Theory Guide

Fedem 8.0

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Fedem, an abbreviation for Finite Element Dynamics in Elastic Mechanisms, is a code for effective modeling, simulation and visualization of finite element assemblies and control systems. The code is based on a non-linear finite element formulation, which predicts the dynamic response of elastic mechanisms experiencing non-linear effects such as large rigid-body rotations. The elastic and rigid-body motions of a mechanism are solved together with control systems. Fedem represents in this respect *Multidisciplinary Mechanical Analysis*.

This theory guide is intended to provide users and others with insight into the mathematical and physical basis of the numerical simulation code.

1.1 History

The theory behind Fedem was originally developed by professor Ole Ivar Sivertsen in the late 1970s and through the 1920s. His work initiated new Ph.D. studies and international R&D projects that contributed to the development of the first FEDEM software product.

On the basis of Professor Sivertsen's work, a company was established in 1992 by Sintef, Northern Europe's largest R&D institute based in Trondheim. Computer speed reached levels that would allow the theories to produce results, and visualization technology made it possible to create a user interface. During this period the Fedem software was strictly an inhouse code at Sintef.

In 1995 the company Fedem, which later became Fedem Technology, continued the development of the user interface and made it possible to offer Fedem as a commercial product in 1998.

The Fedem software was continously developed as a product by Fedem Technology during the 2000s and 2010s while it was also used as an internal tool in various consultancy projects, until the company was aquired by SAP SE in 2016. Since then, the Fedem solvers have been provided as components in the EPD Connected Products by SAP, until the sunsetting of the Connected Products in 2023. It was then decided to release Fedem under a open source license on github, as a service to the existing user community.

1.2 System Simulation Methods

One method of System Simulation is the so-called Multi Body System Simulation, with which the real physical behavior of the product under investigation can be reduced to a few - for instance overall behavior relevant - characteristics and then can be simulated as a numerical model on the computer. Each body is usually treated as completely rigid.

The Finite Element Method is used for instance for a vehicle to calculate fatigue, stiffness, dynamical behavior of the car body, of chassis components, of engines. The size and the shape of the Finite Elements are chosen according to the required accuracy of the results. For many years the Finite Element Method has been a successful tool for product analysis with respect to functionality and safety.

New simulation tools are now commercially available, the so-called Multi Discipline Simulation tools, which combine Multi Body System Simulation, the Finite Element Method and Control Engineering. These computer programs make a much more detailed modeling possible and correspondingly yield much more accurate Multi Body System Simulation results.

1.3 Terminology and Definitions

Most of the terms and names used within Fedem is quite standard within Finite Element technology and dynamic simulation. However, over the years certain terms have evolved among users and developers of the software, to a point where the terminology has become an integral part of the software product.

DOF Degree of Freedom. For the mechanism models of Fedem, a DOF is usually a translational or rotational degree of freedom. Regarding a control system a DOF is defined more broadly as simply an unknown of the equation system to be solved.

Triad Numerically a triad is simply a node with displacement DOFs. The term has been coined within the modeling frame to better to describe the modeling entity that eventually results in a node in the computational model. Because of its frequent use within the modeling vocabulary it has also come to be used synonymously with nodes within the numerical/theoretical vocabulary of Fedem.

Link Regarding numerical algorithms a link is a superelement. The element is linear within its corotated coordinate system. This term also has its

background from the modeling side of Fedem and has since become a part of the theoretical vocabulary.

Model A model is defined as a more or less simplified representation of a system. The model must represent the properties of a system being studied, as accurate as possible.

Simulation Imitation of certain properties of a (mechanical) system in a computational model.

Kinematic simulation Kinematic simulation refers to calculations of motion in a system with no reference to forces and torques necessary to achieve this motion.

Dynamic simulation Dynamic simulation refers to the calculations of motion in a system where both constraint forces and forces necessary to drive the system are taken into account.

Multi discipline dynamic simulation The multi discipline dynamic simulation concept refers to a simulation model where the overall system is modeled as a mechanism, the parts are modeled as Finite Element substructures, and combined with a possible control system, all is integrated in the same simulation model. A control system may include controllers, actuators and sensors (to model feedback loops).

System simulation The term system simulation incorporates the three terms *kinematic simulation*, *dynamic simulation* and *multi discipline dynamic simulation* described above.

