Direct Normal Form Practical Implementation

January 30, 2021

Abstract

This document provides a method for finding nonlinear normal modes up to third order of a generic geometrically nonlinear structure. The use of normal form for the computation of nonlinear modes is limited by the fact that the method is based on a near identity transformation that requires the system to be expressed in modal basis. The method developed here discards this assumption and permits the nonlinear mapping from physical to normal coordinates without passing for the intermediate step of modal coordinates.

List of Symbols

Modal basis	**Physical basis**	**Description**
		Coordinates
$\mathbf{x}$	$\mathbf{X}$	Displacement
$\mathbf{y}$	$\mathbf{Y}$	Velocity
		Equation of Motion Tensors
$\mathbf{I}$	$\mathbf{M}$	Mass matrix
$\mathbf{\Omega}^2$	$\mathbf{K}$	Stiffness Matrix
$\mathbf{g}$	$\mathbf{G}$	Quadratic tensor
$\mathbf{h}$	$\mathbf{H}$	Cubic tensor
		Eigenvectors
$\mathbf{e}_i$	$\boldsymbol{\phi}_i$	i-th eigenvector
$\mathbf{I}$	$\mathbf{V}$	Full eigenvector matrix
		Quadratic Reconstruction vectors
$\mathbf{a}_{ij}$	$\boldsymbol{\alpha}_{ij}$	On displacement term (R_iR_j)
$\mathbf{b}_{ij}$	$\boldsymbol{\beta}_{ij}$	On displacement term (S_iS_j)
$\mathbf{c}_{ij}$	$\boldsymbol{\gamma}_{ij}$	On velocity term (R_iS_j)
		Cubic Reconstruction vectors
$\mathbf{r}_{ijk}$	$\boldsymbol{\rho}_{ijk}$	On displacement term $(R_iR_jR_k)$
$\mathbf{u}_{ijk}$	$\boldsymbol{\upsilon}_{ijk}$	On displacement term $(R_iS_jS_k)$
$\mathbf{m}_{ijk}$	$\boldsymbol{\mu}_{ijk}$	On velocity term $(S_iS_jS_k)$
$\mathbf{n}_{ijk}$	$\boldsymbol{\nu}_{ijk}$	On velocity term $(S_iR_jR_k)$
		Correction vectors
$\mathbf{A}_{ijk}$	$\mathcal{A}_{ijk}$	On displacement term $(R_iR_jR_k)$
$\mathbf{B}_{ijk}$	$\mathcal{B}_{ijk}$	On displacement term $(R_iS_jS_k)$

1 Framework

In this section the underlying theory behind the Direct Normal Form method is briefly summarised.

1.1 Problem

The Direct Normal Form method aims at building a reduced order model of a mechanical system with geometric nonlinearities. Equation of motion in physical coordinates:

$$\dot{\mathbf{X}} = \mathbf{Y} \tag{1a}$$

$$\mathbf{M}\dot{\mathbf{Y}} + \mathbf{K}\mathbf{X} + \mathbf{G}(\mathbf{X},\mathbf{X}) + \mathbf{H}(\mathbf{X},\mathbf{X},\mathbf{X}) = \mathbf{0} \tag{1b}$$

where the geometric nonlinearities are expressed here by means of a quadratic function $\mathbf{G}$ of the displacement vector $\mathbf{X}$, and a cubic one $\mathbf{H}$. The dimension of each system of equations in (1) is equivalent to the number of active dofs in the system N.

Let us suppose one wants to build a reduced system up to third order composed of $n \ll N$ master modes, then the first step of the analysis will be to calculate the linear eigenvalues and eigenvectors of such modes. In the following it is assumed that the eigenvectors are mass normalised. If the eigenvectors are collected in the $n \times N$ matrix $\mathbf{\Phi}$, and the corresponding eigenvalues are collected in the n vector $\boldsymbol{\omega}$, one will have:

$$\mathbf{\Phi}^{\mathrm{T}}\mathbf{M}\mathbf{\Phi} = \mathbf{I} \qquad \mathbf{\Phi}^{\mathrm{T}}\mathbf{K}\mathbf{\Phi} = \mathrm{diag}(\boldsymbol{\omega}) \tag{2}$$

1.2 Nonlinear Mapping

The reduced model is build assuming the following nonlinear mapping for the nodal displacement vector $\mathbf{X}$ and the nodal velocity vector $\mathbf{Y}$:

$$\mathbf{X} = \sum_{i}^{n} \boldsymbol{\phi}_i R_i \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{\alpha}_{ij} R_i R_j + \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{\beta}_{ij} S_i S_j + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} \boldsymbol{\rho}_{ijk} R_i R_j R_k + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} \boldsymbol{\upsilon}_{ijk} R_i S_j S_k \tag{3a}$$

$$\mathbf{Y} = \sum_{i}^{n} \boldsymbol{\phi}_i S_i \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{\gamma}_{ij} R_i S_j + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} \boldsymbol{\mu}_{ijk} S_i S_j S_k + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} \boldsymbol{\nu}_{ijk} S_i R_j R_k \tag{3b}$$

where the normal coordinates R_i and S_i have been introduced as the new variables one has to solve for. One must notice that all the summations span over the master modes from the first to the n-th but it is not necessary that the master modes coincides with the first n modes of the system.

In Eqs. (3), $\boldsymbol{\phi}_i$ is the eigenvector corresponding to the i-th master mode that is found by solving the linear eigenproblem denoted by the function `Eig`() later in the algorithm. The second order tensors $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ are the output of the function `SOT`() and the third order tensors $\boldsymbol{\rho}$, $\boldsymbol{\upsilon}$, $\boldsymbol{\mu}$, and $\boldsymbol{\nu}$ are the output of the function `TOT`().

