A brief introduction to nocs

Daniel Cardona-Ortiz

Robotics and Advanced Manufacturing Group, Cinvestav, Mexico

April 15, 2021

This document presents the theoretical background of the algorithms implemented in the local collocation module of nocs (Numerical optimal control solver). The author suggest to verify (Betts) and (ICRA) for total understanding of the Gauss-Lobatto methods applied in this solver.

1 Optimal control problem for robotic systems

given an articulated rigid body system with n degrees of freedom (DoF) and a state vector

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} \in \mathbb{R}^{2n}, \tag{1}$$

where $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\} \in \mathbb{R}^n$ are the robot configuration, and joint velocities, respectively, find a control law (if exists) that leads the system from its initial state to a final one by solving:

$$\min_{\boldsymbol{u}(t)\in \boldsymbol{U}} \quad \phi(\boldsymbol{x}(0), \boldsymbol{x}(T), t_0, t_F) + \int_0^T \text{Ł}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt \tag{2}$$

$$\begin{aligned} \text{subject to} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \\ & \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \leq 0 \\ & \boldsymbol{x}_{lb} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{ub} \\ & \boldsymbol{x}(0) = \boldsymbol{x}_i \\ & \boldsymbol{x}(T) = \boldsymbol{x}_f \end{aligned} \tag{3}$$

where $\boldsymbol{U} \subset \mathbb{R}^n$ defines the admissible control domain, Ł$(\cdot)$ stands for the integrand of the path integral along the entire trajectory, $\phi(\cdot)$ is the boundary objective function, $\boldsymbol{h}(\cdot) \in \mathbb{R}^p$ are the path constraints, and the dynamics of the robot is expressed in its state equation form as

$$\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \begin{bmatrix} \dot{\boldsymbol{q}}(t) \\ \ddot{\boldsymbol{q}}(t) \end{bmatrix} \in \mathbb{R}^{2n}, \tag{4}$$

with

$$\ddot{\boldsymbol{q}}(t) = \boldsymbol{H}(\boldsymbol{q}(t))^{-1}(\boldsymbol{u}(t) - \boldsymbol{C}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t))\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{q}(t))) \tag{5}$$

where $\boldsymbol{H}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ denotes the inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^n$ contains the Coriolis and centrifugal forces, $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^n$ stands for the gravitational terms, and $\boldsymbol{u}$ is the generalized input torques. The constraint vectors $\boldsymbol{x}_{lb} \in \mathbb{R}^{2n}$ and $\boldsymbol{x}_{ub} \in \mathbb{R}^{2n}$ are the lower and upper bounds of the states of the system, and $\{\boldsymbol{x}_i, \boldsymbol{x}_f\} \in \mathbb{R}^{2n}$ are the initial and final states.

2 Direct collocation formulation

The use of direct collocation methods for the transcription of the optimal control problems allows to define the following numerical optimization problem:

$$\begin{aligned} \underset{\boldsymbol{z}}{\text{minimize}} \quad & \omega(\boldsymbol{z}) \\ \text{subject to} \quad & \boldsymbol{c}_L \leq \boldsymbol{c}(\boldsymbol{z}) \leq \boldsymbol{c}_U \\ & \boldsymbol{x}_{lb} \leq \boldsymbol{x}_k \leq \boldsymbol{x}_{ub}, \quad \forall k \in \mathbb{N} \\ & \boldsymbol{u}_{lb} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{ub}, \quad \forall k \in \mathbb{N} \end{aligned} \tag{6}$$

where $\boldsymbol{z} \in \mathbb{R}^{n_{decVar}}$ contains the NLP decision variables, $\boldsymbol{c}(\boldsymbol{z}) \in \mathbb{R}^{nCns}$ stands for the NLP constrains vector, and $\{\boldsymbol{c_L}, \boldsymbol{c_U}\} \in \mathbb{R}^{n_{cns}}$ are the lower and upper bounds of this vector. The structure and dimensions of this transcribed problem depends entirely of the collocation method applied.

The dynamics of the system are expressed as a finite group of collocation constraints

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\varphi}_0 \\ \boldsymbol{\varphi}_1 \\ \vdots \\ \boldsymbol{\varphi}_{N-1} \end{bmatrix} \in \mathbb{R}^{2n(N-1)} \tag{7}$$

with

$$\boldsymbol{\varphi}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k - h \sum_{i=1}^{s} \beta_i \boldsymbol{b}_i \tag{8}$$

where

$$\boldsymbol{b}_0 = \boldsymbol{x}_k \tag{9}$$

$$\boldsymbol{b}_i = \boldsymbol{f}\left(\boldsymbol{x}_k + h \sum_{j=1}^{s} a_{ij} \boldsymbol{b}_j, t_k + c_i h\right), \ i = 1, \ldots, s. \tag{10}$$

c_i, a_{ij} y β_i stands for the Butcher's array coefficient for a Runge-Kutta method of order s, and

$$\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k) = \begin{bmatrix} \dot{\boldsymbol{q}}_k \\ \boldsymbol{H}(\boldsymbol{q}_k)^{-1}(\boldsymbol{u}_k - \boldsymbol{h}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)) \end{bmatrix} \tag{11}$$

The path constraints are expressed as algebraic equations and are enforced at each collocation point

$$\boldsymbol{\gamma} = \begin{bmatrix} \boldsymbol{g}_0 \\ \boldsymbol{g}_1 \\ \vdots \\ \boldsymbol{g}_N \end{bmatrix} \in \mathbb{R}^{pN} \tag{12}$$

with

$$\boldsymbol{g}_k = \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k). \tag{13}$$

The boundary or event constraints are defined only at the the initial t_0 and final time t_f

$$\boldsymbol{e} = \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_N \end{bmatrix} \in \mathbb{R}^{4n} \tag{14}$$

Finally, the NLP constraints vector is defined as

$$\boldsymbol{c}(\boldsymbol{z}) = \begin{bmatrix} \boldsymbol{\Phi} \\ \gamma \\ \boldsymbol{e} \end{bmatrix} \in \mathbb{R}^{n_{cns}} \tag{15}$$

with its boundaries

$$\boldsymbol{c}_L = [\mathbf{0}\ \mathbf{0}\ ...\ \mathbf{0}\ \boldsymbol{g}_L\ \boldsymbol{g}_L\ ...\ \boldsymbol{g}_L\ \boldsymbol{x}_i\ \boldsymbol{x}_f]^T \in \mathbb{R}^{n_{cns}} \tag{16}$$

$$\boldsymbol{c}_U = [\mathbf{0}\ \mathbf{0}\ ...\ \mathbf{0}\ \boldsymbol{g}_U\ \boldsymbol{g}_U\ ...\ \boldsymbol{g}_U\ \boldsymbol{x}_i\ \boldsymbol{x}_f]^T \in \mathbb{R}^{n_{cns}} \tag{17}$$

3 Trapezoidal collocation method

3.1 Dimensional analysis of the trapezoidal collocation method

$$\begin{aligned} n_{decVar} &= N(n_s + n_u) + 2 \\ n_{cns} &= n_s(N-1) + pN + 4n_s \\ n_{colPoints} &= N \\ n_{colCns} &= n_s(N-1) \end{aligned}$$

3.2 Trapezoidal cost function

Recall the Bolza form of the cost function:

$$\phi(\boldsymbol{x}(0), \boldsymbol{x}(T), t_0, t_F) + \int_0^T \text{Ł}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt \tag{18}$$

To approximate the integral part (known as the Lagrange term) is possible to apply the trapezoidal quadrature

$$\int_0^T \text{Ł}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt = \sum_{k=0}^{N-1} \frac{h_k}{2}(\text{Ł}_{k+1} + \text{Ł}_k) \tag{19}$$

with $\text{Ł}_k \equiv \text{Ł}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k)$, $h_k = t_{k+1} - t_k$.

