathbf{F}(\mathbf{Z}) \leq \mathbf{F}_{\max}.$$

- Variable Bounds

$$\mathbf{Z}_{\min} \leq \mathbf{Z} \leq \mathbf{Z}_{\max}.$$

- Decision Vector

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}^{(1)} \\ \vdots \\ \mathbf{z}^{(P)} \\ s_1 \\ \vdots \\ s_{n_s} \end{bmatrix}, \quad \mathbf{z}^{(p)} = \begin{bmatrix} \mathbf{V}_1^{(p)} \\ \vdots \\ \mathbf{V}_{n_y^{(p)}}^{(p)} \\ \mathbf{W}_1^{(p)} \\ \vdots \\ \mathbf{W}_{n_u^{(p)}}^{(p)} \\ \mathbf{q}^{(p)} \\ t_0^{(p)} \\ t_f^{(p)} \end{bmatrix}, \quad (p = 1, \ldots, P),$$

$$\mathbf{V}^{(p)} = \begin{bmatrix} \mathbf{Y}_1^{(p)} \\ \vdots \\ \mathbf{Y}_{N^{(p)}+1}^{(p)} \end{bmatrix} \in \mathbb{R}^{(N^{(p)}+1) \times n_y^{(p)}}, \quad \mathbf{W}^{(p)} = \begin{bmatrix} \mathbf{U}_1^{(p)} \\ \vdots \\ \mathbf{U}_{N^{(p)}}^{(p)} \end{bmatrix} \in \mathbb{R}^{N^{(p)} \times n_u^{(p)}},$$

- NLP Cost Function in Terms of Optimal Control Cost

$$\Phi(\mathbf{Z}) = \phi$$

- NLP Constraints

$$\mathbf{F} = \begin{bmatrix} \mathbf{f}^{(1)} \\ \vdots \\ \mathbf{f}^{(P)} \\ \mathbf{b} \end{bmatrix},$$

$$\mathbf{f}^{(p)} = \begin{bmatrix} \boldsymbol{\Delta}_1^{(p)} \\ \vdots \\ \boldsymbol{\Delta}_{n_y^{(p)}}^{(p)} \\ \mathbf{C}_1^{(p)} \\ \vdots \\ \mathbf{C}_{n_c^{(p)}}^{(p)} \\ \boldsymbol{\rho}^{(p)} \end{bmatrix}, \quad (p = 1, \ldots, P),$$

- Defect Constraint Matrix (Differential Form)

$$\boldsymbol{\Delta}^{(p)} = \mathbf{D}^{(p)}\mathbf{Y}^{(p)} - \frac{t_f^{(p)} - t_0^{(p)}}{2}\mathbf{A}^{(p)} \in \mathbb{R}^{N^{(p)} \times n_y^{(p)}}.$$

- Defect Constraint Matrix (Integral Form)

$$\boldsymbol{\Delta}^{(p)} = \mathbf{E}^{(p)}\mathbf{Y}^{(p)} - \frac{t_f^{(p)} - t_0^{(p)}}{2}\mathbf{I}^{(p)}\mathbf{A}^{(p)} \in \mathbb{R}^{N^{(p)} \times n_y^{(p)}},$$

$$\mathbf{A}^{(p)} = \begin{bmatrix} \mathbf{a}(\mathbf{Y}_1^{(p)}, \mathbf{U}_1^{(p)}, t_1^{(p)}, \mathbf{s}) \\ \vdots \\ \mathbf{a}(\mathbf{Y}_{N^{(p)}}^{(p)}, \mathbf{U}_{N^{(p)}}^{(p)}, t_{N^{(p)}}^{(p)}, \mathbf{s}) \end{bmatrix} \in \mathbb{R}^{N^{(p)} \times n_y^{(p)}},$$

- Path Constraint Matrix

$$\mathbf{C}^{(p)} = \begin{bmatrix} \mathbf{c}(\mathbf{Y}_1^{(p)}, \mathbf{U}_1^{(p)}, t_1^{(p)}, \mathbf{s}) \\ \vdots \\ \mathbf{c}(\mathbf{Y}_{N^{(p)}}^{(p)}, \mathbf{U}_{N^{(p)}}^{(p)}, t_{N^{(p)}}^{(p)}, \mathbf{s}) \end{bmatrix} \in \mathbb{R}^{N^{(p)} \times n_c^{(p)}},$$

- Integral Constraints

$$\rho_i^{(p)} = q_i^{(p)} - \frac{t_f^{(p)} - t_0^{(p)}}{2} \left[\mathbf{w}^{(p)}\right]^{\mathsf{T}} \mathbf{G}_i^{(p)}, \quad (i = 1, \ldots, n_q^{(p)}),$$

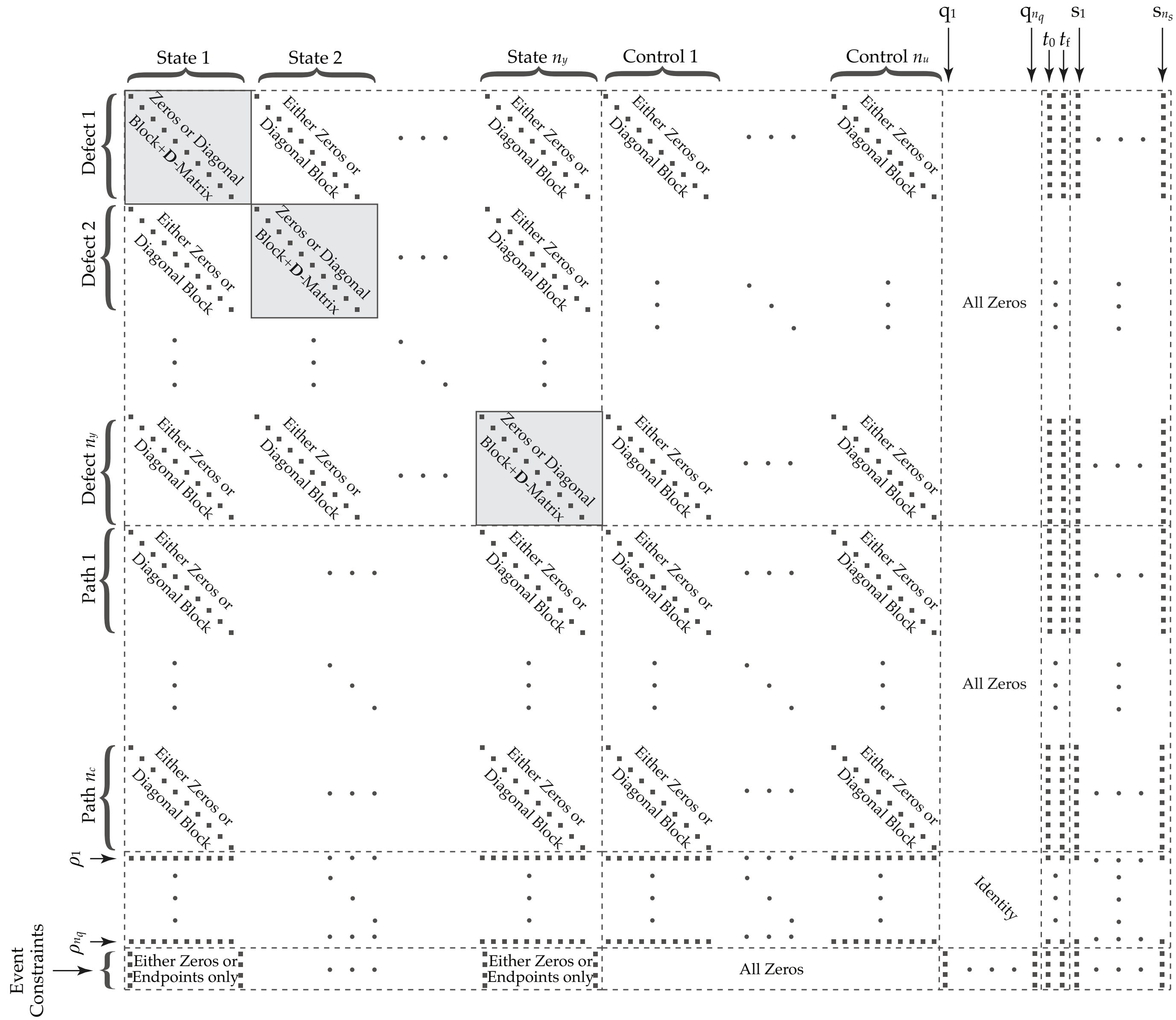
$$\mathbf{G}^{(p)} = \begin{bmatrix} \mathbf{g}(\mathbf{Y}_1^{(p)}, \mathbf{U}_1^{(p)}, t_1^{(p)}, \mathbf{s}) \\ \vdots \\ \mathbf{g}(\mathbf{Y}_{N^{(p)}}^{(p)}, \mathbf{U}_{N^{(p)}}^{(p)}, t_{N^{(p)}}^{(p)}, \mathbf{s}) \end{bmatrix} \in \mathbb{R}^{N^{(p)} \times n_q^{(p)}}.$$