1.3 Reduced Model

Upon the nonlinear mapping defined by Eq. (3), the method of normal form ensures that, in absence of second order internal resonances, the system of equation (1) reduces to the following ROM:

$$\dot{R}_p = S_p \tag{4a}$$

$$\dot{S}_p + \omega_p^2 R_p + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} Ah^p_{ijk} R_i R_j R_k + B^p_{ijk} R_i S_j S_k = 0 \tag{4b}$$

where the tensors $\mathbf{Ah}$ and $\mathbf{B}$ are another output of the method. These tensors coincides with classical normal form tensors $\mathbf{A}$ and $\mathbf{B}$ in modal basis and in particular the tensor $\mathbf{Ah}$ is defined here as:

$$\mathbf{Ah} = \mathbf{A} + \mathbf{h} \tag{5}$$

with $\mathbf{h}$ tensor of cubic modal forces in modal basis.

The presented algorithm allows the user to choose between stopping the method up to second order or performing it up to third order. In the former case, the third order tensors $\boldsymbol{\rho}$, $\boldsymbol{\upsilon}$, $\boldsymbol{\mu}$, and $\boldsymbol{\nu}$ are not computed and the tensors **Ah** and **B** are the output of the function `SOT()`. In the latter case, the third order tensors $\boldsymbol{\rho}$, $\boldsymbol{\upsilon}$, $\boldsymbol{\mu}$, and $\boldsymbol{\nu}$ are computed and the tensors **Ah** and **B** are the output of the function `TOT()`. The main difference in the results would be that in the former case, the tensors **Ah** and **B** will be full because the third order dynamics will not be reduced, whereas in the latter case, they will be nonzero only if a third order resonance is present.

2 Implementation of the Algorithm

The general structure of the algorithm is presented in Fig. 1. The inputs of the algorithm are the stiffness and mass matrix and the choice of the master modes the user wants in the reduced model and a vector of modal amplitudes defined by the user that will be introduced in the paragraph `StEP()`. The outputs are the tensors of the nonlinear mapping and the coefficients of the differential equations of the reduced model. The general structure has been divided for the sake of clarity into three subroutines, preliminary calculations, second order calculations, and third order calculations; each of them will be detailed in the next subsections.

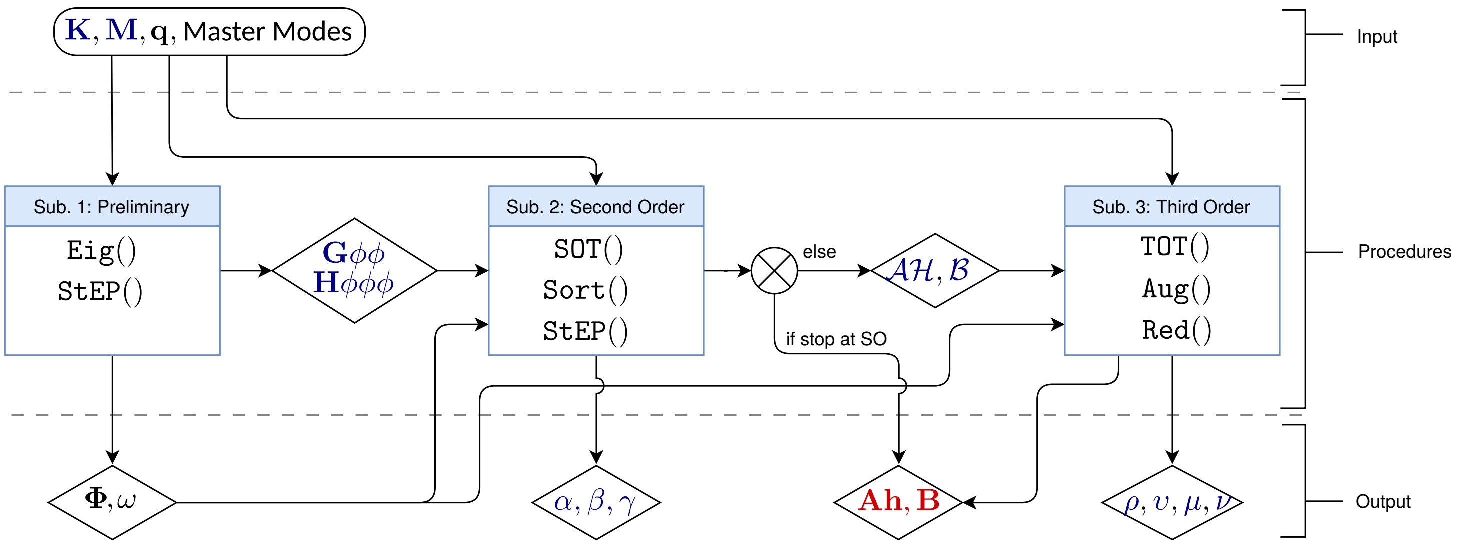

Figure 1 – Structure of the algorithm divided by its three main subroutines

2.1 Preliminary Calculations

The first step of the algorithm consists of the classical application of the StEP method (Stiffness Evaluation Procedure) to the chosen master modes of the system. The StEP method aims at extracting the quadratic and cubic nonlinear forces due to modal displacements by means of a sequence of application of ad hoc combinations of modes. The general description of this subroutine is provided in Alg. 1 and a detailed description of the inner functions `Eig()` and `StEP()` is provided in the next paragraphs. The subroutine detailed in Alg. 1 takes as inputs the stiffness and mass matrices, the selected master modes indexes in the form of an array and a vector of selected modal amplitudes. The latter input will be further explained in the paragraph `StEP()`. The outputs of this subroutine are a matrix containing the selected master modes eigenvectors, an array containing the selected master modes eigenvalues and the nonlinear quadratic and cubic tensors relative to the selected eigenvectors.