Chapter 2 Fundamentals

2.1 Notation

Matrices and matrix vectors (one column matrices), are denoted in text as upright bold symbols, such as $\mathbf{K}$ and $\mathbf{v}$. Tensors are denoted in italicized bold symbols, such as $\boldsymbol{a} = a_i \boldsymbol{i}_i$.

Given two coordinate systems with the three orthonormal base vectors $(\boldsymbol{I}_1\boldsymbol{I}_2\boldsymbol{I}_3)$ and $(\boldsymbol{i}_1\boldsymbol{i}_2\boldsymbol{i}_3)$, system $\boldsymbol{I}_i$ is called the *global system*, whereas system $\boldsymbol{i}_i$ is called the *local system*.

Using *tensor notation*, a vector (first order tensor) $\boldsymbol{a}$ can be represented as

$$\begin{aligned} \boldsymbol{a} &= a_1\boldsymbol{I}_1 + a_2\boldsymbol{I}_2 + a_3\boldsymbol{I}_3 = \sum_{i=1}^{3} a_i\boldsymbol{I}_i = a_i\boldsymbol{I}_i \\ &= \tilde{a}_1\boldsymbol{i}_1 + \tilde{a}_2\boldsymbol{i}_2 + \tilde{a}_3\boldsymbol{i}_3 = \sum_{i=1}^{3} \tilde{a}_i\boldsymbol{i}_i = \tilde{a}_i\boldsymbol{i}_i \end{aligned} \tag{2.1}$$

with respect to the two coordinate systems.

Using *matrix notation*, this same vector will be represented as $\mathbf{a}$ and $\tilde{\mathbf{a}}$ in the global and local coordinate systems respectively:

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad \text{and} \quad \tilde{\mathbf{a}} = \begin{bmatrix} \tilde{a}_1 \\ \tilde{a}_2 \\ \tilde{a}_3 \end{bmatrix} \tag{2.2}$$

2.2 Rigid-body motion

The position and orientation in space of a rigid body, $\mathcal{S}$, can be represented by the position and orientation of a fixed coordinate system. This rigid body can be a link or a superelement. This section addresses the rigid-body motion of the superelement.

The vector $\mathbf{s}$ represents the position of the fixed coordinate system in relation to the global coordinate system. The fixed coordinate system is defined by the three orthonormal base vectors $(\boldsymbol{i}_1, \boldsymbol{i}_2, \boldsymbol{i}_3)$, which comprise the

rotation matrix:

$$\text{Position:} \qquad \mathbf{s} \;=\; \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} \tag{2.3}$$

$$\text{Orientation:} \qquad \mathbf{R}_S \;=\; \begin{bmatrix} \mathbf{i}_1 & \mathbf{i}_2 & \mathbf{i}_3 \end{bmatrix} = \begin{bmatrix} i_{1x} & i_{2x} & i_{3x} \\ i_{1y} & i_{2y} & i_{3y} \\ i_{1z} & i_{2z} & i_{3z} \end{bmatrix} \tag{2.4}$$

It is frequently necessary to describe the motion within the rigid body $\mathcal{S}$ in, for example, joint attachment points. Such points have a position relative to the rigid body coordinate system. The joint itself has an orientation relative to the rigid body's orientation. Defining the point $\mathcal{N}$ (a FE node, for example) within the rigid body with coordinates $\tilde{\mathbf{a}}$ relative to the body's fixed coordinate system, a local coordinate system, $\tilde{\mathbf{R}}_N$, relative to the body-fixed system is represented by

$$\text{Position:} \qquad \tilde{\mathbf{a}} \;=\; \begin{bmatrix} \tilde{a}_x \\ \tilde{a}_y \\ \tilde{a}_z \end{bmatrix} \tag{2.5}$$

$$\text{Orientation:} \qquad \tilde{\mathbf{R}}_N \;=\; \begin{bmatrix} \tilde{\mathbf{i}}_1 & \tilde{\mathbf{i}}_2 & \tilde{\mathbf{i}}_3 \end{bmatrix} = \begin{bmatrix} \tilde{i}_{1x} & \tilde{i}_{2x} & \tilde{i}_{3x} \\ \tilde{i}_{1y} & \tilde{i}_{2y} & \tilde{i}_{3y} \\ \tilde{i}_{1z} & \tilde{i}_{2z} & \tilde{i}_{3z} \end{bmatrix} \tag{2.6}$$