- Equality Constraints

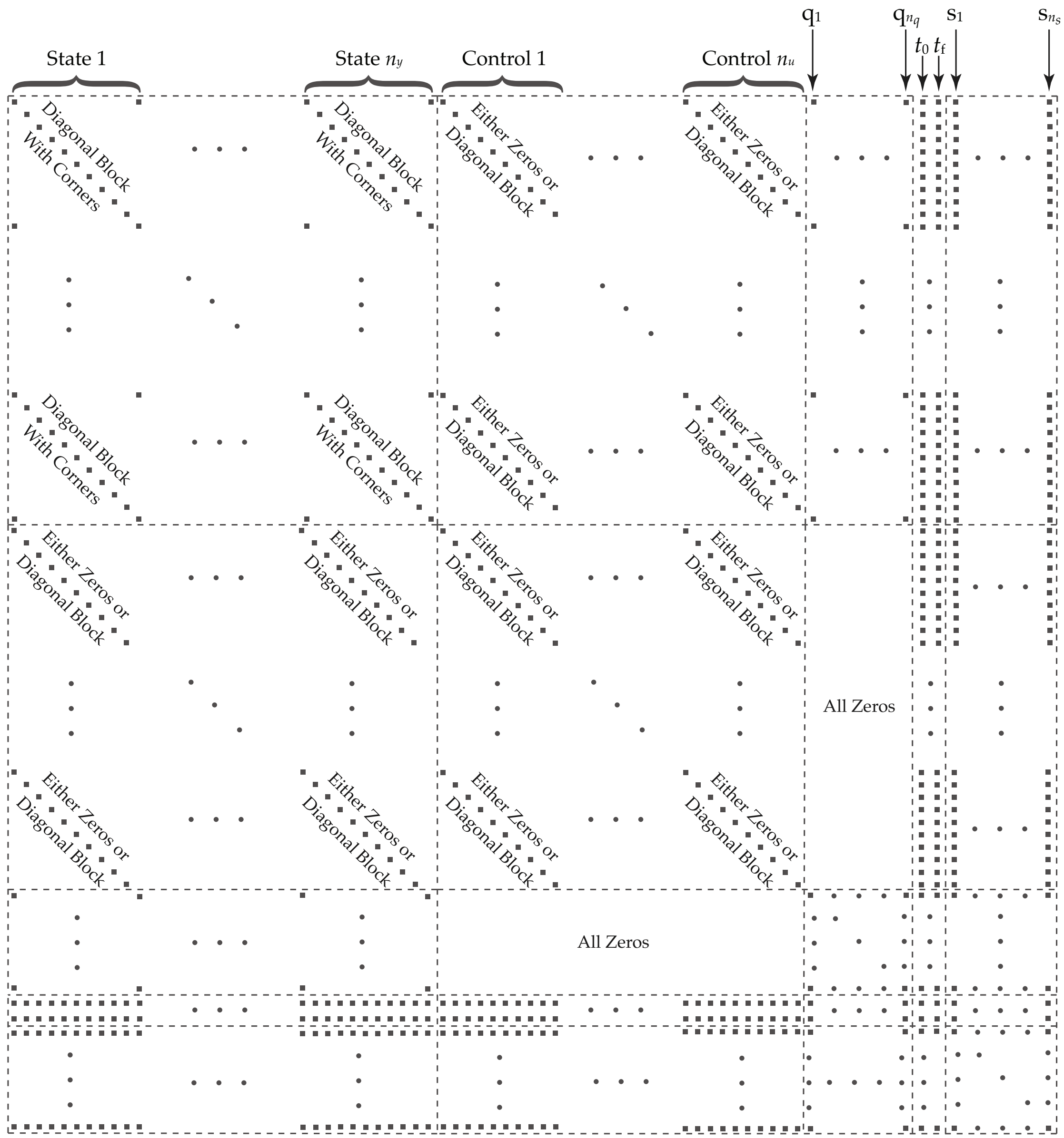
$$\begin{aligned} \boldsymbol{\Delta}^{(p)} &= \mathbf{0}, \\ \boldsymbol{\rho}^{(p)} &= \mathbf{0}, \end{aligned}$$

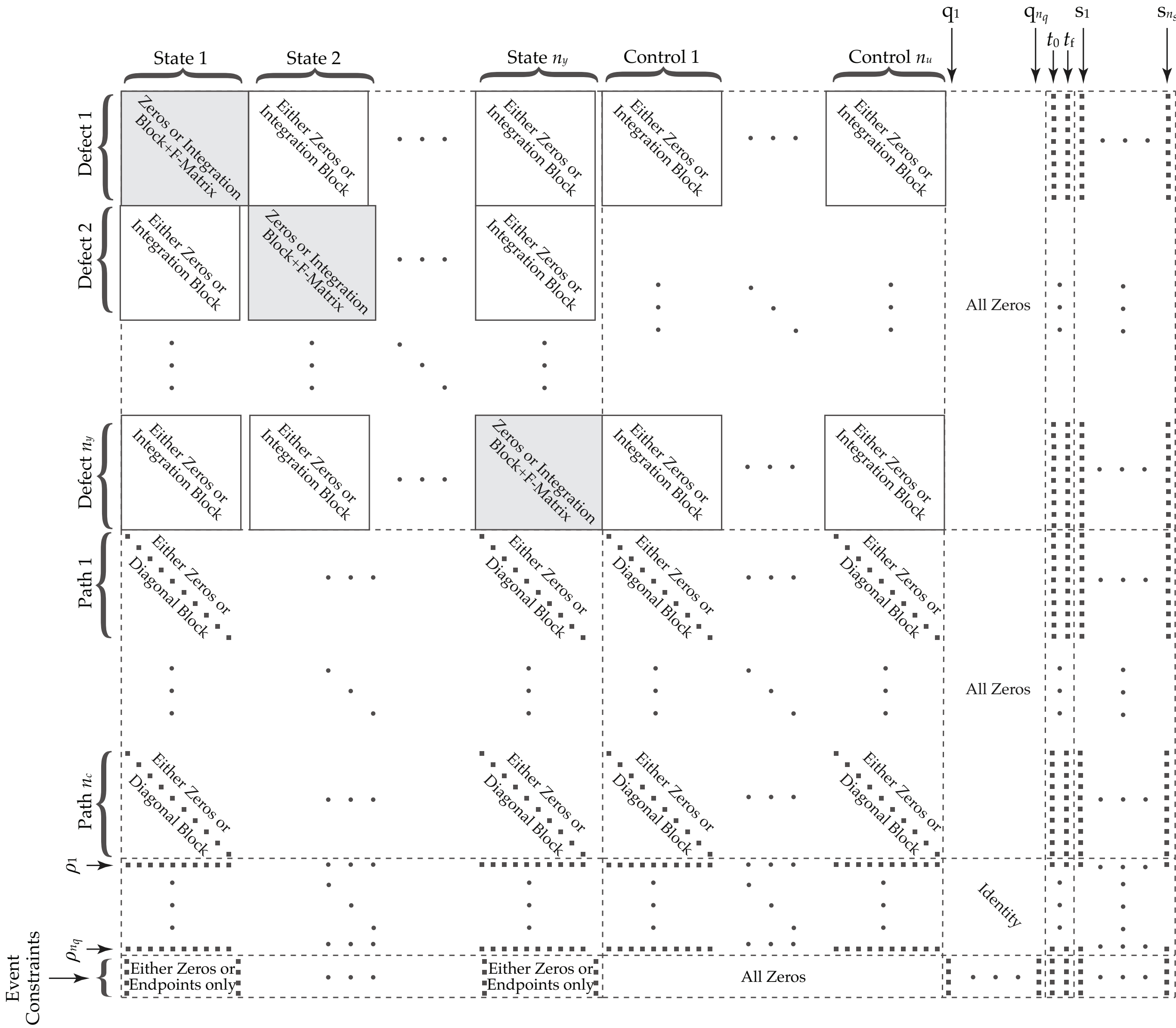
- Inequality Constraints

$$\begin{aligned} \mathbf{C}_{\min}^{(p)} &\leq \mathbf{C}^{(p)} \leq & \mathbf{C}_{\max}^{(p)}, \\ \mathbf{b}_{\min} &\leq \mathbf{b} \leq & \mathbf{b}_{\max}. \end{aligned}$$