Extending the discrete approximation leads to:

$$
\begin{aligned}
\sum_{k=0}^{N-1} \frac{h_k}{2}(Ł_{k+1} + Ł_k) &= \frac{1}{2}(h_0Ł_0 + h_0Ł_1 + h_1Ł_1 + h_1Ł_2 + ... + h_{N-1}Ł_{N-1} + h_{N-1}Ł_N) \\
&= \frac{1}{2}(h_0Ł_0 + (h_0 + h_1)Ł_1 + (h_1 + h_2)Ł_2 + ... + (h_{N-2} + h_{N-1})Ł_{N-1} + h_{N-1}Ł_N) \\
&= \frac{1}{2}(w_0Ł_0 + w_1Ł_1 + w_1Ł_2 + ... + w_{N-1}Ł_{N-1} + w_NŁ_N)
\end{aligned}
$$

where w_k are known as the quadrature weights for the trapezoidal rule and are defined as:

$$
\begin{aligned}
w_0 &= h_0 \\
w_N &= h_{N-1} \\
w_k &= h_{k-1} + h_k \quad for \quad k = 1...(N-1)
\end{aligned}
$$

Let's now define $h_k = (\Delta t \Delta\tau_k) = (t_F - t_0)(\tau_{k+1} + \tau_k)$ where $0 \leq \tau_k \leq 1$ and $\Delta t \geq \Delta\tau_k$. Rewriting the quadrature using this definition leads to:

$$
\frac{\Delta t}{2}\,(\Delta\tau_0Ł_0 + (\Delta\tau_0 + \Delta\tau_1)Ł_1 + (\Delta\tau_1 + \Delta\tau_2)Ł_2 + ... + (\Delta\tau_{N-2} + \Delta\tau_{N-1})Ł_{N-1} + \Delta\tau_{N-1}Ł_N)
$$

Using some basic algebra is possible to rewrite the quadrature weights as:

$$
\begin{aligned}
w_0 &= \Delta\tau_0 \\
w_N &= \Delta\tau_{N-1} \\
w_k &= \tau_{k+2} - \tau_k \quad for \quad k = 0...(N-2)
\end{aligned}
$$

This allows to define the quadrature that nocs applies to compute the Lagrange term approximation for the trapezoidal collocation method

$$
\int_0^T Ł(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt = \frac{1}{2}(w_0Ł_0 + w_NŁ_N + \sum_{k=1}^{N-1} w_kŁ_k) \tag{20}
$$

3.2.1 Analytical gradients of the trapezoidal quadrature on nocs

Recall the extended approximation applied in nocs for the trapezoidal quadrature

$$
L = \frac{\Delta t}{2}(w_0Ł_0 + w_1Ł_1 + w_1Ł_2 + ... + w_{N-1}Ł_{N-1} + w_NŁ_N) \tag{21}
$$

and define the structure of the gradient of this function w.r.t to the NLP decision variables $\boldsymbol{z} \in \mathbb{R}^{n_{decVar}}$ as:

$$
\nabla_{\boldsymbol{z}} L = \begin{bmatrix} \nabla_{t_0} L & \nabla_{t_F} L & \nabla_{x_0} L & \nabla_{u_0} L & \dots & \nabla_{x_N} L & \nabla_{u_N} L \end{bmatrix}^T \in \mathbb{R}^{n_{decVar}}
$$

Then, the gradients of the trapezoidal w.r.t. the initial (t_0) and final times (t_F) are defined as:

$$\begin{aligned}
\nabla_{t_0} L &= \frac{\mathrm{d}L}{\mathrm{d}t_0} = -\frac{1}{2}(w_0 \text{Ł}_0 + w_N \text{Ł}_N + \sum_{k=1}^{N-1} w_k \text{Ł}_k) \in \mathbb{R} && (22)\\
\nabla_{t_F} L &= \frac{\mathrm{d}L}{\mathrm{d}t_F} = -\nabla_{t_0} L \in \mathbb{R} && (23)
\end{aligned}$$

In the other hand, the gradients w.r.t. the states $\boldsymbol{x}_k$ and control variables $\boldsymbol{u}_k$ can be easily computed by taking advantage of the local dependency of the function Ł_k

$$\begin{aligned}
\nabla_{\boldsymbol{x}_k} L &= \frac{w_k}{2}(\frac{\partial \text{Ł}_k}{\partial \boldsymbol{x}_k}) \in \mathbb{R}^{n_s} && (24)\\
\nabla_{\boldsymbol{u}_k} L &= \frac{w_k}{2}(\frac{\partial \text{Ł}_k}{\partial \boldsymbol{u}_k}) \in \mathbb{R}^{n_u} && (25)
\end{aligned}$$

3.3 Constrained system

Let us define the following functions, called in the literature as the trapezoidal constrained system:

$$\begin{aligned}
\boldsymbol{\zeta_k} &= \boldsymbol{x}_{k+1} - \boldsymbol{x}_k - \frac{h_k}{2}(\boldsymbol{f}_{k+1} + \boldsymbol{f}_k) \in \mathbb{R}^{n_s} && (26)\\
\boldsymbol{\zeta_k} &= \boldsymbol{x}_{k+1} - \boldsymbol{x}_k - \frac{(t_F - t_0)(\tau_{k+1} - \tau_k)}{2}(\boldsymbol{f}_{k+1} + \boldsymbol{f}_k) && (27)\\
\boldsymbol{\zeta_k} &= \boldsymbol{x}_{k+1} - \boldsymbol{x}_k - \frac{\Delta t \Delta \tau_k}{2}(\boldsymbol{f}_{k+1} + \boldsymbol{f}_k) && (28)\\
\boldsymbol{\zeta_k} &= [\boldsymbol{x}_{k+1} - \boldsymbol{x}_k] - [\frac{\Delta \tau_k}{2}(\Delta t \boldsymbol{f}_k)] - [\frac{\Delta \tau_k}{2}(\Delta t \boldsymbol{f}_{k+1})] && (29)
\end{aligned}$$

Using the equation (29) the constrains vector (15) can be rewritten as

$$\boldsymbol{Az} + \boldsymbol{By(z)} \tag{30}$$

where the dimensions of two constant matrices A and B and the non-linear vector $\boldsymbol{y(z)}$ are
$\boldsymbol{A} \in \mathbb{R}^{n_{cns} \times n_{decVar}}$
$\boldsymbol{B} \in \mathbb{R}^{n_{cns} \times (n_{colPoints})(n_s+p))+n_e+1}$
$\boldsymbol{y(z)} \in \mathbb{R}^{(n_{colPoints})(n_s+p))+4n_s+1}$

Under some analysis, is straightforward to determine that the structure of the constant matrices are

$$\boldsymbol{A} = \begin{bmatrix}
\boldsymbol{0}_{n_s\times 2} & -\boldsymbol{I}_{n_s\times n_s} & \boldsymbol{0}_{n_s\times n_u} & \boldsymbol{I}_{n_s\times n_s} & & & & & \\
 & & & -\boldsymbol{I}_{n_s\times n_s} & \boldsymbol{0}_{n_s\times n_u} & \boldsymbol{I}_{n_s\times n_s} & & & \\
\vdots & & & & & & \ddots & & \\
\boldsymbol{0}_{n_s\times 2} & & & & & & -\boldsymbol{I}_{n_s\times 2} & \boldsymbol{0}_{n_s\times n_u} & \boldsymbol{I}_{n_s\times n_s} \\
 & & & & & & & &
\end{bmatrix} \tag{31}$$

$$
\boldsymbol{B} = -\frac{1}{2}\begin{bmatrix}
\Delta\tau_0 \boldsymbol{I}_{n_s\times n_s} & \Delta\tau_0 \boldsymbol{I}_{n_s\times n_s} & & & & \\
 & \Delta\tau_1 \boldsymbol{I}_{n_s\times n_s} & \Delta\tau_1 \boldsymbol{I}_{n_s\times n_s} & & & \\
 & & \ddots & & & \\
 & & & \Delta\tau_{(N-1)} \boldsymbol{I}_{n_s\times n_s} & \Delta\tau_{(N-1)} \boldsymbol{I}_{n_s\times n_s} & \\
 & & & & & \boldsymbol{I}_{(pN+n_e)\times(pN+n_e)}
\end{bmatrix} \tag{32}
$$

and the non-linear vector is defined as:

$$
\boldsymbol{y}(\boldsymbol{z}) = \begin{bmatrix} \bar{\boldsymbol{f}} \\ \gamma \\ \boldsymbol{e} \end{bmatrix} \tag{33}
$$

where $\bar{\boldsymbol{f}} = \begin{bmatrix} \Delta t \boldsymbol{f}_1^T & \Delta t \boldsymbol{f}_2^T & \Delta t \boldsymbol{f}_3^T & \cdots & \Delta t \boldsymbol{f}_N^T \end{bmatrix}^T \in \mathbb{R}^{n_s N}$

3.4 The propagation algorithm for the construction of A and B

For simplicity, let us define a block matrix as

$$
\boldsymbol{A}_k = \begin{bmatrix} -\boldsymbol{I}_{n_s\times n_s} & \boldsymbol{0}_{n_s\times n_u} & \boldsymbol{I}_{n_s\times n_s} \end{bmatrix} \in \mathbb{R}^{n_s\times(2n_s+n_u)} \tag{34}
$$

And its triplet $a \in \mathbb{R}^{3\times a_{nz}}$representation can be defined as

Row	Column	Value
i	j	-1
i+1	j+1	-1
$\vdots$	$\vdots$	$\vdots$
i+n_s	j+n_s	-1
i	j+ n_s+n_u	1
i+1	j+n_s+n_u+1	1
$\vdots$	$\vdots$	$\vdots$
i+n_s	j+2n_s+n_u	1

where $a_{nz} = 2n_s$ is the number of non-zero elements in the block matrix A_k.