The point $\mathcal{N}$ has a motion in the global system produced as a result of the motion of the body-fixed coordinate system. With $\mathbf{a}$ being the coordinates of the point relative to the global system, we have

$$\mathbf{a} \;=\; \mathbf{R}_S\tilde{\mathbf{a}} + \mathbf{s} \tag{2.7}$$

The orientation of the point in the local coordinate system, $\tilde{\mathbf{R}}_N$, will then be as follows in the global system:

$$\mathbf{R}_N \;=\; \mathbf{R}_S\tilde{\mathbf{R}}_N \tag{2.8}$$

The above equations can be combined by using 4×4 transformation matrices containing the position and rotation as defined by equations (2.3) and (2.4)

$$\mathbf{P}_S \;=\; \begin{bmatrix} \mathbf{R}_S & \mathbf{s} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} i_{1x} & i_{2x} & i_{3x} & s_x \\ i_{1y} & i_{2y} & i_{3y} & s_y \\ i_{1z} & i_{2z} & i_{3z} & s_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

and

$$\tilde{\mathbf{P}}_N \;=\; \begin{bmatrix} \tilde{\mathbf{R}}_N & \tilde{\mathbf{a}} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \tilde{i}_{1x} & \tilde{i}_{2x} & \tilde{i}_{3x} & \tilde{a}_x \\ \tilde{i}_{1y} & \tilde{i}_{2y} & \tilde{i}_{3y} & \tilde{a}_y \\ \tilde{i}_{1z} & \tilde{i}_{2z} & \tilde{i}_{3z} & \tilde{a}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

The position and orientation of point $\mathcal{N}$ in the global system can then be expressed as

$$\mathbf{P}_N \;=\; \mathbf{P}_S \tilde{\mathbf{P}}_N \;=\; \begin{bmatrix} \mathbf{R}_N & \mathbf{a} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.11}$$

Remark: *In computer implementations, the* 4×4 *transformation matrices are represented as* 3×4 *matrices obtained by deleting the fourth of the* 4×4 *matrix as in. The matrix multiplication of equation* (2.11) *is then modified by taking into account the implicitly defined* [0 0 0 1] *fourth row of the matrix.*

2.3 Finite rotations

2.3.1 Spin of a matrix

The cross product of vectors is written as

$$\boldsymbol{c} \;=\; \boldsymbol{a} \times \boldsymbol{b} \tag{2.12}$$

when using tensor notation. Using matrix notation this can be written

$$\mathbf{c} \;=\; \widehat{\mathbf{a}}\,\mathbf{b} \tag{2.13}$$

where

$$\widehat{\mathbf{a}} \;=\; \begin{bmatrix} 0 & a_z & -a_y \\ -a_z & 0 & a_x \\ a_y & -a_x & 0 \end{bmatrix} \quad \text{when} \quad \mathbf{a} \;=\; \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{2.14}$$

The above equations define the *Spin* of a vector. This notation is frequently used in the matrix expressions in the following.

2.3.2 Rotation of a vector

Any rotation of a vector from its original direction $\mathbf{a}$ to its rotated direction $\mathbf{a}'$ can be described by the general relationship

$$\mathbf{a}' \;=\; \mathbf{R}\mathbf{a} \tag{2.15}$$

where $\mathbf{R}$ is a *orthonormal* 3×3 matrix. A matrix is orthonormal when each row and column represented as a vector is orthogonal to all the other rows and columns of the matrix, and has a unit length. Since $\mathbf{R}$ is orthonormal, the inverse relationship becomes

$$\mathbf{a} \;=\; \mathbf{R}^{-1}\mathbf{a}' \;=\; \mathbf{R}^{T}\mathbf{a}' \tag{2.16}$$

A rotation matrix $\mathbf{R}$ can be established in a number of ways, referred to as different *parameterizations* of the rotation. The rotation matrix is also often called rotation *tensor*, as it has properties of a second order tensor.