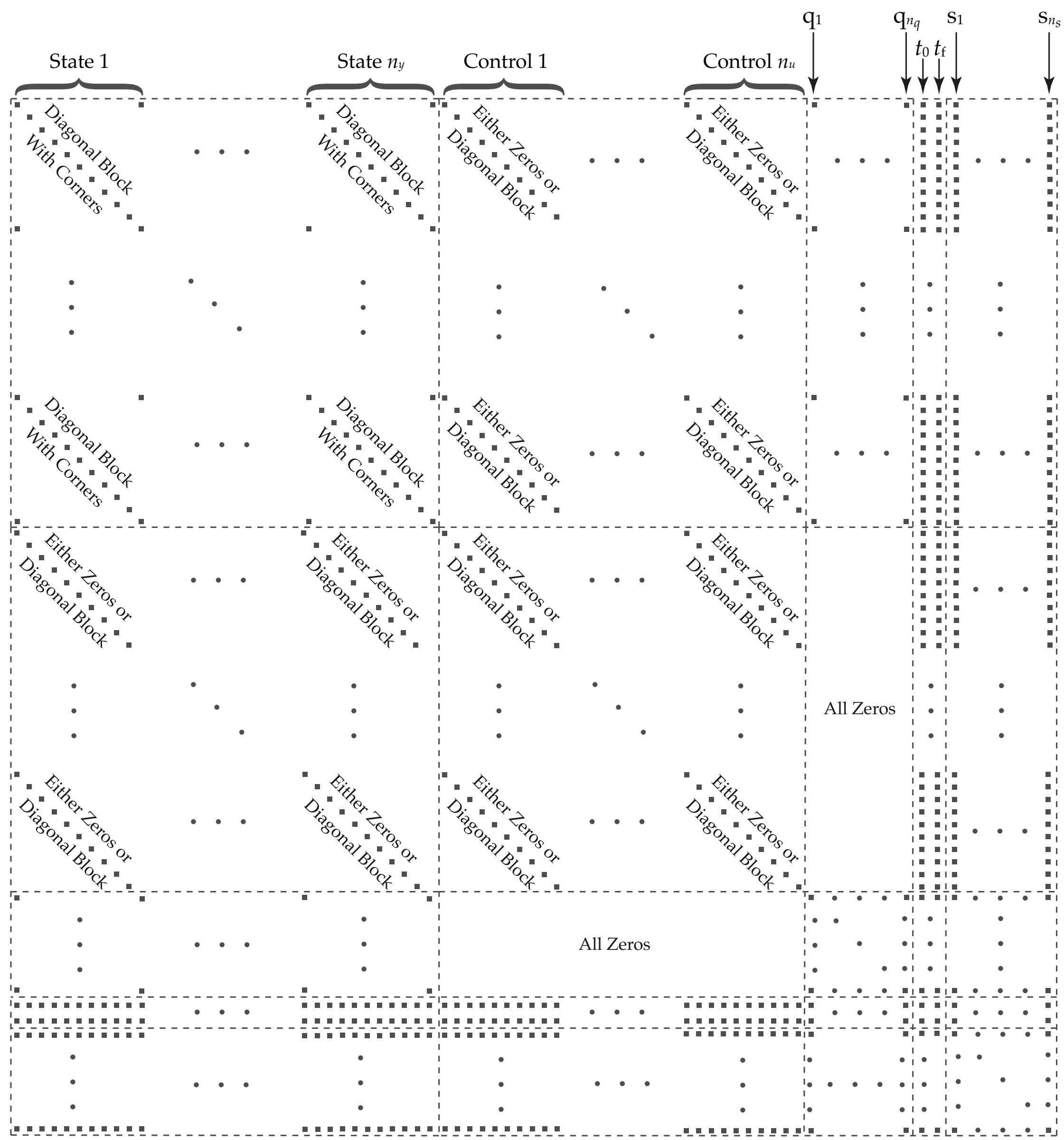


State 1
State 2
State n_y
Control 1
Control n_u
q_1
q_{n_q}
t_0
t_f
s_1
s_{n_s}
Defect 1
Defect 2
Defect n_y
Path 1
Path n_c
ρ_1
ρ_{n_q}
Event Constraints
Zeros or Diagonal Block+D-Matrix
Either Zeros or Diagonal Block
All Zeros
Identity
Either Zeros or Endpoints only


State 1
State n_y
Control 1
Control n_u
q_1
q_{n_q}
t_0
t_f
s_1
s_{n_s}
Diagonal Block With Corners
Either Zeros or Diagonal Block
All Zeros
All Zeros



State 1
State 2
State n_y
Control 1
Control n_u
q_1
q_{n_q}
t_0
t_f
s_1
s_{n_s}
Defect 1
Defect 2
Defect n_y
Path 1
Path n_c
ρ_1
ρ_{n_q}
Event Constraints
Zeros or Integration Block+F-Matrix
Either Zeros or Integration Block
Either Zeros or Diagonal Block
All Zeros
Identity
Either Zeros or Endpoints only


State 1
State n_y
Control 1
Control n_u
q_1
q_{n_q}
t_0 t_f
s_1
s_{n_s}
Diagonal Block With Corners
Either Zeros or Diagonal Block
All Zeros
All Zeros

Sparse Structure of NLP

- Structure Has Been Described in Detail by Patterson and Rao
- Multiple-Phase Optimal Control Problem
  - ▷ Constraint Jacobian: Block Diagonal Version of One-Phase Jacobian
  - ▷ Lagrangian Hessian: Block Diagonal Version of One-Phase Hessian

NLP Scaling

- Variables and Constraints Scaled to Approximately Order One
- Variable Scaling: Put All Variables on Unit Interval $[-1/2, 1/2]$
  - ▷ Suppose $x \in [a, b]$
  - ▷ Then $\tilde{x} \in [-1/2, 1/2]$

$$\tilde{x} = v_x x + r_x,$$

$$\begin{aligned} v_x &= \frac{1}{b-a}, \\ r_x &= \frac{1}{2} - \frac{b}{b-a}. \end{aligned}$$

- Constraint Scaling: Make Constraints $\mathcal{O}(1)$
  - ▷ Defect constraints scaled same as states
  - ▷ Objective, event & path constraints: scale by sampled average of gradient

Computation of Derivatives

- Exploits Sparse Structure of NLP
  - ▷ Patterson and Rao: only need optimal control function derivatives
  - ▷ $\mathbb{GPOPS}-\mathbb{II}$: Sparse forward, central, or backward approximations
- Example: Consider the optimal control function $\mathbf{f}(\mathbf{x})$, $\mathbf{f}:\mathbb{R}^n \longrightarrow \mathbb{R}^m$
  - ▷ Forward Difference Approximation of $\partial\mathbf{f}/\partial\mathbf{x}$

$$\frac{\partial \mathbf{f}}{\partial x_i} \approx \frac{\mathbf{f}(\mathbf{x}+\mathbf{h}_i)-\mathbf{f}(\mathbf{x})}{h_i},$$

  - ▷ $\mathbf{h}_i$ = Perturbation of i^{th} component of $\mathbf{x}$.
  - ▷ Then

$$\mathbf{h}_i = h_i \mathbf{e}_i$$

  - ▷ $\mathbf{e}_i = i^{th}$ row of the $n \times n$ identity matrix
  - ▷ h_i =Perturbation Associated with x_i.
  - ▷ Note that

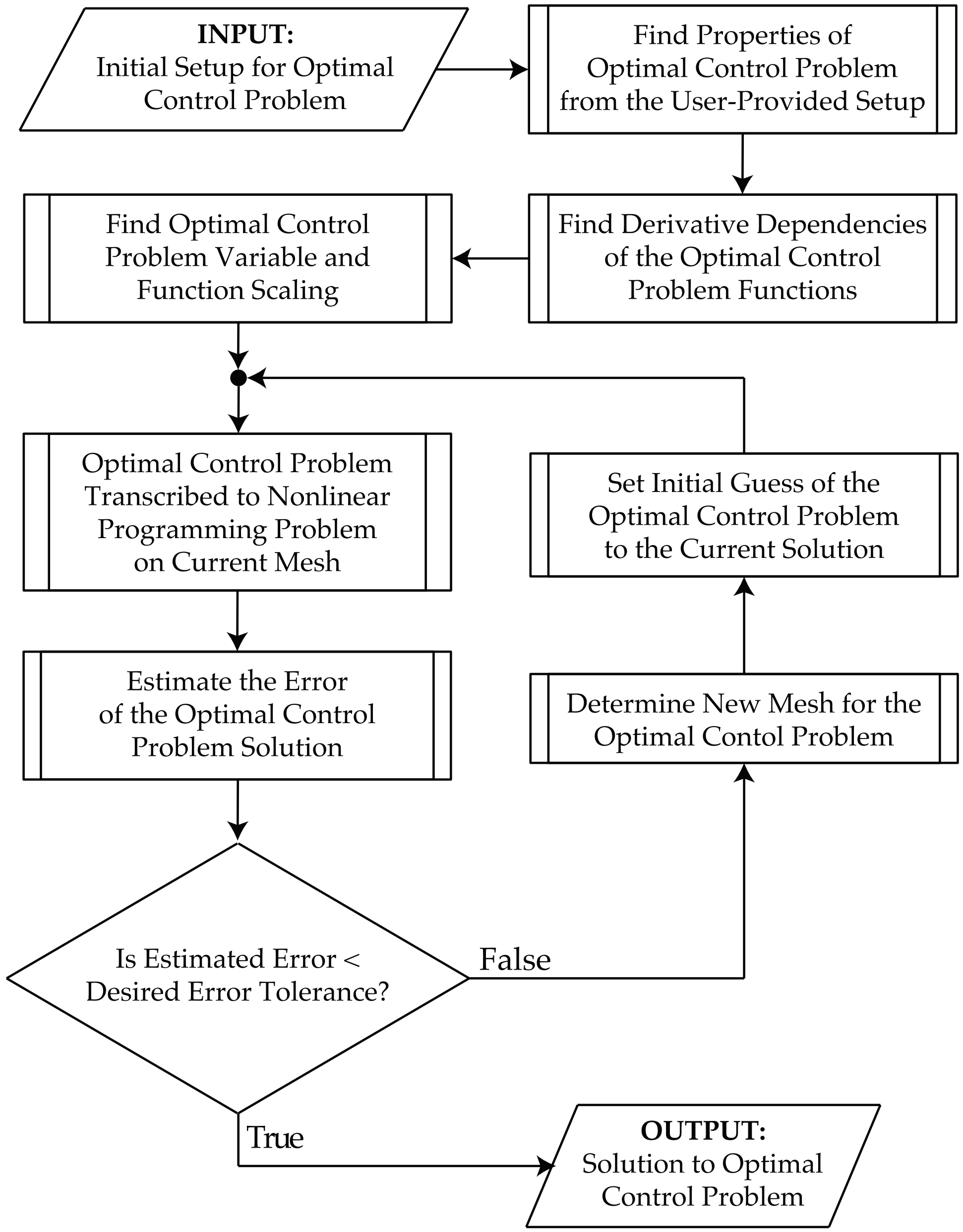
$$h_i = h(1+|x_i|),$$

- Base Perturbation Size = optimal for $\approx \mathcal{O}(1)$ Function

Adaptive Mesh Refinement

- $\mathbb{GPOPS} - \mathbb{II}$ Uses Variable-Order Mesh Refinement
- What Can Change Between Meshes?
  - ▷ Number of mesh intervals
  - ▷ Width of each mesh interval
  - ▷ Polynomial degree within a mesh interval
- Iterate on Mesh Until Specified Accuracy Tolerance is Met
- Either Method: Need to Place Lower and Upper Limits on Polynomial Degree
- Error Based on Relative Error in State
- Fundamental Difference Between Mesh Refinement Methodsf
  - ▷ *hp* method: use curvature to modify mesh
  - ▷ *ph* method: use exponential converge of Radau collocation
  - ▷ If Maximum Polynomial Degree Limit is Hit: Divide Into More Mesh Intervals