Algorithm 1: DNF Preliminary

```
Input     : $\mathbf{K}, \mathbf{M}, \boldsymbol{q}$, Master Modes
Functions: Eig, StEP
Output    : $\boldsymbol{\Phi}, \boldsymbol{\omega}, \mathbf{G}\boldsymbol{\phi\phi}, \mathbf{H}\boldsymbol{\phi\phi\phi}$
Initialise Outputs
$\boldsymbol{\Phi} \leftarrow \mathbf{0}_{N\times n}$
$\boldsymbol{\omega} \leftarrow \mathbf{0}_{n}$
$\mathbf{G}\boldsymbol{\phi\phi} \leftarrow \mathbf{0}_{N\times n\times n}$
$\mathbf{H}\boldsymbol{\phi\phi\phi} \leftarrow \mathbf{0}_{N\times n\times n\times n}$
Compute eigenproblem
$\boldsymbol{\Phi}, \boldsymbol{\omega} \leftarrow$ Eig ($\mathbf{K}, \mathbf{M}$, Master Modes)
Compute StEP method for single index
for $i \in (1, n)$ do
    $\mathbf{G}\boldsymbol{\phi\phi}[i,i], \mathbf{H}\boldsymbol{\phi\phi\phi}[i,i,i] \leftarrow$ StEPs ($\boldsymbol{\phi}_i, q_i$)
end
Compute StEP method for double indexes
for $i \in (1, n)$ do
    for $j \in (i+1, n)$ do
        $\mathbf{G}\boldsymbol{\phi\phi}[i,j], \mathbf{H}\boldsymbol{\phi\phi\phi}[i,i,j], \mathbf{H}\boldsymbol{\phi\phi\phi}[i,j,j] \leftarrow$ StEPd ($\boldsymbol{\phi}_i, q_i, \boldsymbol{\phi}_j, q_j$)
    end
end
Compute StEP method for triple indexes
for $i \in (1, n)$ do
    for $j \in (i+1, n)$ do
        for $k \in (j+1, n)$ do
            $\mathbf{H}\boldsymbol{\phi\phi\phi}[i,j,k] \leftarrow$ StEPt ($\boldsymbol{\phi}_i, q_i, \boldsymbol{\phi}_j, q_j, \boldsymbol{\phi}_k, q_k$)
        end
    end
end
```

2.1.1 Function `Eig()`

In this function the generalised eigenproblem relative to the matrices $\mathbf{K}$ and $\mathbf{M}$ is solved up to the highest master mode. Then, for each value in the vector of master modes defined by the user, the relative mode is saved in the $\boldsymbol{\Phi}$ matrix and the relative natural frequency (in rad/s) is saved in the vector $\boldsymbol{\omega}$.

2.1.2 Function `StEP()`

The StEP functions are classified by s, d, t, based on the number of indexes they solve the nonlinear forces for, respectively single, double, triple indexes.

In `StEPs()`, the input vector is only one mode, say $\boldsymbol{\phi}_i$, and the output are the values of the quadratic and cubic forces arising in the structure when a deformation along the mode is imposed: $\mathbf{G}\boldsymbol{\phi\phi}[i,i] = \mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i)$ and $\mathbf{H}\boldsymbol{\phi\phi\phi}[i,i,i] = \mathbf{H}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i, \boldsymbol{\phi}_i)$. In order to extrapolate these vectors one must have access to the vectors of nodal forces in the FE solver. Practically this can be done in a commercial software by imposing the modal displacement along $\boldsymbol{\phi}_i$ and extracting the forces from a static analysis. In order to isolate the purely nonlinear part of the nodal force, the static analysis has to be performed twice, the first time under the hypothesis of linear behaviour and then under the hypothesis of nonlinear behaviour. The linear nodal forces in the structure when it deforms along $\boldsymbol{\phi}_i$ are:

$$\boldsymbol{F}_L(\mathbf{X} = \boldsymbol{\phi}_i) = \mathbf{K}\boldsymbol{\phi}_i \tag{6}$$

and the nonlinear nodal forces are:

$$\boldsymbol{F}_T(\mathbf{X} = \boldsymbol{\phi}_i) = \mathbf{K}\boldsymbol{\phi}_i + \mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i) + \mathbf{H}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i, \boldsymbol{\phi}_i) \tag{7}$$

The difference between the forces evaluated under the nonlinear behaviour assumption and those evaluated under the linear one is equal to the purely nonlinear part of the force:

$$\boldsymbol{F}_{NL}(\boldsymbol{\phi}_i) = \boldsymbol{F}_T(\boldsymbol{\phi}_i) - \boldsymbol{F}_L(\boldsymbol{\phi}_i) = \mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i) + \mathbf{H}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i, \boldsymbol{\phi}_i) \tag{8}$$

In order to obtain the values of $\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i)$ and $\mathbf{H}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i, \boldsymbol{\phi}_i)$ one must first calculate $\boldsymbol{F}_{NL}(\boldsymbol{\phi}_i)$, then $\boldsymbol{F}_{NL}(-\boldsymbol{\phi}_i)$, and finally find:

$$\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i) = (\boldsymbol{F}_{NL}(\boldsymbol{\phi}_i) + \boldsymbol{F}_{NL}(-\boldsymbol{\phi}_i))/2 \tag{9a}$$

$$\mathbf{H}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i, \boldsymbol{\phi}_i) = (\boldsymbol{F}_{NL}(\boldsymbol{\phi}_i) - \boldsymbol{F}_{NL}(-\boldsymbol{\phi}_i))/2 \tag{9b}$$

For numerical accuracy, one might want to impose a modal amplitude different from 1 when computing these forces. In fact, if the modal amplitude is too low, one might not be able to appropriately activate the geometric nonlinear behaviour and incurs a numerical error. A good way of selecting the modal amplitude is to select the one able to produce a displacement of the order of magnitude of the thickness for clamped clamped structures and of the order of magnitude of the length for cantilever structures.

Upon a user definition of the modal amplitudes of each master modes inside the array $\boldsymbol{q}$, one might then extend Eqs. (9) to the general case of modal amplitudes q_i:

$$\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i) = (\boldsymbol{F}_{NL}(q_i\boldsymbol{\phi}_i) + \boldsymbol{F}_{NL}(-q_i\boldsymbol{\phi}_i))/(2q_i^2) \tag{10a}$$

$$\mathbf{H}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i, \boldsymbol{\phi}_i) = (\boldsymbol{F}_{NL}(q_i\boldsymbol{\phi}_i) - \boldsymbol{F}_{NL}(-q_i\boldsymbol{\phi}_i))/(2q_i^3) \tag{10b}$$

The algorithm of the function `StEPs()` is reported in Alg. 2. The functions `StEPd()` (see Alg. 3) and `StEPt()` (see Alg. 4) are instead used to compute double and triple indexes tensor entries. They follow a similar procedure of sequential application of the modal displacement with different signs in order to extract the desired values of the quadratic and cubic tensors. One must notice that the function `StEPs()` not only takes as input two vectors and two amplitudes for indexes i and j but also it receives the values previously calculated from `StEPs()` (e.g. $\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i), \mathbf{G}(\boldsymbol{\phi}_j, \boldsymbol{\phi}_j), \mathbf{H}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i, \boldsymbol{\phi}_i), \mathbf{H}(\boldsymbol{\phi}_j, \boldsymbol{\phi}_j, \boldsymbol{\phi}_j)$).