To construct the matrix (31) is necessary to insert the matrix A_k, $N-1$ times, by deduction the number of non-zero elements in the matrix A can be determined as

$$
A_{nz} = a_{nz}(N-1) \tag{35}
$$

Following the well structured pattern of the A matrix is possible to propagate the triplet a through the $N-1$ points and obtain a close form to the sparsity template of any A matrix independently of the size of the problem:
with $\alpha = (k)(n_s)$ and $\beta = (k)(n_s+n_u)$ for $k = 0...N-1$

Row	Column	Value
α	$\beta+2$	-1
$\alpha+1$	$\beta+3$	-1
$\vdots$	$\vdots$	$\vdots$
α+n_s	$\beta+n_s+2$	-1
α	$\beta+n_s+n_u+2$	1
$\alpha+1$	$\beta+n_s$+n_u+1+2	1
$\vdots$	$\vdots$	$\vdots$
$\alpha+n_s$	$\beta+2n_s$+n_u+2	1

Algorithm 1 Propagation algorithm for A trapezoidal matrix

```
1: for $k = 0, 1, \ldots, N-1$ do
2:     $i = 0$
3:     $\alpha = (k)(n_s)$
4:     $\beta = (k)(n_s + n_u) + 2$
5:     for $n_z = 0, 1, \ldots, n_s$ do
6:         $iRow = \alpha + i$
7:         $iCol = \beta + i$
8:         Save the information in the triplet ($iRow$, $iCol$, $-1$)
9:         i++
10:    end for
11:    $i = 0$
12:    for $n_z = n_s, n_s+1, \ldots, a_{nz}$ do
13:        $iRow = \alpha + i$
14:        $iCol = \beta + n_s + n_u + i$
15:        Save the information in the triplet ($iRow$, $iCol$, 1)
16:        i++
17:    end for
18: end for
```

Now for simplicity define a block matrix

$$\boldsymbol{B}_k = \frac{\Delta\tau_k}{2}\begin{bmatrix}\boldsymbol{I}_{n_s\times n_s} & \boldsymbol{I}_{n_s\times n_s}\end{bmatrix} \in \mathbb{R}^{n_s\times 2n_s} \tag{36}$$

Following the methodology applied for the matrix A_k, the triplet $b \in \mathbb{R}^{3\times b_{nz}}$ (where $b_{nz} = 2n_s$) can be defined as:

To construct the matrix B, is necessary to insert the B_k matrix $N-1$ times and an identity matrix $I_g \in \mathbb{R}^{(pN+n_e)\times(pN+n_e)}$. The number of non-zero elements in the B matrix can be computed as:

$$B_{nz} = (2n_s)(N-1) + pN + n_e \tag{37}$$

Following the pattern of the B matrix is possible to propagate the triplet b through the $N-1$ points and insert the I_g matrix to obtain a close form sparsity template of any B matrix, independently of the size of the problem:
with $\eta = \nu = (k)(n_s)$ for $k = 0...N-1$

Algorithm 2 Propagation algorithm for B trapezoidal matrix

```
1: for $k = 0, 1, \ldots, N-1$ do
2:     $i = 0$
3:     $\eta = (k)(n_s)$
4:     $\nu = (k)(n_s)$
5:     for $n_z = 0, 1, \ldots, n_s$ do
6:         $iRow = \eta + i$
7:         $iCol = \nu + i$
8:         Save the information in the triplet ($iRow$, $iCol$, $-0.5\Delta\tau_k$)
9:         i++
10:    end for
11:    $i = 0$
12:    for $n_z = n_s, n_s + 1, \ldots, 2n_s$ do
13:        $iRow = \eta + i$
14:        $iCol = \nu + n_s + i$
15:        Save the information in the triplet ($iRow$, $iCol$, $-0.5\Delta\tau_k$)
16:        i++
17:    end for
18: end for
19: $i = 0$
20: for $n_z = (2n_s)(N-1), \ldots, B_{nz}$ do
21:     $iRow = n_{colCns} + i$
22:     $iCol = (n_s)(n_{colPoints}) + i$
23:     Save the information in the triplet ($iRow$, $iCol$, 1)
24:     i++
25: end for
```

Row	Column	Value
i	j	$-\frac{\Delta\tau_k}{2}$
i+1	j+1	$-\frac{\Delta\tau_k}{2}$
$\vdots$	$\vdots$	$\vdots$
i+n_s	j+n_s	$-\frac{\Delta\tau_k}{2}$
i	j+n_s	$-\frac{\Delta\tau_k}{2}$
i+1	j+n_s+1	$-\frac{\Delta\tau_k}{2}$
$\vdots$	$\vdots$	$\vdots$
i+n_s	j+2n_s	$-\frac{\Delta\tau_k}{2}$

Row	Column	Value
η	$\nu + 2$	$-\frac{\Delta\tau_k}{2}$
$\eta + 1$	$\nu + 1$	$-\frac{\Delta\tau_k}{2}$
$\vdots$	$\vdots$	$\vdots$
η+n_s	$\nu + n_s$	$-\frac{\Delta\tau_k}{2}$
η	$\nu + n_s + n_u + 2$	$-\frac{\Delta\tau_k}{2}$
$\eta + 1$	$\nu + n_s$+$n_u + 1 + 2$	$-\frac{\Delta\tau_k}{2}$
$\vdots$	$\vdots$	$\vdots$
$\eta + n_s$	$\nu + 2n_s$+n_u	$-\frac{\Delta\tau_k}{2}$
$\vdots$	$\vdots$	$\vdots$
n_{colCns}	$(n_s)(n_{colPoints})$	1
$n_{colCns} + 1$	$(n_s)(n_{colPoints}) + 1$	1
$\vdots$	$\vdots$	$\vdots$
$n_{colCns} + nP + n_e$	$(n_s)(n_{colPoints}) + nP + n_e$	1

4 Hermite-Simpson Separated (HSS) collocation method

4.1 Dimensional analysis of the HSS collocation method

$$n_{colPoints} = 2N - 1$$
$$n_{decVar} = (n_{colPoints})(n_s + n_u) + 2$$
$$n_{colCns} = n_s(n_{colPoints} - 1)$$
$$n_{cns} = n_{colCns} + p(n_{colPoints}) + n_e$$

4.2 Hermite-Simpson cost function

Recall the Bolza form of the cost function:

$$\phi(\boldsymbol{x}(0), \boldsymbol{x}(T), t_0, t_F) + \int_0^T \text{Ł}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt \tag{38}$$

To approximate the integral part (known as the Lagrange term) is possible to apply the trapezoidal quadrature

$$\int_0^T \text{Ł}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt = \sum_{k=0}^{N-1} \frac{h_k}{6}(\text{Ł}_{k+1} + 4\bar{\text{Ł}}_k + \text{Ł}_k) \tag{39}$$

with $\text{Ł}_k \equiv \text{Ł}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k)$,$\bar{\text{Ł}}_k \equiv \text{Ł}(\boldsymbol{x}_{k+\frac{1}{2}}, \boldsymbol{u}_{k+\frac{1}{2}}, t_{k+\frac{1}{2}})$, $h_k = t_{k+1} - t_k$.