2.3.3 Rodriguez parameterization

Rodriguez parameterization states that any combination of rotations can be represented by a single rotation θ about a single rotational axis defined by a unit vector $\boldsymbol{n}$. The vector is defined as

$$\boldsymbol{\theta} \;=\; \theta\mathbf{n} \tag{2.17}$$

and the rotation matrix can be written as

$$\mathbf{R} \;=\; \mathbf{R}(\boldsymbol{\theta}) \;=\; \mathbf{I} + \frac{\sin\theta}{\theta}\widehat{\boldsymbol{\theta}} + \frac{1}{2}\left(\frac{\sin\frac{\theta}{2}}{\frac{\theta}{2}}\right)^{2}\widehat{\boldsymbol{\theta}}^{2} \tag{2.18}$$

2.3.4 Variation of Rodriguez parameterization

The variation of the instantaneous rotation axis $\boldsymbol{\omega}$ with respect to the finite rotations $\boldsymbol{\theta}$ of the Rodriguez parameterization can be written as

$$\delta\omega \;=\; \frac{\partial\boldsymbol{\omega}}{\partial\boldsymbol{\theta}}\delta\boldsymbol{\theta} \;=\; \mathbf{H}(\boldsymbol{\theta})\delta\boldsymbol{\theta} \tag{2.19}$$

where

$$\mathbf{H}(\boldsymbol{\theta}) \;=\; \frac{1}{\theta^{2}}\boldsymbol{\theta}\boldsymbol{\theta}^{T} + \frac{\sin\theta}{\theta}\left(\mathbf{I} - \frac{1}{\theta^{2}}\boldsymbol{\theta}\boldsymbol{\theta}^{T}\right) + \frac{1-\cos\theta}{\theta^{2}}\widehat{\boldsymbol{\theta}} \tag{2.20}$$

Remark: *Attention should be paid to the* $0/0$ *terms of equation* (2.20) *when calculating* $\mathbf{H}(\theta)$ *for very small values of* θ.

2.3.5 Euler angles parameterization

The Euler angle parameterization actually refers to several parameterizations of a finite rotation. The total rotation is defined by a series of rotations about subsequent follower axes. To establish the relationship for the follower axes we first establish a relationship for subsequent rotations about rigid (global) axes.

Sequential rotations about rigid axes

A vector $\mathbf{a}_0$ is first rotated an angle θ_x about the global X-axis to obtain vector $\mathbf{a}_1$. The vector $\mathbf{a}_1$ is then rotated an angle θ_y about the global Y-axis to obtain vector $\mathbf{a}_2$. Finally, the vector $\mathbf{a}_2$ is rotated an angle θ_z about the global Z-axis to obtain vector $\mathbf{a}_3$. This yields the equations:

$$\mathbf{a}_1 = \mathbf{R}_x\mathbf{a}_0 \tag{2.21}$$

$$\mathbf{a}_2 = \mathbf{R}_y\mathbf{a}_1 = \mathbf{R}_y\mathbf{R}_x\mathbf{a}_0 = \mathbf{R}_{Rxy}\mathbf{a}_0 \tag{2.22}$$

$$\mathbf{a}_3 = \mathbf{R}_z\mathbf{a}_2 = \mathbf{R}_z\mathbf{R}_y\mathbf{R}_x\mathbf{a}_0 = \mathbf{R}_{Rxyz}\mathbf{a}_0 \tag{2.23}$$

where $\mathbf{R}_x$, $\mathbf{R}_y$, and $\mathbf{R}_z$ are established from equation (2.18) by inserting $\boldsymbol{\theta}^T = [\theta_x\ 0\ 0]$, $\boldsymbol{\theta}^T = [0\ \theta_y\ 0]$ and $\boldsymbol{\theta}^T = [0\ 0\ \theta_x]$, respectively.

2

Sequential rotations about follower axes

When rotating a vector $\mathbf{a}_0$ with rotation matrix $\mathbf{R}_x$, a new coordinate system consisting of the global axes rotated with matrix $\mathbf{R}_x$, can be obtained.

Remark: *The columns of* $\mathbf{R}_x$ *form the basis vectors of the new coordinate system.*

As vector $\mathbf{a}_0$ rotates with the coordinate system to form vector $\mathbf{a}_1$, the vector has not changed in the co-rotated coordinate system:

$$\tilde{\mathbf{a}}_1 = \mathbf{a}_0 \tag{2.24}$$

where superimposed ˜ is used to designate a vector in the local (co-rotated) coordinate system of $\mathbf{R}_x$.