Algorithm 2: `StEPs()`

```
Input     : $\boldsymbol{\phi}_i, q_i$
Functions: $\boldsymbol{F}_{NL}()$
Output    : $\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i), \mathbf{H}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i, \boldsymbol{\phi}_i)$
1 Compute the purely nonlinear part of the nodal forces
2 $\boldsymbol{F}_{NL}^{+} \leftarrow \boldsymbol{F}_{NL}(+q_i\boldsymbol{\phi}_i)$
3 $\boldsymbol{F}_{NL}^{-} \leftarrow \boldsymbol{F}_{NL}(-q_i\boldsymbol{\phi}_i)$
4 Extract quadratic and cubic forces
5 $\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i) \leftarrow (\boldsymbol{F}_{NL}^{+} + \boldsymbol{F}_{NL}^{-})/(2q_i^2)$
6 $\mathbf{H}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i, \boldsymbol{\phi}_i) \leftarrow (\boldsymbol{F}_{NL}^{+} - \boldsymbol{F}_{NL}^{-})/(2q_i^3)$
```

Algorithm 3: `StEPd()`

```
Input     : $\boldsymbol{\phi}_i, q_i, \boldsymbol{\phi}_j, q_j, \mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i), \mathbf{H}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i, \boldsymbol{\phi}_i), \mathbf{G}(\boldsymbol{\phi}_j, \boldsymbol{\phi}_j), \mathbf{H}(\boldsymbol{\phi}_j, \boldsymbol{\phi}_j, \boldsymbol{\phi}_j)$
Functions: $\boldsymbol{F}_{NL}()$
Output    : $\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j), \mathbf{H}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i, \boldsymbol{\phi}_j), \mathbf{H}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j, \boldsymbol{\phi}_j)$
1 Compute the purely nonlinear part of the nodal forces
2 $\boldsymbol{F}_{NL}^{++} \leftarrow \boldsymbol{F}_{NL}(+q_i\boldsymbol{\phi}_i + q_j\boldsymbol{\phi}_j)$
3 $\boldsymbol{F}_{NL}^{+-} \leftarrow \boldsymbol{F}_{NL}(+q_i\boldsymbol{\phi}_i - q_j\boldsymbol{\phi}_j)$
4 $\boldsymbol{F}_{NL}^{--} \leftarrow \boldsymbol{F}_{NL}(-q_i\boldsymbol{\phi}_i - q_j\boldsymbol{\phi}_j)$
5 Extract quadratic and cubic forces
6 $\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j) \leftarrow (\boldsymbol{F}_{NL}^{++} + \boldsymbol{F}_{NL}^{--})/(4q_iq_j) - (q_i^2\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i) + q_j^2\mathbf{G}(\boldsymbol{\phi}_j, \boldsymbol{\phi}_j))/(2q_iq_j)$
7 $\mathbf{H}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i, \boldsymbol{\phi}_j) \leftarrow (-\boldsymbol{F}_{NL}^{+-} - \boldsymbol{F}_{NL}^{--})/(6q_i^2q_j) + (q_i^2\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i) + q_j^2\mathbf{G}(\boldsymbol{\phi}_j, \boldsymbol{\phi}_j) - q_j^3\mathbf{H}(\boldsymbol{\phi}_j, \boldsymbol{\phi}_j, \boldsymbol{\phi}_j))/(3q_i^2q_j)$
8 $\mathbf{H}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j, \boldsymbol{\phi}_j) \leftarrow (+\boldsymbol{F}_{NL}^{++} + \boldsymbol{F}_{NL}^{+-})/(6q_iq_j^2) - (q_i^2\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i) + q_j^2\mathbf{G}(\boldsymbol{\phi}_j, \boldsymbol{\phi}_j) + q_i^3\mathbf{H}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i, \boldsymbol{\phi}_i))/(3q_iq_j^2)$
```

```
Algorithm 4: StEPt()
    Input     : $\boldsymbol{\phi}_i, q_i, \boldsymbol{\phi}_j, q_j, \boldsymbol{\phi}_k, q_k, \mathbf{G}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_i), \mathbf{H}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_i,\boldsymbol{\phi}_i), \mathbf{G}(\boldsymbol{\phi}_j,\boldsymbol{\phi}_j), \mathbf{H}(\boldsymbol{\phi}_j,\boldsymbol{\phi}_j,\boldsymbol{\phi}_j), \mathbf{G}(\boldsymbol{\phi}_k,\boldsymbol{\phi}_k),$
                $\mathbf{H}(\boldsymbol{\phi}_k,\boldsymbol{\phi}_k,\boldsymbol{\phi}_k), \mathbf{G}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_j), \mathbf{H}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_i,\boldsymbol{\phi}_j), \mathbf{H}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_j,\boldsymbol{\phi}_j), \mathbf{G}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_k),$
                $\mathbf{H}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_i,\boldsymbol{\phi}_k), \mathbf{H}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_k,\boldsymbol{\phi}_k), \mathbf{G}(\boldsymbol{\phi}_j,\boldsymbol{\phi}_k), \mathbf{H}(\boldsymbol{\phi}_j,\boldsymbol{\phi}_j,\boldsymbol{\phi}_k), \mathbf{H}(\boldsymbol{\phi}_j,\boldsymbol{\phi}_k,\boldsymbol{\phi}_k)$
    Functions: $\boldsymbol{F}_{NL}()$
    Output    : $\mathbf{H}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_j,\boldsymbol{\phi}_k)$
1   Compute the purely nonlinear part of the nodal forces
2   $\boldsymbol{F}_{NL} \leftarrow \boldsymbol{F}_{NL}(+q_i\boldsymbol{\phi}_i + q_j\boldsymbol{\phi}_j + q_k\boldsymbol{\phi}_k)$
3   Extract cubic forces
4   $\mathbf{H}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_j,\boldsymbol{\phi}_k) \leftarrow \boldsymbol{F}_{NL}/(4q_iq_j) - (q_i^2\mathbf{G}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_i) + q_j^2\mathbf{G}(\boldsymbol{\phi}_j,\boldsymbol{\phi}_j) + q_k^2\mathbf{G}(\boldsymbol{\phi}_k,\boldsymbol{\phi}_k))/(6q_iq_jq_k) +$
      $-(q_iq_j\mathbf{G}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_j) + q_iq_k\mathbf{G}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_k) + q_jq_k\mathbf{G}(\boldsymbol{\phi}_j,\boldsymbol{\phi}_k))/(3q_iq_jq_k) +$
      $-(q_i^3\mathbf{H}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_i,\boldsymbol{\phi}_i) + q_j^3\mathbf{H}(\boldsymbol{\phi}_j,\boldsymbol{\phi}_j,\boldsymbol{\phi}_j) + q_k^3\mathbf{H}(\boldsymbol{\phi}_k,\boldsymbol{\phi}_k,\boldsymbol{\phi}_k))/(6q_iq_jq_k) +$
      $-(q_i^2q_j\mathbf{H}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_i,\boldsymbol{\phi}_j) + q_iq_j^2\mathbf{H}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_j,\boldsymbol{\phi}_j) + q_i^2q_k\mathbf{H}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_i,\boldsymbol{\phi}_k) +$
      $q_iq_k^2\mathbf{H}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_k,\boldsymbol{\phi}_k) + q_j^2q_k\mathbf{H}(\boldsymbol{\phi}_j,\boldsymbol{\phi}_j,\boldsymbol{\phi}_k) + q_jq_k^2\mathbf{H}(\boldsymbol{\phi}_j,\boldsymbol{\phi}_k,\boldsymbol{\phi}_k))/(6q_iq_jq_k)$
```