Extending the discrete approximation leads to:

$$\begin{aligned}
\sum_{k=0}^{N-1} \frac{h_k}{6}(\text{Ł}_k + 4\bar{\text{Ł}}_k + \text{Ł}_{k+1}) &= \frac{1}{6}(h_0\text{Ł}_0 + 4h_0\bar{\text{Ł}}_0 + h_0\text{Ł}_1 + h_1\text{Ł}_1 + 4h_1\bar{\text{Ł}}_1 + h_1\text{Ł}_2 + ... \\
&... +h_{N-1}\text{Ł}_{N-1} + 4h_{N-1}\bar{\text{Ł}}_{N-1} + h_{N-1}\text{Ł}_N) \\
&= \frac{1}{6}(h_0\text{Ł}_0 + 4h_0\bar{\text{Ł}}_0 + (h_0 + h_1)\text{Ł}_1 + 4h_1\bar{\text{Ł}}_1 + (h_1 + h_2)\text{Ł}_2 + ... \\
&... +(h_{N-2} + h_{N-1})\text{Ł}_{N-1} + 4h_{N-1}\bar{\text{Ł}}_{N-1} + h_{N-1}\text{Ł}_N) \\
&= \frac{1}{6}(w_0\text{Ł}_0 + \bar{w}_0\bar{\text{Ł}}_0 + w_1\text{Ł}_1 + \bar{w}_1\bar{\text{Ł}}_1 + w_2\text{Ł}_2 + ... \\
&... +w_{N-1}\text{Ł}_{N-1} + \bar{w}_{N-1}\bar{\text{Ł}}_{N-1} + w_N\text{Ł}_N)
\end{aligned}$$

where w_k are known as the quadrature weights for the Simpson rule and are defined as:

$$\begin{aligned}
w_0 &= h_0 \\
w_N &= h_{N-1} \\
\bar{w}_k &= 4h_k \qquad for \quad k = 0...(N-1) \\
w_k &= h_{k-1} + h_k \quad for \quad k = 1...(N-1)
\end{aligned}$$

Let's now define $h_k = (\Delta t \Delta\tau_k) = (t_F - t_0)(\tau_{k+1} + \tau_k)$ where $0 \leq \tau_k \leq 1$ and $\Delta t \geq \Delta\tau_k$. Rewriting the quadrature using this definition leads to:

$$\begin{aligned}
w_0 &= \Delta\tau_0 \\
w_N &= \Delta\tau_{N-1} \\
\bar{w}_k &= 4\Delta\tau_k \qquad for \quad k = 0...(N-1) \\
w_k &= \tau_{k+2} - \tau_k \quad for \quad k = 0...(N-2)
\end{aligned}$$

This allows to define the quadrature that NOCS applies to compute the Lagrange term approximation for the trapezoidal collocation method

$$\int_0^T \text{Ł}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt = \frac{1}{6}(w_0\text{Ł}_0 + w_N\text{Ł}_N + \sum_{k=0}^{N-1} \bar{w}_k\bar{\text{Ł}}_k + \sum_{k=1}^{N-1} w_k\text{Ł}_k) \tag{40}$$

4.2.1 Analytical gradients of the Simspon quadrature on NOCS

Recall the extended approximation applied in NOCS for the Simpson quadrature

$$L = \frac{1}{6}(w_0\text{Ł}_0 + \bar{w}_0\bar{\text{Ł}}_0 + w_1\text{Ł}_1 + \bar{w}_1\bar{\text{Ł}}_1 + w_2\text{Ł}_2 + ... + w_{N-1}\text{Ł}_{N-1} + \bar{w}_{N-1}\bar{\text{Ł}}_{N-1} + w_N\text{Ł}_N) \tag{41}$$

and define the structure of the gradient $\nabla_{\boldsymbol{z}} L \in \mathbb{R}^{n_{decVar}}$ of this function w.r.t to the NLP decision variables $\boldsymbol{z} \in \mathbb{R}^{n_{decVar}}$ as:

$$\nabla_{\boldsymbol{z}} L = \left[\nabla_{t_0} L \quad \nabla_{t_F} L \quad \nabla_{x_0} L \quad \nabla_{u_0} L \quad \nabla_{x_{0.5}} L \quad \nabla_{u_{0.5}} L \quad \dots \quad \nabla_{x_{N-\frac{1}{2}}} L \quad \nabla_{u_{N-\frac{1}{2}}} L \quad \nabla_{x_N} L \quad \nabla_{u_N} L\right]^T$$

Then, the gradients of the Simpson quadrature w.r.t. the initial (t_0) and final times (t_F) are defined as:

$$\nabla_{t_0} L = \frac{\mathrm{d}L}{\mathrm{d}t_0} = -\frac{1}{6}(w_0 Ł_0 + w_N Ł_N + \sum_{k=1}^{N-1} w_k Ł_k) \in \mathbb{R} \tag{42}$$

$$\nabla_{t_F} L = \frac{\mathrm{d}L}{\mathrm{d}t_F} = -\nabla_{t_0} L \in \mathbb{R} \tag{43}$$

In the other hand, the gradients w.r.t. the states $\boldsymbol{x}_k$ and control variables $\boldsymbol{u}_k$ can be easily computed by taking advantage of the local dependency of the function $Ł_k$

$$\nabla_{\boldsymbol{x}_k} L = \frac{w_k}{6}(\frac{\partial Ł_k}{\partial \boldsymbol{x}_k}) \in \mathbb{R}^{n_s} \tag{44}$$

$$\nabla_{\boldsymbol{u}_k} L = \frac{w_k}{6}(\frac{\partial Ł_k}{\partial \boldsymbol{u}_k}) \in \mathbb{R}^{n_u} \tag{45}$$

4.3 Constrained system

Let us define the following functions, called in the literature as the Hermite-Simpson constrained system:

$$\boldsymbol{\zeta_{k,0}} = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k - \frac{h_k}{6}(\boldsymbol{f}_k + 4\boldsymbol{f}_{k+\frac{1}{2}} + \boldsymbol{f}_{k+1}) \in \mathbb{R}^{n_s} \tag{46}$$

$$\boldsymbol{\zeta_{k,1}} = \boldsymbol{x}_{k+\frac{1}{2}} - \frac{1}{2}(\boldsymbol{x}_k + \boldsymbol{x}_{k+1}) - \frac{h_k}{8}(\boldsymbol{f}_k - \boldsymbol{f}_{k+1}) \in \mathbb{R}^{n_s} \tag{47}$$

where $\boldsymbol{\zeta_{k,0}}$ is known as the Simpson's quadrature and $\boldsymbol{\zeta_{k,1}}$ is the well known Hermite's interpolant.

By rearranging the HSS constrained system

$$\begin{aligned}\boldsymbol{\zeta_{k,0}} &= \boldsymbol{x}_{k+1} - \boldsymbol{x}_k - \frac{\Delta\tau_k \Delta t}{6}(\boldsymbol{f}_k + 4\boldsymbol{f}_{k+\frac{1}{2}} + \boldsymbol{f}_{k+1}) \\ \boldsymbol{\zeta_{k,1}} &= \boldsymbol{x}_{k+\frac{1}{2}} - \frac{1}{2}(\boldsymbol{x}_k + \boldsymbol{x}_{k+1}) - \frac{\Delta\tau_k \Delta t}{8}(\boldsymbol{f}_k - \boldsymbol{f}_{k+1})\end{aligned}$$

$$\boldsymbol{\zeta_{k,0}} = [\boldsymbol{x}_{k+1} - \boldsymbol{x}_k] - [\frac{\Delta\tau_k \Delta t}{6}\boldsymbol{f}_k] - [\frac{2\Delta\tau_k \Delta t}{3}\boldsymbol{f}_{k+\frac{1}{2}}] - [\frac{\Delta\tau_k \Delta t}{6}\boldsymbol{f}_{k+1}] \tag{48}$$

$$\boldsymbol{\zeta_{k,1}} = [\boldsymbol{x}_{k+\frac{1}{2}} - \frac{1}{2}\boldsymbol{x}_k - \frac{1}{2}\boldsymbol{x}_{k+1}] - [\frac{\Delta\tau_k \Delta t}{8}\boldsymbol{f}_k] + [\frac{\Delta\tau_k \Delta t}{8}\boldsymbol{f}_{k+1}] \tag{49}$$

Following the same procedure as the trapezoidal method, is possible to reconstruct the HSS constrained system as

$$\boldsymbol{A}\boldsymbol{z} + \boldsymbol{B}\boldsymbol{y}(\boldsymbol{z}) \tag{50}$$

where the dimensions of two constant matrices A and B and the non-linear vector $\boldsymbol{y}(\boldsymbol{z})$ are
$\boldsymbol{A} \in \mathbb{R}^{n_{cns} \times n_{decVar}}$
$\boldsymbol{B} \in \mathbb{R}^{n_{cns} \times (n_{colPoints})(n_s+p))+n_e+1}$
$\boldsymbol{y}(\boldsymbol{z}) \in \mathbb{R}^{(n_{colPoints})(n_s+p))+n_e+1}$