Algorithmic Flow of GPOPS – II



Overview of $\mathbb{GPOPS} - \mathbb{II}$ Usage

Main call to $\mathbb{GPOPS} - \mathbb{II}$:

$$\textit{output}=\textbf{gpops2}(\textit{input}),$$

Required Fields:

- **name**: a string *with no blank spaces* that contains the name of the problem;
- **functions**: a structure that contains the name of the continuous function and the endpoint function;
- **bounds**: an structure that contains the information about the lower and upper bounds on the different variables and constraints in the problem;
- **guess**: an structure that contains a guess of the time, state, control, integrals, and static parameters in the problem;

Optional Fields:

- **auxdata**: a structure containing auxiliary data that may be used by different functions in the problem. Including **auxdata** eliminates any need to specify global variables for use in the problem. The following table provided the possible values and their defaults for the field *setup*.**auxdata**:

Field	Possible Values	Default
setup.**auxdata**	Any Problem-Specific Data	Not Provided

- **derivatives**: a structure that specifies the derivative approximation to be used by the NLP solver and the derivative order ('first' or 'second') to be used by the NLP solver. The field *setup*.**derivatives** contains three fields **supplier**, **derivativelevel**, and **dependencies** where the field *setup*.**derivatives.supplier** contains the type of derivative approximation , the field *setup*.**derivatives.derivativelevel** contains the derivative order, while the field *setup*.**derivatives.dependencies** determines how the dependencies are found. The following table provided the possible values and their defaults for the field *setup*.**derivatives**:

Field	Possible Values	Default
setup.**derivatives.supplier**	'sparseFD' or 'sparseBD' or 'sparseCD' or 'adigator'	'sparseFD'
setup.**derivatives.derivativelevel**	'first' or 'second'	'first'
setup.**derivatives.dependencies**	'full', 'sparse' or 'sparseNaN'	'sparseNaN'

  Note that the option *setup*.**derivatives.supplier** is ignored when the derivative option is set to either 'analytic' or 'adigator'.

- **scales**: a structure that specifies how the problem to be solved is scaled. The field **scales** itself contains a field **method** that can be set to one of the following:

Field	Possible Values	Default
setup.**scales**.**method**	'none' or 'automatic-bounds' or 'automatic-bounds' or 'automatic-guess' 'automatic-guessUpdate' or 'automatic-hybrid' or 'automatic-hybridUpdate' or 'defined'	'none'

- **method**: a string that defines the version of the collocation to be used when solving the problem. Valid options are

Field	Possible Values	Default
setup.**method**	'RPM-Differentiation' or 'RPM-Integration'	'RPM-Differentiation'

- **mesh**: a structure that specifies the information as to the type of mesh refinement method to be used and the mesh refinement accuracy tolerance, as well as the initial mesh. The structure *setup*.**mesh** contains the fields **method**, **tolerance**, **maxiterations**, and **phase**. The field *setup*.**mesh**.**method** is a string that specified the particular mesh refinement method to be used, while the field *setup*.**mesh**.**tolerance** contains the desired accuracy tolerance of the mesh, while the field *setup*.**mesh**.**maxiterations** contains the maximum number of allowed mes iterations.

Field	Possible Values	Default
setup.**mesh.method**	'hp-PattersonRao' or 'hp-DarbyRao' or 'hp-LiuRao'	'hp-PattersonRao'
setup.**mesh.tolerance**	Positive Number Between 0 and 1	10^{-3}
setup.**mesh.maxiteration**	Non-Negative Integer	10

Field	Possible Values	Default
setup.**mesh.phase(p).fraction**	Row Vector of Length $M \geq 1$ of Positive Numbers > 0 and < 1 that Sum to Unity	0.1*ones(1,10)
setup.**mesh.phase(p).colpoints**	Row Vector of Length $M \geq 1$ of Positive Integers > 1 and < 10 (M is the same as in *setup*.**mesh.phase(p).fraction**)	4*ones(1,10)

- **nlp**: a structure that specifies the NLP solver to be used and the options to be used within the chosen NLP solver. *setup*.**nlp** contains the field **solver** and **options**. The field **solver** contains a string indicating the NLP solver to be used. The fields **ipoptoptions** and **snoptoptions** are structures that themselves contains fields with options for the NLP solvers IPOPT and SNOPT, respectively, which can be set from $\mathbb{GPOPS}-\mathbb{II}$.

Field	Possible Values	Default
setup.**nlp.solver**	'snopt' or 'ipopt'	'ipopt'

Field **nlp.ipoptoptions** for the NLP solver IPOPT:

Field	Possible Values	Default
setup.**nlp.ipoptoptions.linear_solver**	'mumps' or 'ma57'	'mumps'
setup.**nlp.ipoptoptions.tolerance**	Positive Real Number	10^{-7}
setup.**nlp.ipoptoptions.maxiterations**	Positive Integer	2000

Field **nlp.snoptoptions** for the NLP solver SNOPT:

setup.**nlp.snoptoptions.tolerance**	Positive Real Number	10^{-6}
setup.**nlp.snoptoptions.maxiterations**	Positive Integer	2000

- **displaylevel**: a integer that takes on the values 0, 1, or 2 and provides the amount of output that is sent to the MATLAB command window during an execution of $\mathbb{GPOPS} - \mathbb{II}$. The following table provided the possible values and their defaults for the field *setup*.**auxdata**:

Field	Possible Values	Default
setup.**displaylevel**	0, 1, or 2	2

Syntax for Structure *setup*.**functions**

setup.**functions**.**continuous**	=	*@continuousfun.m*
setup.**functions**.**endpoint**	=	*@endpointfun.m*

Syntax for **bounds** Structure

- **bounds.phase(p).initialtime.{lower,upper}**: scalars that contain the information about the lower and upper bounds on the initial time in phase $p \in [1, \ldots, P]$.

$$\begin{aligned} \textbf{bounds.phase}(p)\textbf{.initialtime.lower} &= t_0^{\text{lower}} \\ \textbf{bounds.phase}(p)\textbf{.initialtime.upper} &= t_0^{\text{upper}} \end{aligned}$$

- **bounds.phase(p).finaltime.{lower,upper}**: scalars that contain the information about the lower and upper bounds on the final time in phase $p \in [1, \ldots, P]$.

$$\begin{aligned} \textbf{bounds.phase}(p)\textbf{.finaltime.lower} &= t_f^{\text{lower}} \\ \textbf{bounds.phase}(p)\textbf{.finaltime.upper} &= t_f^{\text{upper}} \end{aligned}$$

- **bounds.phase(p).initialstate.{lower,upper}**: row vectors of length $n_y^{(p)}$ that contain the lower and upper bounds on the initial state in phase

$p \in [1, \ldots, P]$.

$$
\begin{aligned}
\textbf{bounds.phase}(p)\textbf{.initialstate.lower} &= \begin{bmatrix} y_{0,1}^{\text{lower}} & \cdots & y_{0,n_y^{(p)}}^{\text{lower}} \end{bmatrix} \\
\textbf{bounds.phase}(p)\textbf{.initialstate.upper} &= \begin{bmatrix} y_{0,1}^{\text{upper}} & \cdots & y_{0,n_y^{(p)}}^{\text{upper}} \end{bmatrix}
\end{aligned}
$$

- **bounds.phase(p).state.{lower,upper}**: row vectors of length $n_y^{(p)}$ that contain the lower and upper bounds on the state during phase $p \in [1, \ldots, P]$.

$$
\begin{aligned}
\textbf{bounds.phase}(p)\textbf{.state.lower} &= \begin{bmatrix} y_{1}^{\text{lower}} & \cdots & y_{n_y^{(p)}}^{\text{lower}} \end{bmatrix} \\
\textbf{bounds.phase}(p)\textbf{.state.upper} &= \begin{bmatrix} y_{1}^{\text{upper}} & \cdots & y_{n_y^{(p)}}^{\text{upper}} \end{bmatrix}
\end{aligned}
$$

- **bounds.phase(p).finalstate.{lower,upper}**: row vectors of length $n_y^{(p)}$ that contain the lower and upper bounds on the final state in phase $p \in [1, \ldots, P]$.

$$
\begin{aligned}
\textbf{bounds.phase}(p)\textbf{.finalstate.lower} &= \begin{bmatrix} y_{f,1}^{\text{lower}} & \cdots & y_{f,n_y^{(p)}}^{\text{lower}} \end{bmatrix} \\
\textbf{bounds.phase}(p)\textbf{.finalstate.upper} &= \begin{bmatrix} y_{f,1}^{\text{upper}} & \cdots & y_{f,n_y^{(p)}}^{\text{upper}} \end{bmatrix}
\end{aligned}
$$