2.2 Second Order Calculations

This subroutine provides the second order tensors of the nonlinear mapping $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ and, if the calculations are stopped at second order, it provides the reduced dynamics coefficients tensors $\mathbf{Ah}, \mathbf{B}$; else if the calculations are up to third order, it produces the tensors $\mathcal{AH}, \mathcal{B}$ to be used as inputs for the subroutine Third Order Calculations. One must notice that the reduced dynamics tensors $\mathbf{Ah}, \mathbf{B}$ are equivalent to those in classical normal form. In fact the tensors $\boldsymbol{\alpha}, \boldsymbol{\beta}$ coincide with $\mathbf{Va}, \mathbf{Vb}$ where $\mathbf{V}$ is the full eigenvector matrix (not needed in this method) and $\mathbf{a}, \mathbf{b}$ are the classical normal form quadratic tensors in modal basis. The operations of lines 28,29 are then a sort of equivalent of the summation of classical normal form:

$$\mathbf{A}_{ijk} = 2\sum_s^N \mathbf{g}_{is} a^s_{jk} \tag{11a}$$

$$\mathbf{B}_{ijk} = 2\sum_s^N \mathbf{g}_{is} b^s_{jk} \tag{11b}$$

In fact, in lines 28,29 the calculation of $\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\alpha}_{jk})$ and $\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\beta}_{jk})$ are equivalent to:

$$\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\alpha}_{jk}) = \mathbf{G}(\boldsymbol{\phi}_i, \mathbf{Va}_{jk}) = \sum_s^N \mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_s) a^s_{jk} \tag{12a}$$

$$\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\beta}_{jk}) = \mathbf{G}(\boldsymbol{\phi}_i, \mathbf{Vb}_{jk}) = \sum_s^N \mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_s) b^s_{jk} \tag{12b}$$

and the operations of projection on the master modes done at lines 41,42 transform $\mathbf{G}$ into $\mathbf{g}$:

$$\mathbf{g}_{is} = \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_s) \tag{13}$$

In this method, the reduced dynamics are then calculated simply as:

$$\mathbf{A}_{ijk} = 2\boldsymbol{\Phi}^{\mathrm{T}} \mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\alpha}_{jk}) \tag{14}$$

$$\mathbf{B}_{ijk} = 2\boldsymbol{\Phi}^{\mathrm{T}} \mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\beta}_{jk}) \tag{15}$$

The inputs of this subroutines are $\mathbf{K}, \mathbf{M}, \boldsymbol{q}, \boldsymbol{\omega}, \boldsymbol{\Phi}, \mathbf{G}\boldsymbol{\phi\phi}, \mathbf{H}\boldsymbol{\phi\phi\phi}$ produced in the preliminary calculations. The functions `SOT()`, `Sorted()` used by this subroutine will be detailed in the next paragraphs. The function `StEP()` is used in this subroutine in the same way as in the preliminary results with the only difference being that here the tensors $\boldsymbol{\alpha}, \boldsymbol{\beta}$ are given as input and their amplitudes are the product of two modal amplitudes. The structure of the `StEP()` functions should not be changed, only their inputs are.

2.2.1 Function `Sort`()

The entries $\mathbf{H}\boldsymbol{\phi\phi\phi}[i,j,k]$ are nonzero only for $i \leq j \leq k$ because the loops of the StEP method are not full loops on j and k. However, due to the symmetry of the cubic tensor stemming from the conservativeness of the elastic energy, the entries for $i < j < k$ are equal to those for $i \leq j \leq k$ (e.g. $\mathbf{H}\boldsymbol{\phi\phi\phi}[i,j,k] = \mathbf{H}\boldsymbol{\phi\phi\phi}[j,i,k] = \mathbf{H}\boldsymbol{\phi\phi\phi}[k,j,i]$, etc.). Since the loops in this subroutine are not for $i \leq j$ but j goes from 1 to n, for each combination of i,j,k one must find the nonzero entry of $\mathbf{H}\boldsymbol{\phi\phi\phi}$ and use it assignment of line 31 of Alg. 5. To do so, one must simply sort the indexes i,j,k before picking the corresponding entry on $\mathbf{H}\boldsymbol{\phi\phi\phi}$.