Under the same analysis, the structure of the constant matrices is

$$\boldsymbol{A} = \begin{bmatrix}
\mathbf{0}_{n_s\times 2} & -\frac{1}{2}\boldsymbol{I} & \mathbf{0} & \boldsymbol{I} & \mathbf{0} & -\frac{1}{2}\boldsymbol{I} & & & & & & & & & & \\
\mathbf{0}_{n_s\times 2} & -\boldsymbol{I} & \mathbf{0} & & \mathbf{0} & \boldsymbol{I} & & & & & & & & & & \\
 & & & & & -\frac{1}{2}\boldsymbol{I} & \mathbf{0} & \boldsymbol{I} & \mathbf{0} & -\frac{1}{2}\boldsymbol{I} & & & & & & \\
 & & & & & -\boldsymbol{I} & \mathbf{0} & & \mathbf{0} & \boldsymbol{I} & & & & & & \\
 & & & & & & & & & & \ddots & & & & & \\
 & & & & & & & & & & & -\frac{1}{2}\boldsymbol{I} & \mathbf{0} & \boldsymbol{I} & \mathbf{0} & -\frac{1}{2}\boldsymbol{I} \\
 & & & & & & & & & & & -\boldsymbol{I} & \mathbf{0} & & \mathbf{0} & \boldsymbol{I}
\end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix}
-\frac{\Delta\tau_0}{8}\boldsymbol{I} & \mathbf{0}_{n_s} & \frac{\Delta\tau_0}{8}\boldsymbol{I} & & & & & & \\
-\frac{\Delta\tau_0}{6}\boldsymbol{I} & -\frac{2\Delta\tau_0}{3}\boldsymbol{I} & -\frac{\Delta\tau_0}{6}\boldsymbol{I} & & & & & & \\
 & & -\frac{\Delta\tau_1}{8}\boldsymbol{I} & \mathbf{0}_{n_s} & \frac{\Delta\tau_1}{8}\boldsymbol{I} & & & & \\
 & & -\frac{\Delta\tau_1}{6}\boldsymbol{I} & -\frac{2\Delta\tau_1}{3}\boldsymbol{I} & -\frac{\Delta\tau_1}{6}\boldsymbol{I} & & & & \\
 & & & & & \ddots & & & \\
 & & & & & & -\frac{\Delta\tau_{N-1}}{8}\boldsymbol{I} & \mathbf{0}_{n_s} & \frac{\Delta\tau_{(N-1)}}{8}\boldsymbol{I} \\
 & & & & & & -\frac{\Delta\tau_{(N-1)}}{6}\boldsymbol{I} & -\frac{2\Delta\tau_{N-1}}{3}\boldsymbol{I} & -\frac{\Delta\tau_{(N-1)}}{6}\boldsymbol{I}
\end{bmatrix}$$

and the non-linear vector is defined as:

$$\boldsymbol{y}(\boldsymbol{z}) = \begin{bmatrix} \bar{\boldsymbol{f}} \\ \gamma \\ \boldsymbol{e} \end{bmatrix} \tag{51}$$

where $\bar{\boldsymbol{f}} = \begin{bmatrix} \Delta t \boldsymbol{f}_1^T & \Delta t \boldsymbol{f}_{1.5}^T & \Delta t \boldsymbol{f}_2^T & \Delta t \boldsymbol{f}_{2.5}^T & \Delta t \boldsymbol{f}_3^T & \Delta t \boldsymbol{f}_{3.5}^T & \cdots & \Delta t \boldsymbol{f}_N^T \end{bmatrix}^T \in \mathbb{R}^{n_s(n_{colPoints})}$, $\boldsymbol{I} \in \mathbb{R}^{n_s \times n_s}$ is an identity matrix, $\mathbf{0} \in \mathbb{R}^{n_s \times n_u}$ and $\mathbf{0}_{ns} \in \mathbb{R}^{n_s \times n_s}$ are null matrices.

4.4 The propagation algorithm for the construction of A and B

For simplicity, let us define a block matrix as

$$\boldsymbol{A_k} = \begin{bmatrix} -\frac{1}{2}\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & -\frac{1}{2}\boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{0} & & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \in \mathbb{R}^{2n_s \times (3n_s+2n_u)} \tag{52}$$

And its triplet $a \in \mathbb{R}^{3\times a_{nz}}$representation can be defined as

Row	Column	Value
i	j	-0.5
i+1	j+1	-0.5
$\vdots$	$\vdots$	$\vdots$
i+n_s	j+n_s	-0.5
i	j+ n_s+n_u	1
i+1	j+n_s+n_u+1	1
$\vdots$	$\vdots$	$\vdots$
i+n_s	j+2n_s+n_u	1
i	j+2n_s + 2n_u	-0.5
i+1	j+2n_s + 2n_u+1	-0.5
$\vdots$	$\vdots$	$\vdots$
i+n_s	j+3n_s+2n_u	-0.5
i+ n_s	j	-1
i+n_s+1	j+1	-1
$\vdots$	$\vdots$	$\vdots$
i+2n_s	j+n_s	-1
i+ n_s	j+2n_s+2n_u	1
i+n_s+1	j+2n_s+2n_u+1	1
$\vdots$	$\vdots$	$\vdots$
i+2n_s	j+3n_s+2n_u	1

where $a_{nz} = 5n_s$ is the number of non-zero elements in the block matrix A_k.

To construct the matrix A is necessary to insert the matrix A_k, $N-1$ times, by deduction the number of non-zero elements in the matrix A can be determined as

$$A_{nz} = 5n_s(N-1) \tag{53}$$

Once again, following the pattern of the A matrix is possible to propagate the triplet a through the $N-1$ points and obtain a close form to the sparsity template of any A matrix independently of the size of the problem:

Row	Column	Value
α	$\beta + 2$	-0.5
$\alpha +1$	$\beta+3$	-0.5
$\vdots$	$\vdots$	$\vdots$
$\alpha+n_s$	$\beta+n_s + 2$	-0.5
α	$\beta+ n_s+n_u + 2$	1
$\alpha+1$	$\beta+n_s+n_u+3$	1
$\vdots$	$\vdots$	$\vdots$
$\alpha+n_s$	$\beta+2n_s+n_u+2$	1
α	$\beta+2n_s + 2n_u + 2$	-0.5
$\alpha+1$	$\beta+2n_s + 2n_u+3$	-0.5
$\vdots$	$\vdots$	$\vdots$
$\alpha+n_s$	$\beta+3n_s+2n_u+2$	-0.5
$\alpha+ n_s$	$\beta + 2$	-1
$\alpha+n_s+1$	$\beta+3$	-1
$\vdots$	$\vdots$	$\vdots$
$\alpha+2n_s$	$\beta+n_s+2$	-1
$\alpha+ n_s$	$\beta+2n_s+2n_u+2$	1
$\alpha+n_s+1$	$\beta+2n_s+2n_u+3$	1
$\vdots$	$\vdots$	$\vdots$
$\alpha+2n_s$	$\beta+3n_s+2n_u+2$	1

with $\alpha = (2k)(n_s)$ and $\beta = (k)(2(n_s + n_u))$ for $k = 0...N - 1$

Algorithm 3 Propagation algorithm for A trapezoidal matrix

```
for $k = 0, 1, \dots, N-1$ do
    $i = 0$
    $\alpha = (2k)(n_s)$
    $\beta = (k)(2(n_s + n_u)) + 2$
    for $n_z = 0, 1, \dots, n_s$ do
        $iRow = \alpha + i$
        $iCol = \beta + i$
        Save the information in the triplet ($iRow$, $iCol$, $-0.5$)
        i++
    end for
    $i = 0$
    for $n_z = n_s, n_s + 1, \dots, 2n_s$ do
        $iRow = \alpha + i$
        $iCol = \beta + n_s + n_u + i$
        Save the information in the triplet ($iRow$, $iCol$, 1)
        i++
    end for
    $i = 0$
    for $n_z = 2n_s, 2n_s + 1, \dots, 3n_s$ do
        $iRow = \alpha + i$
        $iCol = \beta + 2n_s + 2n_u + i$
        Save the information in the triplet ($iRow$, $iCol$, $-0.5$)
        i++
    end for
    $i = 0$
    for $n_z = 3n_s, 3n_s + 1, \dots, 4n_s$ do
        $iRow = \alpha + n_s + i$
        $iCol = \beta + i$
        Save the information in the triplet ($iRow$, $iCol$, $-1$)
        i++
    end for
    $i = 0$
    for $n_z = 4n_s, 4n_s + 1, \dots, a_{nz}$ do
        $iRow = \alpha + n_s + i$
        $iCol = \beta + 2n_s + 2n_u + i$
        Save the information in the triplet ($iRow$, $iCol$, 1)
        i++
    end for
end for
```

Now for simplicity define a block matrix

$$\boldsymbol{B}_k = \begin{bmatrix} -\frac{\Delta\tau_k}{8}\boldsymbol{I} & \boldsymbol{0}_{n_s} & \frac{\Delta\tau_k}{8}\boldsymbol{I} \\ -\frac{\Delta\tau_k}{6}\boldsymbol{I} & -\frac{2\Delta\tau_k}{3}\boldsymbol{I} & -\frac{\Delta\tau_k}{6}\boldsymbol{I} \end{bmatrix}$$

Following the methodology applied for the matrix A_k, the triplet $b \in \mathbb{R}^{3\times b_{nz}}$ (where $b_{nz} = 5n_s$) can be defined as:

Row	Column	Value
i	j	$-\frac{\Delta\tau_k}{8}$
i+1	j+1	$-\frac{\Delta\tau_k}{8}$
$\vdots$	$\vdots$	$\vdots$
i+n_s	j+n_s	$-\frac{\Delta\tau_k}{8}$
i	j+2n_s	$\frac{\Delta\tau_k}{8}$
i+1	j+2n_s+1	$\frac{\Delta\tau_k}{8}$
$\vdots$	$\vdots$	$\vdots$
i+n_s	j+3n_s	$\frac{\Delta\tau_k}{8}$
i+ n_s	j	$-\frac{\Delta\tau_k}{6}$
i+n_s+1	j+1	$-\frac{\Delta\tau_k}{6}$
$\vdots$	$\vdots$	$\vdots$
i+2n_s	j+n_s	$-\frac{\Delta\tau_k}{6}$
i+ n_s	j+n_s	$-\frac{2\Delta\tau_k}{3}$
i+n_s+1	j+n_s+1	$-\frac{2\Delta\tau_k}{3}$
$\vdots$	$\vdots$	$\vdots$
i+2n_s	j+2n_s	$-\frac{2\Delta\tau_k}{3}$
i+ n_s	j+2n_s	$-\frac{\Delta\tau_k}{6}$
i+n_s+1	j+2n_s+1	$-\frac{\Delta\tau_k}{6}$
$\vdots$	$\vdots$	$\vdots$
i+2n_s	j+3n_s	$-\frac{\Delta\tau_k}{6}$

To construct the matrix B, is necessary to insert the B_k matrix $N-1$ times and an identity matrix $I_g \in \mathbb{R}^{(p(n_{colPoints})+n_e)\times(pN+n_e)}$. The number of non-zero elements in the B matrix can be computed as:

$$B_{nz} = (5n_s)(N-1) + p(n_{colPoints}) + n_e \tag{54}$$

Following the pattern of the B matrix is possible to propagate the triplet b through the $N-1$ points and insert the I_g matrix to obtain a close form sparsity template of any B matrix, independently of the size of the problem:

Row	Column	Value
η	ν	$-\frac{\Delta\tau_k}{8}$
η+1	ν+1	$-\frac{\Delta\tau_k}{8}$
$\vdots$	$\vdots$	$\vdots$
η+n_s	ν+n_s	$-\frac{\Delta\tau_k}{8}$
η	ν+2n_s	$\frac{\Delta\tau_k}{8}$
η+1	ν+2n_s+1	$\frac{\Delta\tau_k}{8}$
$\vdots$	$\vdots$	$\vdots$
η+n_s	ν+3n_s	$\frac{\Delta\tau_k}{8}$
η+ n_s	ν	$-\frac{\Delta\tau_k}{6}$
η+n_s+1	ν+1	$-\frac{\Delta\tau_k}{6}$
$\vdots$	$\vdots$	$\vdots$
η+2n_s	ν+n_s	$-\frac{\Delta\tau_k}{6}$
η+ n_s	ν+n_s	$-\frac{2\Delta\tau_k}{3}$
η+n_s+1	ν+n_s+1	$-\frac{2\Delta\tau_k}{3}$
$\vdots$	$\vdots$	$\vdots$
η+2n_s	ν+2n_s	$-\frac{2\Delta\tau_k}{3}$
η+ n_s	ν+2n_s	$-\frac{\Delta\tau_k}{6}$
η+n_s+1	ν+2n_s+1	$-\frac{\Delta\tau_k}{6}$
$\vdots$	$\vdots$	$\vdots$
η+2n_s	ν+3n_s	$-\frac{\Delta\tau_k}{6}$
$\vdots$	$\vdots$	$\vdots$
n_{colCns}	$(n_s)(n_{colPoints})$	1
$n_{colCns} + 1$	$(n_s)(n_{colPoints}) + 1$	1
$\vdots$	$\vdots$	$\vdots$
$n_{colCns} + p(n_{colPoints}) + n_e$	$(n_s)(n_{colPoints}) + n(n_{colPoints}) + n_e$	1

with $\eta = \nu = 2(k)(n_s)$ for $k = 0...N-1$

Algorithm 4 Propagation algorithm for B HSS matrix

```
for $k = 0, 1, \ldots, N-1$ do
    $i = 0$
    $\eta = (2k)(n_s)$
    $\nu = (2k)(n_s)$
    for $n_z = 0, 1, \ldots, n_s$ do
        $iRow = \eta + i$
        $iCol = \nu + i$
        Save the information in the triplet ($iRow$, $iCol$, $-\frac{\Delta\tau_k}{8}$)
        i++
    end for
    $i = 0$
    for $n_z = n_s, n_s + 1, \ldots, 2n_s$ do
        $iRow = \eta + i$
        $iCol = \nu + 2n_s + i$
        Save the information in the triplet ($iRow$, $iCol$, $\frac{\Delta\tau_k}{8}$)
        i++
    end for
    $i = 0$
    for $n_z = 2n_s, 2n_s + 1, \ldots, 3n_s$ do
        $iRow = \eta + n_s + i$
        $iCol = \nu + i$
        Save the information in the triplet ($iRow$, $iCol$, $-\frac{\Delta\tau_k}{6}$)
        i++
    end for
    $i = 0$
    for $n_z = 3n_s, 3n_s + 1, \ldots, 4n_s$ do
        $iRow = \eta + n_s + i$
        $iCol = \nu + n_s + i$
        Save the information in the triplet ($iRow$, $iCol$, $-\frac{2\Delta\tau_k}{3}$)
        i++
    end for
    $i = 0$
    for $n_z = 4n_s, 4n_s + 1, \ldots, b_{nz}$ do
        $iRow = \eta + n_s + i$
        $iCol = \nu + 2n_s + i$
        Save the information in the triplet ($iRow$, $iCol$, $-\frac{\Delta\tau_k}{6}$)
        i++
    end for
end for
$i = 0$
for $n_z = (5n_s)(N-1), \ldots, B_{nz}$ do
    $iRow = n_{colCns} + i$
    $iCol = (n_s)(n_{colPoints}) + i$
    Save the information in the triplet ($iRow$, $iCol$, 1)
    i++
end for
```

5 Computation of the Derivative Matrix using the propagation algorithm

Recall the structure of the constrains vector (Either trapezoidal or HSS)

$$\boldsymbol{c}(\boldsymbol{z}) = \boldsymbol{A}\boldsymbol{z} + \boldsymbol{B}\boldsymbol{y}(\boldsymbol{z}) \tag{55}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are constant matrices and $\boldsymbol{y}(\boldsymbol{z})$ is the non-linear vector. Suppose that the Jacobian of the constraints is computed given the following expression

$$\frac{\partial \boldsymbol{c}(\boldsymbol{z})}{\partial \boldsymbol{z}} = \boldsymbol{A} + \boldsymbol{B}\frac{\partial \boldsymbol{y}(\boldsymbol{z})}{\partial \boldsymbol{z}} \tag{56}$$

$$\frac{\partial \boldsymbol{c}(\boldsymbol{z})}{\partial \boldsymbol{z}} = \boldsymbol{A} + \boldsymbol{B}\boldsymbol{D} \tag{57}$$

where $\boldsymbol{D} \in \mathbb{R}^{(n_{colPoints}(n_s+p)+n_e+1)\times n_{decVar}}$ is known as **the derivative matrix**. This derivatives matrix exploits the sparsity of the local collocation methods allowing to propose a well-defined structure of the sparsity pattern

$$\boldsymbol{D} = \left[\begin{array}{ccccc}
\boldsymbol{1}_{n_s\times 2} & \boldsymbol{D}_{f_0} & & & \\
\boldsymbol{1}_{n_s\times 2} & \boldsymbol{0} & \boldsymbol{D}_{f_1} & & \\
\vdots & \vdots & \vdots & \ddots & \\
\boldsymbol{1}_{n_s\times 2} & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{D}_{f_N} \\
\boldsymbol{D}_{t_0}\ \boldsymbol{D}_{t_F} & \boldsymbol{D}_{e_0} & \boldsymbol{0} & \dots & \boldsymbol{D}_{e_N} \\
\boldsymbol{0} & \boldsymbol{D}_{p_0} & \boldsymbol{0} & & \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{D}_{p_1} & & \\
 & & & \ddots & \\
 & & & & \boldsymbol{D}_{p_N} \\
1\ \ 1 & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{0}
\end{array}\right] \tag{58}$$

where

$$\boldsymbol{D}_{f_k} = \Delta t \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{z}_k} = \Delta t \begin{bmatrix} \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{x}_k} & \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{u}_k} \end{bmatrix} \in \mathbb{R}^{n_s\times(n_s+n_u)} \tag{59}$$