- **bounds.phase(p).control.{lower,upper}**: row vectors of length $n_y^{(p)}$ that contain the lower and upper bounds on the control during phase $p \in [1, \ldots, P]$.

$$\begin{array}{lcl} \textbf{bounds.phase}(p)\textbf{.control.lower} & = & \left[\begin{array}{ccc} u_1^{\text{lower}} & \cdots & u_{n_u^{(p)}}^{\text{lower}} \end{array}\right] \\ \textbf{bounds.phase}(p)\textbf{.control.upper} & = & \left[\begin{array}{ccc} u_1^{\text{upper}} & \cdots & u_{n_u^{(p)}}^{\text{upper}} \end{array}\right] \end{array}$$

- **bounds.phase(p).path.{lower,upper}**: row vectors of length $n_c^{(p)}$ that contain the lower and upper bounds on the path constraints during phase $p \in [1, \ldots, P]$.

$$\begin{array}{lcl} \textbf{bounds.phase}(p)\textbf{.path.lower} & = & \left[\begin{array}{ccc} c_1^{\text{lower}} & \cdots & c_{n_u^{(p)}}^{\text{lower}} \end{array}\right] \\ \textbf{bounds.phase}(p)\textbf{.path.upper} & = & \left[\begin{array}{ccc} c_1^{\text{upper}} & \cdots & c_{n_u^{(p)}}^{\text{upper}} \end{array}\right] \end{array}$$

- **bounds.phase(p).integral.{lower,upper}**: row vectors of length $n_q^{(p)}$ that contain the lower and upper bounds on the integrals in phase

$p \in [1, \ldots, P]$.

$$\begin{array}{lcl} \textbf{bounds.phase}(p)\textbf{.integral.lower} & = & \left[\begin{array}{ccc} q_1^{\text{lower}} & \cdots & q_{n_q^{(p)}}^{\text{lower}} \end{array} \right] \\ \textbf{bounds.phase}(p)\textbf{.integral.upper} & = & \left[\begin{array}{ccc} q_1^{\text{upper}} & \cdots & q_{n_q^{(p)}}^{\text{upper}} \end{array} \right] \end{array}$$

- **bounds.phase(p).duration.{lower,upper}**: scalars that contain the lower and upper bounds on the duration of a phases $p \in [1, \ldots, P]$. The duration is the difference between the final time of the phase and the initial time of the phase, $t_f^{(p)} - t_0^{(p)}$.
- **bounds.parameter.{lower,upper}**: row vectors of length n_s that contain the lower and upper bounds on the static parameters in the problem.

$$\begin{array}{lcl} \textbf{bounds.parameter.lower} & = & \left[\begin{array}{ccc} s_1^{\text{lower}} & \cdots & s_{n_s}^{\text{lower}} \end{array} \right] \\ \textbf{bounds.parameter.upper} & = & \left[\begin{array}{ccc} s_1^{\text{upper}} & \cdots & s_{n_s}^{\text{upper}} \end{array} \right] \end{array}$$

- **bounds.eventgroup(g).{lower,upper}**: row vectors of length $n_b^{(g)}$ that contain the lower and upper bounds on the group $g = 1, \ldots, G$ of event constraints.

$$
\begin{aligned}
\textbf{bounds.eventgroup}(g)\textbf{.lower} &= \begin{bmatrix} b_1^{\text{lower}} & \cdots & b_{n_b^{(g)}}^{\text{lower}} \end{bmatrix} \\
\textbf{bounds.eventgroup}(g)\textbf{.upper} &= \begin{bmatrix} b_1^{\text{upper}} & \cdots & b_{n_b^{(g)}}^{\text{upper}} \end{bmatrix}
\end{aligned}
$$

Syntax for Endpoint Function

function output=endpointfun(input)

- *input*.**phase(p).initialtime**: a scalar that contains the initial time in phase $p = 1, \ldots, P$;
- *input*.**phase(p).finaltime**: a scalar that contains the final time in phase $p = 1, \ldots, P$;
- *input*.**phase(p).initialstate**: a row vector of length $n_y^{(p)}$ that contains the initial state in phase $p = 1, \ldots, P$;
- *input*.**phase(p).finalstate**: a row vector of length $n_y^{(p)}$ that contains the final state in phase $p = 1, \ldots, P$;
- *input*.**phase(p).integral**: a row vector of length $n_d^{(p)}$ that contains the integrals in phase $p = 1, \ldots, P$;
- *input*.**parameter**: a row vector of length n_s that contains the static parameters in phase $p = 1, \ldots, P$;
- *output*.**objective**: a scalar that contains the result of computing the

objective function on the current call to *input*.**functions**.**endpoint**;

- *output*.**eventgroup**: an array of structures of length G (where G is the number of event groups) such that the g^{th} element in *output*.**eventgroup** is a row vector of length $n_b^{(g)}$ that contains the result of evaluating g^{th} group of event constraints at the values given in the call to the function *input*.**functions**.**endpoint**;

Syntax for Endpoint Function

function output=continuousfun(input)

Fields of Structure *input*:

- *input*.**phase(p).time**: a column vector of length $N^{(p)}$, where $N^{(p)}$ is the number of collocation points in phase $p = 1, \ldots, P$.
- *input*.**phase(p).state**: a matrix of size $N^{(p)} \times n_y^{(p)}$, where $N^{(p)}$ and $n_y^{(p)}$ are, respectively, the number of collocation points and the dimension of the state in phase $p = 1, \ldots, P$;
- *input*.**phase(p).control**: a matrix of size $N^{(p)} \times n_u^{(p)}$, where $N^{(p)}$ and $n_u^{(p)}$ are, respectively, the number of collocation points and the dimension of the control in phase $p = 1, \ldots, P$;
- *input*.**phase(p).parameter**: a matrix of size $N^{(p)} \times n_s$, where $N^{(p)}$ is the number of collocation points in phase $p = 1, \ldots, P$ and and n_s is the dimension of the static parameter. [**Note:** see below for the reason why the static parameter has a size $N^{(p)} \times n_s$];

Fields of Structure *output*:

- *output*.**dynamics**: a matrix of size $N^{(p)} \times n_y^{(p)}$, where $N^{(p)}$ and $n_y^{(p)}$ are, respectively, the number of collocation points and the dimension of the state in phase $p = 1, \ldots, P$;
- *output*.**path**: a matrix of size $N^{(p)} \times n_c^{(p)}$, where $N^{(p)}$ and $n_c^{(p)}$ are, respectively, the number of collocation points and the number of path constraints in phase $p = 1, \ldots, P$;
- *output*.**integrand**: a matrix of size $N^{(p)} \times n_d^{(p)}$, where $N^{(p)}$ and $n_d^{(p)}$ are, respectively, the number of collocation points and the number of integrals in phase $p = 1, \ldots, P$;

Specifying an Initial Guess for a $\mathbb{GPOPS} - \mathbb{II}$ Run

- *setup*.**guess.phase(**p**).time**: a column vector of length $M^{(p)}$ in phase $p = 1, \ldots, P$;
- *setup*.**guess.phase(**p**).state**: a matrix of size $M^{(p)} \times n_y^{(p)}$, where $n_y^{(p)}$ is the dimension of the state in phase $p = 1, \ldots, P$;
- *setup*.**guess.phase(**p**).control**: a matrix of size $M^{(p)} \times n_u^{(p)}$, where $n_u^{(p)}$ is the dimension of the control in phase $p = 1, \ldots, P$;
- *setup*.**guess.phase(**p**).integral**: a row vector of length $n_d^{(p)}$, where $n_d^{(p)}$ is the number of integrals in phase $p = 1, \ldots, P$;
- *setup*.**guess.parameter**: a row vector of length size n_s, where n_s is the number of static parameters in the problem.

NLP Solver Options with GPOPS – II

- IPOPT (Default)
  - ▷ Open source
  - ▷ Employs and interior-point method
  - ▷ Two different derivative modes
    - ∘ Quasi-Newton: first derivatives only
    - ∘ Full-Newton: first and second derivatives
  - ▷ Key characteristics of IPOPT
    - ∘ Mediocre performance in quasi-Newton mode
    - ∘ Extremely fast in full-Newton mode
    - ∘ Scales extremely well as problem size increases
    - ∘ Issue using mesh refinement
      - · Does not take full advantage of good initial guess
      - · Backs away From solution due to interior-point method
    - ∘ Kinds of problems for which IPOPT works well (in full-Newton mode)
      - · Multiple time-scale (ill-conditioned) problems

- Problems with not so well defined minima
- Problems that do not require a good initial guess

- SNOPT (not included; must be obtained separately)
  - Available commercially
  - Employs a quasi-Newton SQP method
  - Key characteristics of SNOPT
    - Very good general-purpose NLP solver
    - Not always fast, but good for reasonable range of problems
    - Does not scale well as problem size increases
    - Converges very quickly with good initial guess