2.2.2 Function `SOT`()

The second order tensors $\boldsymbol{\alpha}_{ij}$, $\boldsymbol{\beta}_{ij}$, $\boldsymbol{\gamma}_{ij}$, $\boldsymbol{\gamma}_{ji}$ of Eq. (3) are obtained through the following procedure.

Define the matrices:

$$\mathbf{Ds} = ((+\omega_i + \omega_j)^2\mathbf{M} - \mathbf{K}) \tag{16}$$
$$\mathbf{Dd} = ((-\omega_i + \omega_j)^2\mathbf{M} - \mathbf{K}) \tag{17}$$

Define the vectors:

$$\mathbf{Zs} = \mathbf{Ds}^{-1}\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j) \tag{18}$$
$$\mathbf{Zd} = \mathbf{Dd}^{-1}\mathbf{G}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j) \tag{19}$$

that can be computationally obtained easily by solving the linear system. This operation is a nonstandard operation in a finite element code (unlike $\mathbf{K}^{-1}$ that would correspond to a static linear analysis) but takes a little computational time even for large systems, same time than a static linear analysis. One must notice that in case any of the eigenvalues of the system, say the s-th, has a resonance relation such as $\omega_i + \omega_j = \omega_s$ or $-\omega_i + \omega_j = \omega_s$, a second order internal resonance occurs and either the $\mathbf{Ds}$ or $\mathbf{Dd}$ matrix becomes singular. The DNF method cannot at the moment deal with second order resonances but this feature will be soon implemented.

Obtain the second order tensors:

$$\boldsymbol{\alpha}_{ij} = \frac{1}{2}(\mathbf{Zd} + \mathbf{Zs}) \tag{20}$$
$$\boldsymbol{\beta}_{ij} = \frac{1}{2\omega_i\omega_j}(\mathbf{Zd} - \mathbf{Zs}) \tag{21}$$
$$\boldsymbol{\gamma}_{ij} = \frac{\omega_j - \omega_i}{\omega_j}\mathbf{Zd} + \frac{\omega_j + \omega_i}{\omega_j}\mathbf{Zs} \tag{22}$$
$$\boldsymbol{\gamma}_{ji} = \frac{\omega_i - \omega_j}{\omega_i}\mathbf{Zd} + \frac{\omega_i + \omega_j}{\omega_i}\mathbf{Zs} \tag{23}$$

Algorithm 5: DNF Second Order

```
Input     : $\mathbf{K}, \mathbf{M}, \boldsymbol{q}, \boldsymbol{\omega}, \boldsymbol{\Phi}, \mathbf{G}\boldsymbol{\phi\phi}, \mathbf{H}\boldsymbol{\phi\phi\phi}$
Functions: SOT, StEP, Sort
Output    : $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$, (either $\mathcal{AH}, \mathcal{B}$; or $\mathbf{Ah}, \mathbf{B}$)

*Initialise Outputs*
$\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma} \leftarrow \mathbf{0}_{N\times n\times n}, \mathbf{0}_{N\times n\times n}, \mathbf{0}_{N\times n\times n}$
$\mathcal{AH}, \mathcal{B} \leftarrow \mathbf{0}_{N\times n\times n\times n}, \mathbf{0}_{N\times n\times n\times n}$
$\mathbf{Ah}, \mathbf{B} \leftarrow \mathbf{0}_{n\times n\times n\times n}, \mathbf{0}_{n\times n\times n\times n}$

*Initialise temporary tensors*
$\mathbf{G}\boldsymbol{\phi\alpha}, \mathbf{G}\boldsymbol{\phi\beta} \leftarrow \mathbf{0}_{N\times n\times n\times n}, \mathbf{0}_{N\times n\times n\times n}$
$\mathbf{G}\boldsymbol{\alpha\alpha}, \mathbf{G}\boldsymbol{\beta\beta} \leftarrow \mathbf{0}_{N\times n\times n}, \mathbf{0}_{N\times n\times n}$

*Compute second order tensors*
for $i \in (1, n)$ do
    for $j \in (i, n)$ do
        $\boldsymbol{\alpha}_{ij}, \boldsymbol{\beta}_{ij}, \boldsymbol{\gamma}_{ij}, \boldsymbol{\gamma}_{ji} \leftarrow$ SOT $(\mathbf{K}, \mathbf{M}, \omega_i, \omega_j, \mathbf{G}\boldsymbol{\phi\phi}[i, j])$
        $\boldsymbol{\alpha}_{ji} \leftarrow \boldsymbol{\alpha}_{ij}$
        $\boldsymbol{\beta}_{ji} \leftarrow \boldsymbol{\beta}_{ij}$
    end
end

*Compute StEP method for single index on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$*
for $j \in (1, n)$ do
    for $k \in (j, n)$ do
        $\mathbf{G}\boldsymbol{\alpha\alpha}[j, k] \leftarrow$ StEPs $(\boldsymbol{\alpha}_{jk}, q_j q_k)$
        $\mathbf{G}\boldsymbol{\beta\beta}[j, k] \leftarrow$ StEPs $(\boldsymbol{\beta}_{jk}, q_j q_k)$
    end
end

*Compute StEP method for double indexes on $\boldsymbol{\alpha}, \boldsymbol{\phi}$ and $\boldsymbol{\beta}, \boldsymbol{\phi}$*
*and obtain dynamic correction tensors $\mathcal{AH}$ and $\mathcal{B}$*
for $i \in (1, n)$ do
    for $j \in (1, n)$ do
        for $k \in (j, n)$ do
            $\mathbf{G}\boldsymbol{\phi\alpha}[i, j, k] \leftarrow$ StEPd $(\boldsymbol{\phi}_i, q_i, \boldsymbol{\alpha}_{jk}, q_j q_k)$
            $\mathbf{G}\boldsymbol{\phi\beta}[i, j, k] \leftarrow$ StEPd $(\boldsymbol{\phi}_i, q_i, \boldsymbol{\beta}_{jk}, q_j q_k)$
            $l, m, n \leftarrow$ Sort $(i, j, k)$
            $\mathcal{AH}_{ijk} \leftarrow 2\,\mathbf{G}\boldsymbol{\phi\alpha}[i, j, k] + \mathbf{H}\boldsymbol{\phi\phi\phi}[l, m, n])$
            $\mathcal{B}_{ijk} \leftarrow 2\,\mathbf{G}\boldsymbol{\phi\beta}[i, j, k]$
        end
    end
end

*Compute reduced dynamics tensors for a second order normal form rom*
if *DNF up to second order* then
    for $i \in (1, n)$ do
        for $j \in (1, n)$ do
            for $k \in (j, n)$ do
                $\mathbf{Ah}_{ijk} \leftarrow \boldsymbol{\Phi}^{\mathrm{T}} \mathcal{AH}_{ijk}$
                $\mathbf{B}_{ijk} \leftarrow \boldsymbol{\Phi}^{\mathrm{T}} \mathcal{B}_{ijk}$
            end
        end
    end
else
    pass
end
```