$$\boldsymbol{D}_{p_k} = \frac{\partial \boldsymbol{\gamma}_k}{\partial \boldsymbol{z}_k} = \begin{bmatrix} \frac{\partial \boldsymbol{\gamma}_k}{\partial \boldsymbol{x}_k} & \frac{\partial \boldsymbol{\gamma}_k}{\partial \boldsymbol{u}_k} \end{bmatrix} \in \mathbb{R}^{p\times(n_s+n_u)} \tag{60}$$

$$\boldsymbol{D}_{t_0} = \frac{\partial \boldsymbol{e}}{\partial t_0} \in \mathbb{R}^{n_e} \tag{61}$$

$$\boldsymbol{D}_{t_F} = \frac{\partial \boldsymbol{e}}{\partial t_F} \in \mathbb{R}^{n_e} \tag{62}$$

$$\boldsymbol{D}_{e_0} = \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{z}_0} = \begin{bmatrix} \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}_0} & \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}_0} \end{bmatrix} \in \mathbb{R}^{n_e\times(n_s+n_u)} \tag{63}$$

$$\boldsymbol{D}_{e_N} = \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{z}_N} = \begin{bmatrix} \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}_N} & \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}_N} \end{bmatrix} \in \mathbb{R}^{n_e\times(n_s+n_u)} \tag{64}$$

5.1 Sparsity detection of the Derivative Matrix using the propagation algorithm

Unlike the constant matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ the sparsity pattern of the Derivative Matrix not only depends on the dimensions of the problem but also of the structure of the dynamics, path constraints and event constraints of the problem. To apply the propagation algorithm for the computation of this sparsity pattern the following methodology is propose:

1. Set the constant values of the Derivative matrix ($I_{n_s \times 2}$) into the triplet.
2. Compute the sparsity template of the D_{f_k} and D_{p_k} matrices.
3. Propagate the sparsity templates.
4. Compute the sparsity templates of the events constrains ($\boldsymbol{D}_{t_0}, \boldsymbol{D}_{t_F}, \boldsymbol{D}_{e_0}, \boldsymbol{D}_{e_N}$).
5. Insert the sparsity templates of the event constraints in to the triplet.

5.1.1 Constant values of the Derivative Matrix

Following the structure of (58) it is easy to notice that some elements remain constant independently of the structure of the system and its constraints (Varying only with the dimensions of the problem). In particular, the first two columns of the matrix follow this pattern along the propagation of the derivatives of the dynamics ($\boldsymbol{D}_{f_k}$) and at the end of the matrix for the time constraint (This constraint may be deprecated). The sparsity pattern can be described as:

Row	Column	Value
0	0	1
1	0	1
$\vdots$	$\vdots$	$\vdots$
$(n_s)(n_{colPoints}) - 1$	0	1
0	1	1
1	1	1
$\vdots$	$\vdots$	$\vdots$
$(n_s)(n_{colPoints}) - 1$	1	1
$(n_{colPoints})(n_s + p) + n_e$	0	1
$(n_{colPoints})(n_s + p) + n_e$	1	1

5.1.2 The derivative of the dynamics and its propagation

Let us propose a generalized sparsity template for D_{f_k}

Row	Column
α_1	Ω_1
α_2	Ω_2
$\vdots$	$\vdots$
$\alpha_{f_{nz}}$	$\Omega_{f_{nz}}$

where f_{nz} is the number of non-zero elements on D_{f_k}.

Using the structure of the derivative matrix, the following generalized template can be proposed for the propagation of D_{f_k}

Row	Column
$\alpha_1 + (k)(n_s)$	$\Omega_1 + (k)(n_s + n_u) + 2$
$\alpha_2 + (k)(n_s)$	$\Omega_2 + (k)(n_s + n_u) + 2$
$\vdots$	$\vdots$
$\alpha_{f_{nz}} + (k)(n_s)$	$\Omega_{f_{nz}} + (k)(n_s + n_u) + 2$

for $k = 0...(n_{colPoints} - 1)$

5.1.3 The derivative of the path constraints and its propagation

Let us propose a generalized sparsity template for D_{p_k}

Row	Column
μ_1	Φ_1
μ_2	Φ_2
$\vdots$	$\vdots$
$\mu_{p_{nz}}$	$\Phi_{p_{nz}}$

where p_{nz} is the number of non-zero elements on D_{p_k}.

Using the structure of the derivative matrix, the following generalized template can be proposed for the propagation of D_{p_k}

Row	Column
$\mu_1 + (k)(n_p) + p_{offset}$	$\Phi_1 + (k)(n_s + n_u) + 2$
$\mu_2 + (k)(n_p) + p_{offset}$	$\Phi_2 + (k)(n_s + n_u) + 2$
$\vdots$	$\vdots$
$\mu_{p_{nz}} + (k)(n_p) + p_{offset}$	$\Phi_{p_{nz}} + (k)(n_s + n_u) + 2$

where $p_{offset} = (n_{colPoints})(n_s) + n_e$ and for $k = 0...(n_{colPoints} - 1)$

5.1.4 The derivative of the event constraints

The event constraints not need any propagation through the derivative matrix, instead the insertion of the elements into the triplet is enough to define its sparsity.

Let us propose a generalized sparsity template for D_{t_0} and D_{t_F}

Row	Column
ε_1^0	0
ε_2^0	0
$\vdots$	$\vdots$
$\varepsilon_{t_{nz_0}}^0$	0

Row	Column
ε_1^F	1
ε_2^F	1
$\vdots$	$\vdots$
$\varepsilon_{t_{nz_F}}^F$	1

where t_{nz_0} is the number of non-zero elements on D_{t_0} and t_{nz_F} is the number of non-zero elements on D_{t_F}.

Using the structure of the derivative matrix, the following generalized template can be proposed for the insertion of D_{t_0} and D_{t_F}:

Row	Column
$\varepsilon_1^0 + e_{offset}$	0
$\varepsilon_2^0 + e_{offset}$	0
$\vdots$	$\vdots$
$\varepsilon_{t_{nz_0}}^0 + e_{offset}$	0
$\varepsilon_1^F + e_{offset}$	1
$\varepsilon_2^F + e_{offset}$	1
$\vdots$	$\vdots$
$\varepsilon_{t_{nz_F}}^F + e_{offset}$	1

where $e_{offset} = (n_{colPoints})(n_s)$.

Following the same procedure, the proposed generalized template of D_{e_0} :

Row	Column
i_1	j_1
i_2	j_2
$\vdots$	$\vdots$
$i_{e_{nz_0}}$	$j_{e_{nz_0}}$

and D_{e_N}:

Row	Column
a_1	b_1
a_2	b_2
$\vdots$	$\vdots$
$a_{e_{nz_N}}$	$b_{e_{nz_N}}$

where e_{nz_0} is the number of non-zero elements on D_{e_0} and e_{nz_N} is the number of non-zero elements on D_{e_N}.

The following generalized template can be proposed for the insertion of D_{e_0} and D_{e_N}:

Row	Column
$i_1 + e_{offset}$	$j_1 + 2$
$i_2 + e_{offset}$	$j_2 + 2$
$\vdots$	$\vdots$
$i_{e_{nz_0}} + e_{offset}$	$j_{e_{nz_0}} + 2$
$a_1 + e_{offset}$	$n_{decVar} - (n_s + n_u) + b_1$
$a_2 + e_{offset}$	$n_{decVar} - (n_s + n_u) + b_2$
$\vdots$	$\vdots$
$a_{e_{nz_N}} + e_{offset}$	$n_{decVar} - (n_s + n_u) + b_{e_{nz_N}}$

5.1.5 Computing the derivative matrix sparsity

The total number of non-zero on $\boldsymbol{D}$ can be computed as:

$$\boldsymbol{D}_{nz} = (f_{nz} + p_{nz} + 2n_s)(n_{colPoints}) + t_{nz_0} + t_{nz_F} + e_{nz_0} + e_{nz_N} + 2$$