Examples of Using $\mathbb{GPOPS} - \mathbb{II}$

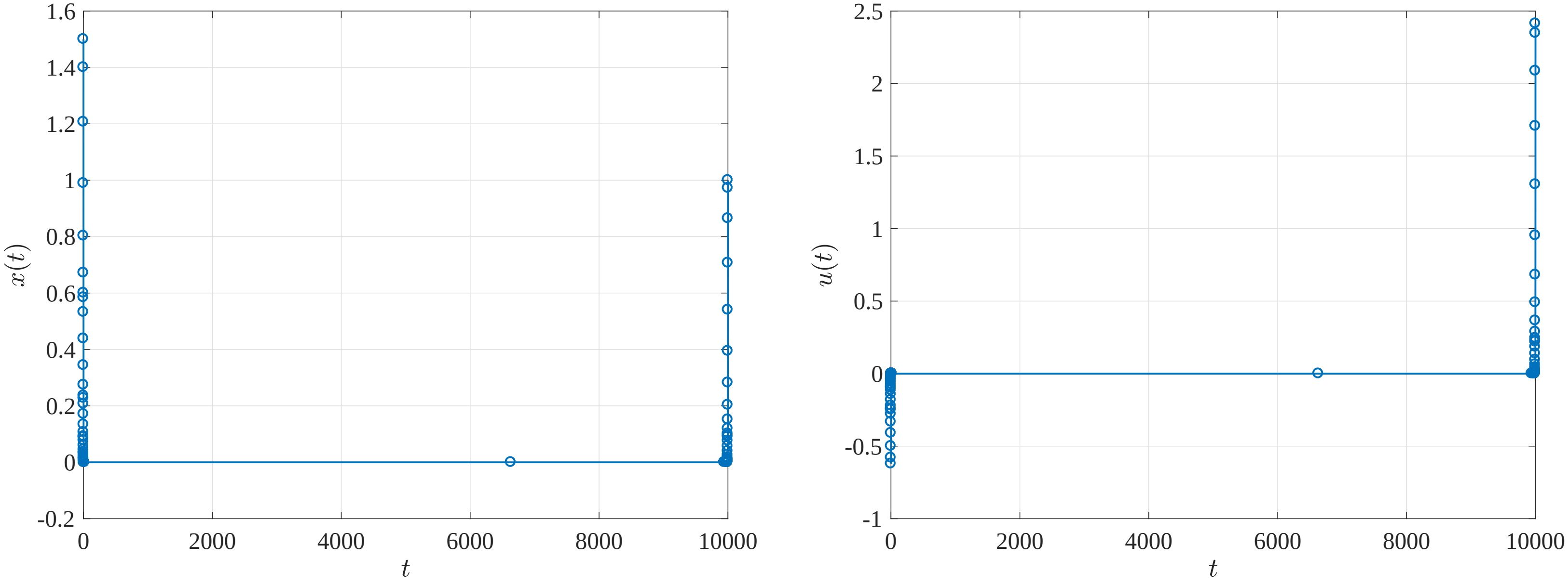
Hyper-Sensitive Problem

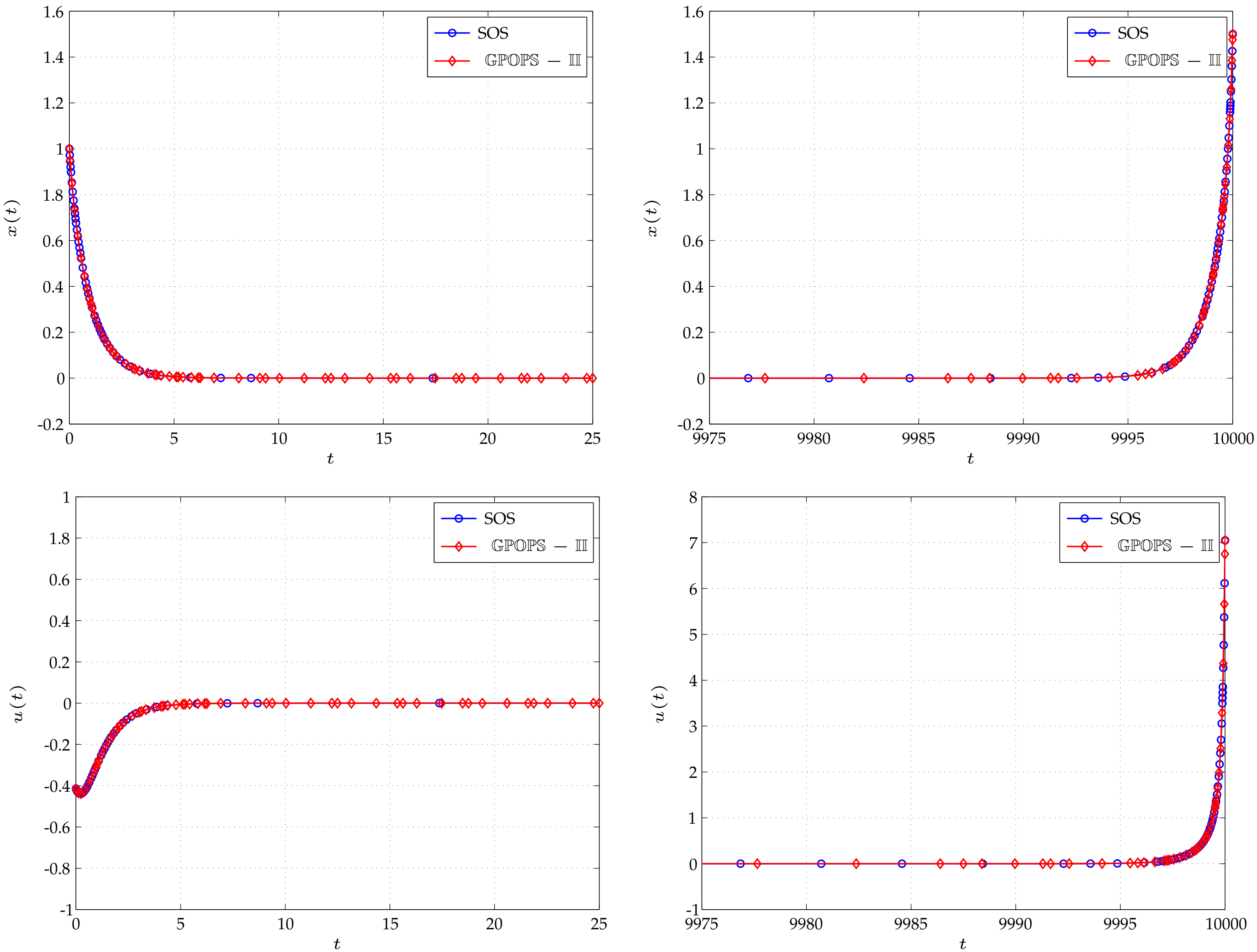
$$\text{minimize } \frac{1}{2}\int_0^{t_f}(x^2+u^2)dt \text{ subject to } \begin{cases} \dot{x} & = & -x^3+u, \\ x(0) & = & 1, \\ x(t_f) & = & 1.5, \\ t_f = 10000. \end{cases}$$

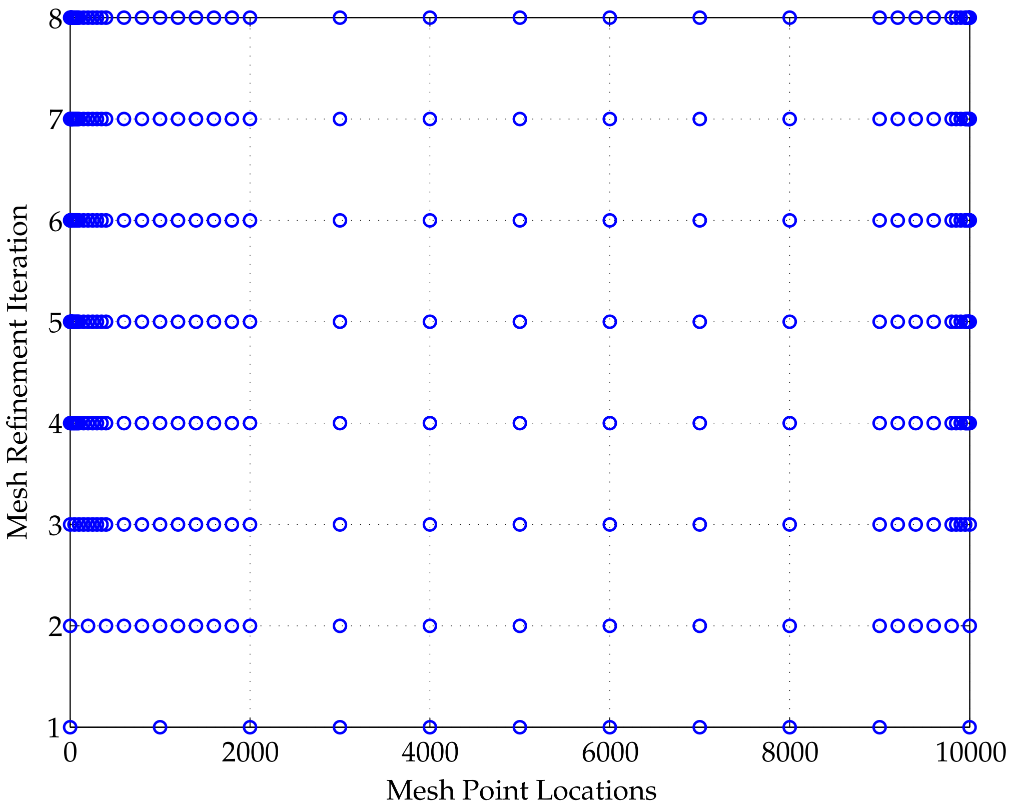
Initial Guess

$$\begin{aligned} x^{\text{guess}} &= \mathbf{linear}(0, t_f, x(0), x(t_f)), \\ u^{\text{guess}} &= 0, \\ t_f^{\text{guess}} &= t_f. \end{aligned}$$