2.3 Third Order Calculations

This subroutine is only called if the user chooses to compute the normal form up to third order. The main differences with the results obtained with second order normal form are: (i) third order tensors will be present in the nonlinear mapping, and (ii) the reduced dynamics coefficients tensors will have a nonzero entry Ath^r_{ijk} only when a resonance condition between the eigenvalues relative to their indexes (r, i, j, k) is present, and will be zero elsewhere. The reduced model will then be much simpler and more "reduced" than that obtained with second order DNF. The zero entries in the third order reduced dynamics are in fact terms cancelled by the presence of the third order tensors in the nonlinear mapping. In this subroutine, the trivial resonances are made explicit and saved into a list. Every time the loops for i, j, k are in a resonance condition with another master mode r, the condition of mass-orthogonality between each of the tensors $\boldsymbol{\rho}_{ijk}, \boldsymbol{\upsilon}_{ijk}, \boldsymbol{\mu}_{ijk}, \boldsymbol{\nu}_{ijk}$ and $\boldsymbol{\phi}_r$ will be ensured by the augmentation of the stiffness matrix (see `Aug()`).

Algorithm 6: DNF Third Order

```
Input      : $\mathbf{K}, \mathbf{M}, \boldsymbol{\omega}, \boldsymbol{\Phi}, \mathcal{AH}, \mathcal{B}$
Functions: TOT, Aug, Red, Sort
Output     : $\boldsymbol{\rho}, \boldsymbol{\upsilon}, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{Ah}, \boldsymbol{B}$
1  Initialise Outputs
2  $\boldsymbol{\rho}, \boldsymbol{\upsilon}, \boldsymbol{\mu}, \boldsymbol{\nu} \leftarrow \mathbf{0}_{N\times n\times n,\times n}, \mathbf{0}_{N\times n\times n,\times n}, \mathbf{0}_{N\times n\times n,\times n}, \mathbf{0}_{N\times n\times n,\times n}$
3  $\mathbf{Ah}, \mathbf{B} \leftarrow \mathbf{0}_{n\times n\times n\times n}, \mathbf{0}_{n\times n\times n\times n}$
4  Build a list of all possible third order trivial resonances
5  for $i \in (1, n)$ do
6      for $j \in (i, n)$ do
7          Int Res $\leftarrow$ Append $([i, i, j, j])$
8      end
9  end
10 Compute third order tensors
11 for $i \in (1, n)$ do
12     for $j \in (1, n)$ do
13         for $k \in (j, n)$ do
14             for $r \in (1, n)$ do
15                 if Sort $(i,j,k,r) \in$ Int Res then
16                     $\tilde{\mathbf{K}}, \tilde{\mathbf{M}}, \tilde{\mathcal{AH}}_{ijk}, \tilde{\mathcal{B}}_{ijk} \leftarrow$ Aug $(\mathbf{K}, \mathbf{M}, \mathcal{AH}_{ijk}, \mathcal{B}_{ijk}, \boldsymbol{\phi}_r)$
17                     $\tilde{\boldsymbol{\rho}}_{ijk}, \tilde{\boldsymbol{\upsilon}}_{ijk}, \tilde{\boldsymbol{\mu}}_{ijk}, \tilde{\boldsymbol{\nu}}_{ijk} \leftarrow$ TOT $(\tilde{\mathbf{K}}, \tilde{\mathbf{M}}, \omega_i, \omega_j, \omega_k, \tilde{\mathcal{AH}}_{ijk}, \tilde{\mathcal{B}}_{ijk})$
18                     $\boldsymbol{\rho}_{ijk}, \boldsymbol{\upsilon}_{ijk}, \boldsymbol{\mu}_{ijk}, \boldsymbol{\nu}_{ijk} \leftarrow$ Red $(\tilde{\boldsymbol{\rho}}_{ijk}, \tilde{\boldsymbol{\upsilon}}_{ijk}, \tilde{\boldsymbol{\mu}}_{ijk}, \tilde{\boldsymbol{\nu}}_{ijk})$
19                     $Ah^r_{ijk} \leftarrow \boldsymbol{\phi}_r^{\mathrm{T}} \mathcal{AH}_{ijk}$
20                     $B^r_{ijk} \leftarrow \boldsymbol{\phi}_r^{\mathrm{T}} \mathcal{B}_{ijk}$
21                 else
22                     $\boldsymbol{\rho}_{ijk}, \boldsymbol{\upsilon}_{ijk}, \boldsymbol{\mu}_{ijk}, \boldsymbol{\nu}_{ijk} \leftarrow$ TOT $(\mathbf{K}, \mathbf{M}, \omega_i, \omega_j, \omega_k, \mathcal{AH}_{ijk}, \mathcal{B}_{ijk})$
23                 end
24             end
25         end
26     end
27 end
```