Algorithm 5 Sparsity detection of D using propagation algorithm

```
 1: $p_{offset} = (n_{colPoints})(n_s) + n_e$
 2: $e_{offset} = (n_{colPoints})(n_s)$
 3: **for** $k = 0, 1, \ldots, (n_s)(n_{colPoints}) - 1$ **do**
 4:     $iRow = k$
 5:     $iCol = 0$
 6:     Save the information in the triplet ($iRow$, $iCol$, 1)
 7:     $iCol = 1$
 8:     Save the information in the triplet ($iRow$, $iCol$, 1)
 9: **end for**
10: **for** $k = 0, 1, \ldots, (n_{colPoints} - 1)$ **do**
11:     **for** $d = 0, 1, \ldots, f_{nz}$ **do**
12:         $iRow = \alpha_d + (k)(n_s)$
13:         $iCol = \Omega_d + (k)(n_s + n_u) + 2$
14:         Save the information in the triplet ($iRow$, $iCol$, 1)
15:     **end for**
16:     **for** $d = 0, 1, \ldots, p_{nz}$ **do**
17:         $iRow = \mu_d + (k)(p) + p_{offset}$
18:         $iCol = \Phi_d + (k)(n_s + n_u) + 2$
19:         Save the information in the triplet ($iRow$, $iCol$, 1)
20:     **end for**
21: **end for**
22: **for** $d = 0, 1, \ldots, t_{nz_0}$ **do**
23:     $iRow = \varepsilon_d^0 + e_{offset}$
24:     $iCol = 0$
25:     Save the information in the triplet ($iRow$, $iCol$, 1)
26: **end for**
27: **for** $d = 0, 1, \ldots, t_{nz_F}$ **do**
28:     $iRow = \varepsilon_d^F + e_{offset}$
29:     $iCol = 1$
30:     Save the information in the triplet ($iRow$, $iCol$, 1)
31: **end for**
32: **for** $d = 0, 1, \ldots, e_{nz_0}$ **do**
33:     $iRow = i_d + e_{offset}$
34:     $iCol = j_d + 2$
35:     Save the information in the triplet ($iRow$, $iCol$, 1)
36: **end for**
37: **for** $d = 0, 1, \ldots, e_{nz_N}$ **do**
38:     $iRow = a_d + e_{offset}$
39:     $iCol = n_{decVar} - (n_s + n_u) + b_d$
40:     Save the information in the triplet ($iRow$, $iCol$, 1)
41: **end for**
```

5.2 Computing the Derivative Matrix using the propagation algorithm

Let us propose the following Derivative Matrix D

$$\boldsymbol{D} = \begin{bmatrix} \frac{\partial \Delta t \boldsymbol{f}_0}{\partial t_0} & \frac{\partial \Delta t \boldsymbol{f}_0}{\partial t_F} & \boldsymbol{D}_{f_0} & & & \\ \frac{\partial \Delta t \boldsymbol{f}_1}{\partial t_0} & \frac{\partial \Delta t \boldsymbol{f}_1}{\partial t_F} & \boldsymbol{0} & \boldsymbol{D}_{f_1} & & \\ \vdots & \vdots & \vdots & & \ddots & \\ \frac{\partial \Delta t \boldsymbol{f}_N}{\partial \boldsymbol{t}_0} & \frac{\partial \Delta t \boldsymbol{f}_N}{\partial \boldsymbol{t}_F} & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{D}_{f_N} \\ \boldsymbol{D}_{t_0} & \boldsymbol{D}_{t_F} & \boldsymbol{D}_{e_0} & \boldsymbol{0} & \dots & \boldsymbol{D}_{e_N} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{D}_{p_0} & \boldsymbol{0} & & \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{D}_{p_1} & & \\ \vdots & \vdots & & & \ddots & \\ \boldsymbol{0} & \boldsymbol{0} & & & & \boldsymbol{D}_{p_N} \\ 1 & 1 & 0 & 0 & \dots & 0 \end{bmatrix} \tag{65}$$

where

$$\boldsymbol{D}_{f_k} = \Delta t \begin{bmatrix} \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{x}_k} & \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{u}_k} \end{bmatrix} \in \mathbb{R}^{n_s \times (n_s + n_u)} \tag{66}$$

$$\boldsymbol{D}_{p_k} = \begin{bmatrix} \frac{\partial \boldsymbol{\gamma}_k}{\partial \boldsymbol{x}_k} & \frac{\partial \boldsymbol{\gamma}_k}{\partial \boldsymbol{u}_k} \end{bmatrix} \in \mathbb{R}^{p \times (n_s + n_u)} \tag{67}$$

$$\boldsymbol{D}_{t_0} = \frac{\partial \boldsymbol{e}}{\partial t_0} \in \mathbb{R}^{n_e} \tag{68}$$

$$\boldsymbol{D}_{t_F} = \frac{\partial \boldsymbol{e}}{\partial t_F} \in \mathbb{R}^{n_e} \tag{69}$$

$$\boldsymbol{D}_{e_0} = \begin{bmatrix} \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}_0} & \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}_0} \end{bmatrix} \in \mathbb{R}^{n_e \times (n_s + n_u)} \tag{70}$$

$$\boldsymbol{D}_{e_N} = \begin{bmatrix} \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}_N} & \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}_N} \end{bmatrix} \in \mathbb{R}^{n_e \times (n_s + n_u)} \tag{71}$$

$$\frac{\partial \Delta t \boldsymbol{f}_k}{\partial t_0} = -\boldsymbol{f}_k \in \mathbb{R}^{n_s} \tag{72}$$

$$\frac{\partial \Delta t \boldsymbol{f}_k}{\partial t_F} = \boldsymbol{f}_k \in \mathbb{R}^{n_s} \tag{73}$$

Exploiting the propagation algorithm applied in the sparsity detection of the derivative matrix is possible to propose the following algorithm to compute the value of the derivative matrix

Algorithm 6 Computation of the Derivative Matrix

```
1: $p_{offset} = (n_{colPoints})(n_s) + n_e$
2: $e_{offset} = (n_{colPoints})(n_s)$
3: $h_k = t_F - t_0$
4: for $k = 0, 1, \dots, (n_{colPoints} - 1)$ do
5:     Compute the dynamics $\boldsymbol{f}_k$
6:     Compute $\boldsymbol{D}_{f_k}$ and $\boldsymbol{D}_{p_k}$
7:     for $d = kn_s, kn_s + 1, \dots, kn_s + n_s$ do
8:         $iRow = d$
9:         $iCol = 0$
10:        $f_d$ = The $d$ element of the $\boldsymbol{f}_k$ vector
11:        Save the information in the triplet ($iRow$, $iCol$, $-f_d$)
12:    end for
13:    for $d = 0, 1, \dots, f_{nz}$ do
14:        $iRow = \alpha_d + (k)(n_s)$
15:        $iCol = \Omega_d + (k)(n_s + n_u) + 2$
16:        $d_{f_k}$ =The element of $\boldsymbol{D}_{f_k}$ at $(\alpha_d, \Omega_d)$
17:        Save the information in the triplet ($iRow$, $iCol$, $h_k d_{f_k}$)
18:    end for
19:    for $d = 0, 1, \dots, p_{nz}$ do
20:        $iRow = \mu_d + (k)(p) + p_{offset}$
21:        $iCol = \Phi_d + (k)(n_s + n_u) + 2$
22:        $d_{p_k}$ =The element of $\boldsymbol{D}_{p_k}$ at $(\mu_d, \Phi_d)$
23:        Save the information in the triplet ($iRow$, $iCol$, $d_{p_k}$)
24:    end for
25: end for
26: Compute $\boldsymbol{D}_{t_0}$, $\boldsymbol{D}_{t_F}$, $\boldsymbol{D}_{e_0}$ , $\boldsymbol{D}_{e_N}$
27: for $d = 0, 1, \dots, t_{nz_0}$ do
28:     $iRow = \varepsilon_d^0 + e_{offset}$
29:     $d_{t_0}$ = The element of $\boldsymbol{D}_{t_0}$ at $\varepsilon_d^0$
30:     Save the information in the triplet ($iRow$, 0, $d_{t_0}$)
31: end for
32: for $d = 0, 1, \dots, t_{nz_F}$ do
33:     $iRow = \varepsilon_d^F + e_{offset}$
34:     $d_{t_F}$ = The element of $\boldsymbol{D}_{t_F}$ at $\varepsilon_d^F$
35:     Save the information in the triplet ($iRow$, 1, $d_{t_F}$)
36: end for
37: for $d = 0, 1, \dots, e_{nz_0}$ do
38:     $iRow = i_d + e_{offset}$
39:     $iCol = j_d + 2$
40:     $d_{e_0}$ = The element of $\boldsymbol{D}_{e_0}$ at $(i_d, j_d)$
41:     Save the information in the triplet ($iRow$, $iCol$, $d_{e_0}$)
42: end for
43: for $d = 0, 1, \dots, e_{nz_N}$ do
44:     $iRow = a_d + e_{offset}$
45:     $iCol = n_{decVar} - (n_s + n_u) + b_d$
46:     $d_{e_N}$ = The element of $\boldsymbol{D}_{e_N}$ at $(a_d, b_d)$
47:     Save the information in the triplet ($iRow$, $iCol$, $d_{e_N}$)
48: end for
```