Next: Closer Examination of the $\mathbb{GPOPS}-\mathbb{II}$ Code


x(t)
t
u(t)
t


SOS
GPOPS − II
x(t)
u(t)
t
-0.2
0
0.2
0.4
0.6
1.8
1
1.2
1.4
1.6
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
1.8
1
2
3
4
5
6
7
8
0
5
10
15
20
25
9975
9980
9985
9990
9995
10000



Mesh Refinement Iteration
1
2
3
4
5
6
7
8
0
2000
4000
6000
8000
10000
Mesh Point Locations

Mesh	Relative Error Estimate
1	2.827×10^{1}
2	2.823×10^{0}
3	7.169×10^{-1}
4	1.799×10^{-1}
5	7.092×10^{-2}
6	8.481×10^{-3}
7	1.296×10^{-3}
8	5.676×10^{-7}

- Note Performance of $\mathbb{GPOPS} - \mathbb{II}$
- Accurately Captures Rapid Transients in Solution
- Achieves a High-Accuracy Solution
- Concentrates Mesh Points Near Key Features in Solution

Reusable Launch Vehicle Entry

Maximize

$$J = \phi(t_f)$$

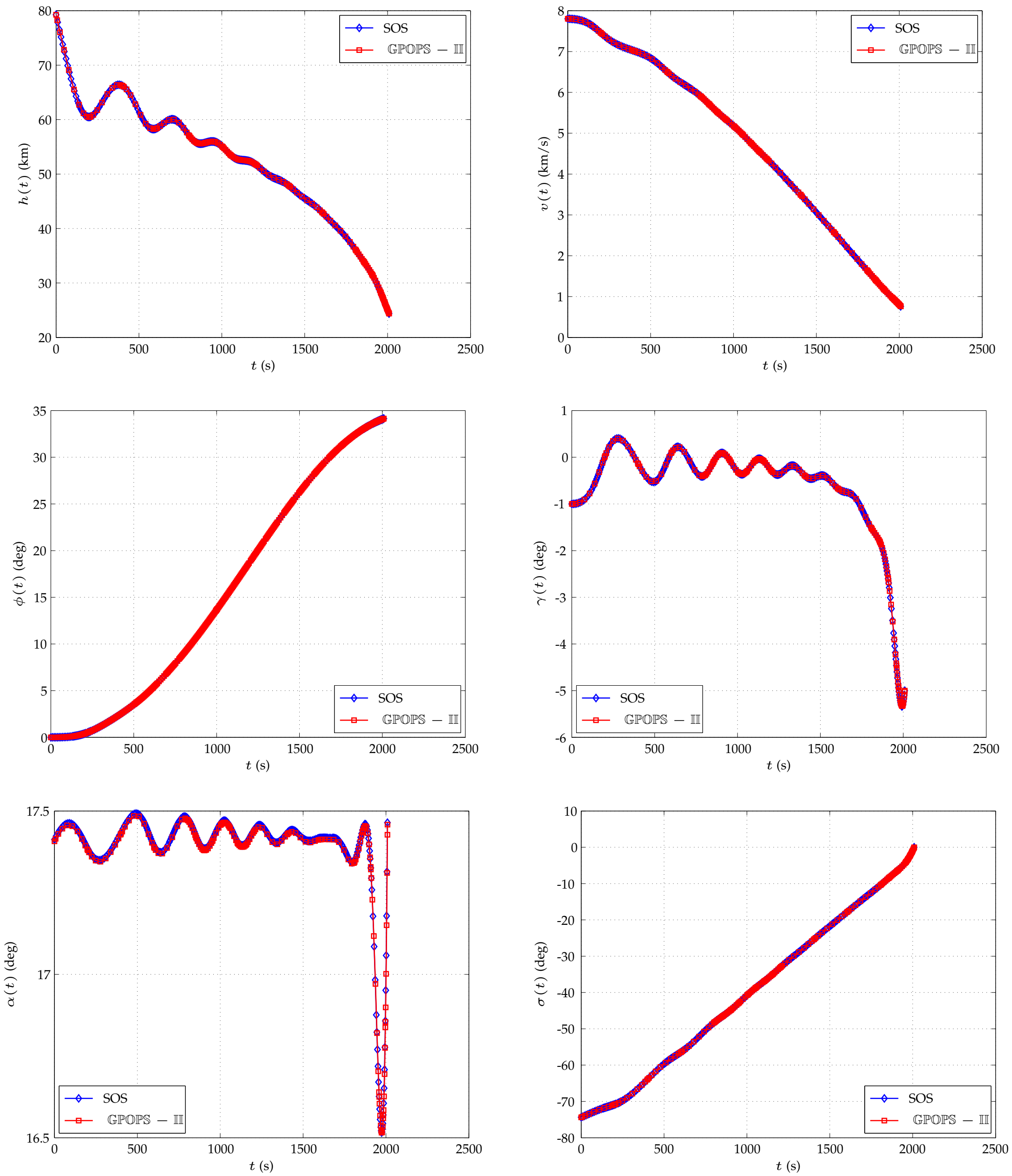
Subject to

$$\begin{aligned}
\dot{r} &= v \sin\gamma &, \quad \dot{\theta} &= \frac{v \cos\gamma \sin\psi}{r \cos\phi}, \\
\dot{\phi} &= \frac{v \cos\gamma \cos\psi}{r} &, \quad \dot{v} &= -\frac{D}{m} - g \sin\gamma, \\
\dot{\gamma} &= \frac{L \cos\sigma}{mv} - \left(\frac{g}{v} - \frac{v}{r}\right) \cos\gamma &, \quad \dot{\psi} &= \frac{L \sin\sigma}{mv \cos\gamma} + \frac{v \cos\gamma \sin\psi \tan\phi}{r},
\end{aligned}$$

and the boundary conditions

$$\begin{aligned}
h(0) &= 79248 \text{ km} &, \quad h(t_f) &= 24384 \text{ km}, \\
\theta(0) &= 0 \text{ deg} &, \quad \theta(t_f) &= \text{Free}, \\
\phi(0) &= 0 \text{ deg} &, \quad \phi(t_f) &= \text{Free}, \\
v(0) &= 7.803 \text{ km/s} &, \quad v(t_f) &= 0.762 \text{ km/s} \\
\gamma(0) &= -1 \text{ deg} &, \quad \gamma(t_f) &= -5 \text{ deg}, \\
\psi(0) &= 90 \text{ deg} &, \quad \psi(t_f) &= \text{Free}
\end{aligned}$$

Next: Closer Examination of the $\mathbb{GPOPS} - \mathbb{II}$ Code


SOS
GPOPS − II
$h(t)$ (km)
$v(t)$ (km/s)
$\phi(t)$ (deg)
$\gamma(t)$ (deg)
$\alpha(t)$ (deg)
$\sigma(t)$ (deg)
t (s)

Mesh Iteration	Estimated Error ($\mathbb{GPOPS} - \mathbb{II}$)	Number of Collocation Points	Estimated Error (SOS)	Number of Collocation Points
1	2.463×10^{-3}	41	1.137×10^{-2}	51
2	2.946×10^{-4}	103	1.326×10^{-3}	101
3	1.202×10^{-5}	132	3.382×10^{-5}	101
4	8.704×10^{-8}	175	1.314×10^{-6}	101
5	–	–	2.364×10^{-7}	201
6	–	–	2.364×10^{-7}	232
7	–	–	1.006×10^{-7}	348
8	–	–	9.933×10^{-8}	353

- Note Performance of $\mathbb{GPOPS} - \mathbb{II}$ vs SOS
- $\mathbb{GPOPS} - \mathbb{II}$ Uses Fewer Mesh Refinement Iterations
- Also, $\mathbb{GPOPS} - \mathbb{II}$ Generates a Much Smaller Mesh

Space Station Attitude Control

$$J = \frac{1}{2}\int_{t_0}^{t_f} \mathbf{u}^{\mathsf{T}}\mathbf{u}dt$$

subject to the dynamic constraints

$$\begin{aligned}
\dot{\boldsymbol{\omega}} &= \mathbf{J}^{-1}\left\{\boldsymbol{\tau}_{gg}(\mathbf{r}) - \boldsymbol{\omega}^{\otimes}\left[\mathbf{J}\boldsymbol{\omega} + \mathbf{h}\right] - \mathbf{u}\right\}, \\
\dot{\mathbf{r}} &= \frac{1}{2}\left[\mathbf{r}\mathbf{r}^{\mathsf{T}} + \mathbf{I} + \mathbf{r}\right]\left[\boldsymbol{\omega} - \boldsymbol{\omega}(\mathbf{r})\right], \\
\dot{\mathbf{h}} &= \mathbf{u},
\end{aligned}$$

the inequality path constraint

$$\|\mathbf{h}\| \leq h_{\max},$$

and the boundary conditions

$$\begin{aligned}
t_0 &= 0, \\
t_f &= 1800, \\
\boldsymbol{\omega}(0) &= \bar{\boldsymbol{\omega}}_0, \\
\mathbf{r}(0) &= \bar{\mathbf{r}}_0, \\
\mathbf{h}(0) &= \bar{\mathbf{h}}_0, \\
\mathbf{0} &= \mathbf{J}^{-1}\left\{\boldsymbol{\tau}_{gg}(\mathbf{r}(t_f)) - \boldsymbol{\omega}^{\otimes}(t_f)\left[\mathbf{J}\boldsymbol{\omega}(t_f) + \mathbf{h}(t_f)\right]\right\}, \\
\mathbf{0} &= \tfrac{1}{2}\left[\mathbf{r}(t_f)\mathbf{r}^{\mathsf{T}}(t_f) + \mathbf{I} + \mathbf{r}(t_f)\right]\left[\boldsymbol{\omega}(t_f) - \boldsymbol{\omega}_0(\mathbf{r}(t_f))\right],
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{\omega}_0(\mathbf{r}) &= -\omega_{\text{orb}}\mathbf{C}_2, \\
\boldsymbol{\tau}_{gg} &= 3\omega_{\text{orb}}^2\mathbf{C}_3^{\otimes}\mathbf{J}\mathbf{C}_3,
\end{aligned}$$

and $\mathbf{C}_2$ and $\mathbf{C}_3$ are the second and third column, respectively, of the matrix