2.3.1 Function `TOT()`

The third order tensors are build through the following procedure. Define the matrices:

$$\mathbf{Da} = ((+\omega_i + \omega_j + \omega_k)^2 \mathbf{M} - \mathbf{K}) \tag{24a}$$

$$\mathbf{Di} = ((-\omega_i + \omega_j + \omega_k)^2 \mathbf{M} - \mathbf{K}) \tag{24b}$$

$$\mathbf{Dj} = ((+\omega_i - \omega_j + \omega_k)^2 \mathbf{M} - \mathbf{K}) \tag{24c}$$

$$\mathbf{Dk} = ((+\omega_i + \omega_j - \omega_k)^2 \mathbf{M} - \mathbf{K}) \tag{24d}$$

Define the vectors: Combination of third order forces:

$$\mathbf{Pa} = \mathcal{AH}_{ijk} + \mathcal{AH}_{jki} + \mathcal{AH}_{kij} - \omega_j\omega_k\mathcal{B}_{ijk} - \omega_k\omega_i\mathcal{B}_{jki} - \omega_i\omega_j\mathcal{B}_{kij} \tag{25a}$$
$$\mathbf{Pi} = \mathcal{AH}_{ijk} + \mathcal{AH}_{jki} + \mathcal{AH}_{kij} - \omega_j\omega_k\mathcal{B}_{ijk} + \omega_k\omega_i\mathcal{B}_{jki} + \omega_i\omega_j\mathcal{B}_{kij} \tag{25b}$$
$$\mathbf{Pj} = \mathcal{AH}_{ijk} + \mathcal{AH}_{jki} + \mathcal{AH}_{kij} + \omega_j\omega_k\mathcal{B}_{ijk} - \omega_k\omega_i\mathcal{B}_{jki} + \omega_i\omega_j\mathcal{B}_{kij} \tag{25c}$$
$$\mathbf{Pk} = \mathcal{AH}_{ijk} + \mathcal{AH}_{jki} + \mathcal{AH}_{kij} + \omega_j\omega_k\mathcal{B}_{ijk} + \omega_k\omega_i\mathcal{B}_{jki} - \omega_i\omega_j\mathcal{B}_{kij} \tag{25d}$$

Solve for the vectors (again a non-standard operation but not a computationally expensive one):

$$\mathbf{Za} = \mathbf{Da}^{-1}\mathbf{Pa} \tag{26a}$$
$$\mathbf{Zi} = \mathbf{Di}^{-1}\mathbf{Pi} \tag{26b}$$
$$\mathbf{Zj} = \mathbf{Dj}^{-1}\mathbf{Pj} \tag{26c}$$
$$\mathbf{Zk} = \mathbf{Dk}^{-1}\mathbf{Pk} \tag{26d}$$

Construct the third order tensors:

$$\boldsymbol{\rho}_{ijk} = \frac{1}{12}\left(\mathbf{Za} + \mathbf{Zi} + \mathbf{Zj} + \mathbf{Zk}\right) \tag{27a}$$
$$\boldsymbol{\mu}_{ijk} = \frac{1}{4\omega_j\omega_k}\left(-\mathbf{Za} - \mathbf{Zi} + \mathbf{Zj} + \mathbf{Zk}\right) \tag{27b}$$
$$\boldsymbol{\upsilon}_{ijk} = \boldsymbol{\mu}_{ijk} \tag{27c}$$
$$\boldsymbol{\nu}_{ijk} = \frac{1}{4\omega_i}\left((+\omega_i + \omega_j + \omega_k)\mathbf{Za} - (-\omega_i + \omega_j + \omega_k)\mathbf{Zi} + (+\omega_i - \omega_j + \omega_k)\mathbf{Zj} + (+\omega_i + \omega_j - \omega_k)\mathbf{Zk}\right) \tag{27d}$$

Depending on the inputs this function takes in, either augmented matrices and vectors or not augmented ones, the size of the temporary variables in this function will change but the operations will not.

2.3.2 Function `Aug()`

If (i,j,k,r) satisfy the resonance condition, one of the matrices $\mathbf{D}$ will be singular and the system in the function `TOT()` cannot be solved. The kernel corresponding to the zero eigenvalue (the r-th one resonating with i, j, k) is the r-th eigenvector $\boldsymbol{\phi}_r$. If one impose a constraint on the solution of the system to be mass-orthogonal to the r-th eigenvector by augmenting the stiffness matrix as:

$$\tilde{\mathbf{K}} = \begin{bmatrix} \mathbf{K} & \mathbf{M}\boldsymbol{\phi}_s \\ (\mathbf{M}\boldsymbol{\phi}_s)^{\mathrm{T}} & 0 \end{bmatrix} \tag{28}$$

and append a row and a columns of zeros to the mass matrix:

$$\tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 0 \end{bmatrix} \tag{29}$$

the system to be solved in `TOT()` becomes nonsingular.

The nonlinear force vectors must also be augmented as:

$$\tilde{\mathcal{AH}}_{ijk} = \begin{bmatrix} \mathcal{AH}_{ijk} \\ 0 \end{bmatrix}, \qquad \tilde{\mathcal{B}}_{ijk} = \begin{bmatrix} \mathcal{B}_{ijk} \\ 0 \end{bmatrix} \tag{30}$$

2.3.3 Function `Red()`

Extract the meaningful part ($\boldsymbol{\rho}_{ijk}$) from the solution vector ($\tilde{\boldsymbol{\rho}}_{ijk}$) by discarding the last entry:

$$\tilde{\boldsymbol{\rho}}_{ijk} = \begin{bmatrix} \boldsymbol{\rho}_{ijk} \\ \sim \end{bmatrix}, \qquad \tilde{\boldsymbol{\upsilon}}_{ijk} = \begin{bmatrix} \boldsymbol{\upsilon}_{ijk} \\ \sim \end{bmatrix}, \qquad \tilde{\boldsymbol{\mu}}_{ijk} = \begin{bmatrix} \boldsymbol{\mu}_{ijk} \\ \sim \end{bmatrix}, \qquad \tilde{\boldsymbol{\nu}}_{ijk} = \begin{bmatrix} \boldsymbol{\nu}_{ijk} \\ \sim \end{bmatrix} \tag{31}$$