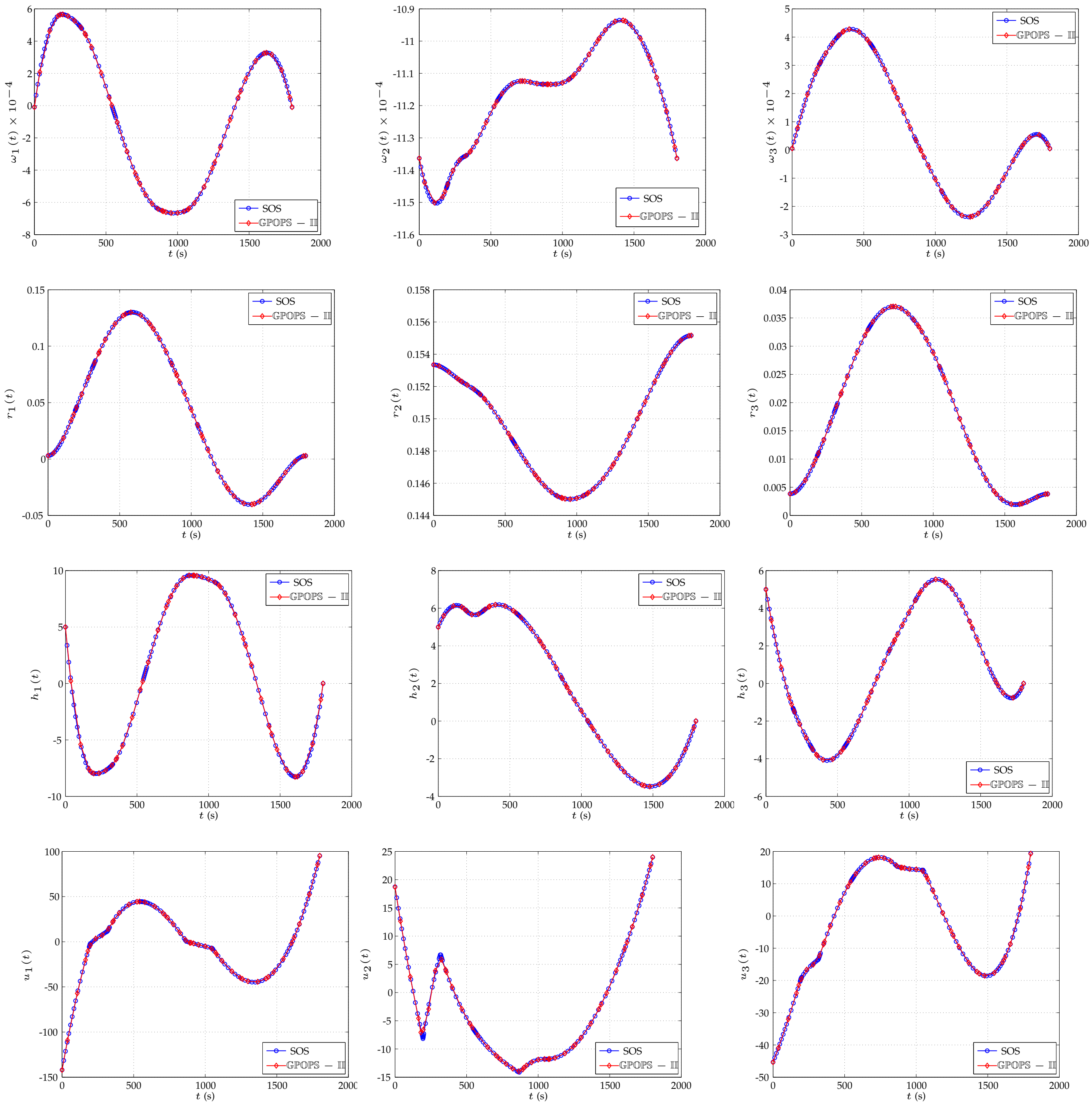
$$\mathbf{C} = \mathbf{I} + \frac{2}{1+\mathbf{r}^{\mathsf{T}}\mathbf{r}}\left(\mathbf{r}^{\otimes}\mathbf{r}^{\otimes} - \mathbf{r}^{\otimes}\right).$$

In this example the matrix $\mathbf{J}$ is given as

$$\mathbf{J} = \begin{bmatrix} 2.80701911616 \times 10^{7} & 4.822509936 \times 10^{5} & -1.71675094448 \times 10^{7} \\ 4.822509936 \times 10^{5} & 9.5144639344 \times 10^{7} & 6.02604448 \times 10^{4} \\ -1.71675094448 \times 10^{7} & 6.02604448 \times 10^{4} & 7.6594401336 \times 10^{7} \end{bmatrix},$$

$$\begin{aligned} \bar{\boldsymbol{\omega}}_0 &= \begin{bmatrix} -9.5380685844896 \times 10^{-6} \\ -1.1363312657036 \times 10^{-3} \\ +5.3472801108427 \times 10^{-6} \end{bmatrix}, \\ \bar{\mathbf{r}}_0 &= \begin{bmatrix} 2.9963689649816 \times 10^{-3} \\ 1.5334477761054 \times 10^{-1} \\ 3.8359805613992 \times 10^{-3} \end{bmatrix}, \\ \bar{\mathbf{h}}_0 &= \begin{bmatrix} 5000 \\ 5000 \\ 5000 \end{bmatrix}. \end{aligned}$$

Next: Closer Examination of the $\mathbb{GPOPS} - \mathbb{II}$ Code


$\omega_1(t) \times 10^{-4}$
$\omega_2(t) \times 10^{-4}$
$\omega_3(t) \times 10^{-4}$
$r_1(t)$
$r_2(t)$
$r_3(t)$
$h_1(t)$
$h_2(t)$
$h_3(t)$
$u_1(t)$
$u_2(t)$
$u_3(t)$
t (s)
SOS
GPOPS − II

Multiple-Stage Launch Vehicle Ascent

$$J = m(t_f^{(4)})$$

Subject to

$$\begin{aligned}
\dot{\mathbf{r}}^{(p)} &= \mathbf{v}^{(p)}, \\
\dot{\mathbf{v}}^{(p)} &= -\frac{\mu}{\|\mathbf{r}^{(p)}\|^3}\mathbf{r}^{(p)} + \frac{T^{(p)}}{m^{(p)}}\mathbf{u}^{(p)} + \frac{\mathbf{D}^{(p)}}{m^{(p)}}, \qquad (p = 1, \ldots, 4), \\
\dot{m}^{(p)} &= -\frac{T^{(p)}}{g_0 I_{sp}},
\end{aligned}$$

$$\begin{aligned}
\mathbf{r}(t_0) &= \mathbf{r}_0 = (5605.2, 0, 3043.4) \times 10^3 \text{ m}, \\
\mathbf{v}(t_0) &= \mathbf{v}_0 = (0, 0.4076, 0) \times 10^3 \text{ m/s}, \\
m(t_0) &= m_0 = 301454 \text{ kg}.
\end{aligned}$$

$$
\begin{aligned}
\mathbf{r}^{(p)}(t_f^{(p)}) - \mathbf{r}^{(p+1)}(t_0^{(p+1)}) &= \mathbf{0}, \\
\mathbf{v}^{(p)}(t_f^{(p)}) - \mathbf{v}^{(p+1)}(t_0^{(p+1)}) &= \mathbf{0}, \qquad (p = 1, \ldots, 3) \\
m^{(p)}(t_f^{(p)}) - m_{\text{dry}}^{(p)} - m^{(p+1)}(t_0^{(p+1)}) &= 0,
\end{aligned}
$$

$$
\begin{aligned}
a(t_f^{(4)}) &= a_f = 24361.14 \text{ km}, \\
e(t_f^{(4)}) &= e_f = 0.7308, \\
i(t_f^{(4)}) &= i_f = 28.5 \,\text{deg}, \\
\theta(t_f^{(4)}) &= \theta_f = 269.8 \,\text{deg}, \\
\phi(t_f^{(4)}) &= \phi_f = 130.5 \,\text{deg},
\end{aligned}
$$

$$
\begin{aligned}
\|\mathbf{r}^{(p)}\|_2^2 &\geq R_e, \\
\|\mathbf{u}^{(p)}\|_2^2 &= 1,
\end{aligned} \qquad (p = 1, \ldots, 4).
$$

Next: Closer Examination of the $\mathbb{GPOPS} - \mathbb{II}$ Code