A second rotation of the vector $\mathbf{a}_1$, an angle θ_y about the new local y-axis, is established in the co-rotated coordinate system

$$\tilde{\mathbf{a}}_2 = \mathbf{R}_y\tilde{\mathbf{a}}_1 = \mathbf{R}_y\mathbf{a}_0 \tag{2.25}$$

where matrix $\mathbf{R}_y$ is identical to the matrix established in equation (2.22). Transforming the vector $\tilde{\mathbf{a}}_2$ back to the global coordinate system can be achieved by the relationship

$$\mathbf{a}_2 = \mathbf{R}_x\tilde{\mathbf{a}}_2 = \mathbf{R}_x\mathbf{R}_y\mathbf{a}_0 = \mathbf{R}_{Fxy}\mathbf{a}_0 \tag{2.26}$$

A third rotation about the new z-axis of the second updated coordinate system is established in the local coordinate system

$$\tilde{\tilde{\mathbf{a}}}_\mathbf{3} = \mathbf{R}_\mathbf{z}\tilde{\tilde{\mathbf{a}}}_\mathbf{2} = \mathbf{R}_\mathbf{z}\mathbf{a}_\mathbf{0} \quad \text{since} \quad \tilde{\tilde{\mathbf{a}}}_\mathbf{2} = \tilde{\mathbf{a}}_\mathbf{1} = \mathbf{a}_\mathbf{0} \tag{2.27}$$

where superimposed $\tilde{}$ is used to designate a vector in the local (co-rotated) coordinate system of $\mathbf{R}_{Fxy}$.

Transforming $\tilde{\tilde{\mathbf{a}}}_{\mathbf{3}}$ back to the global coordinate system is performed in steps that reverse the intermediate local coordinate systems

$$\tilde{\mathbf{a}}_3 = \mathbf{R}_y\tilde{\tilde{\mathbf{a}}}_{\mathbf{3}} = \mathbf{R_y R_z a_0} \tag{2.28}$$

$$\mathbf{a}_3 = \mathbf{R}_x\tilde{\mathbf{a}}_3 = \mathbf{R}_x\mathbf{R}_y\mathbf{R}_z\mathbf{a}_0 = \mathbf{R}_{Fxyz}\mathbf{a}_0 \tag{2.29}$$

Comparing equations (2.23) and (2.29), the sequential rotations about follower axes are obtained by reversing the order of the matrix multiplication in relation to rigid axis rotation and forming the resultant rotation matrix:

$$\text{Rigid (fixed) axis } X\text{-}Y\text{-}Z \text{ rotation:} \quad \mathbf{R}_{Rxyz} = \mathbf{R}_z\mathbf{R}_y\mathbf{R}_x \tag{2.30}$$

$$\text{Follower axis } X\text{-}Y\text{-}Z \text{ rotation:} \quad \mathbf{R}_{Fxyz} = \mathbf{R}_x\mathbf{R}_y\mathbf{R}_z \tag{2.31}$$

Euler angles

To define a number of Euler angle parameterizations, the axis and the sequence in which to perform the rotations are determined. The most common parameterizations use all axes in the sequences X-Y-Z, Y-Z-X, and Z-X-Y, and their reverse-sequenced alternatives. Two axis combinations such as X-Y-X are also possible. These parameterizations are easily established by substitutions into the expression (2.31).

This gives the 12 different parameterizations of Table 2.1. The Euler angle parameterization used in Fedem is Z-Y-X.

2.3.6 Euler angles extraction

Section 2.3.5 describes how to obtain an incremental rotation matrix, $\mathbf{R}$, given three Euler angles θ_x, θ_y and θ_z. In some cases we would also like to perform

Table 2.1: Euler Angle Parameterizations

Three axis combinations:	X-Y-Z, Z-Y-X,	Y-Z-X, X-Z-Y,	Z-X-Y, Y-X-Z
Two axis combinations:	X-Y-X, Y-Z-Y, Z-X-Z,	Y-X-Y, Z-Y-Z, X-Z-X	

the inverse transformation, that is, given an (incremental) rotation matrix, find the corresponding Euler angles.

Assuming we use the Euler Z-Y-X parameterization, we start by creating the rotation matrix that rotates a vector an angle $\theta_z = \alpha$ about the Z-axis:

$$\mathbf{R}_z = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.32}$$

Next, create the rotation matrix to rotate a vector an angle $\theta_y = \beta$ about the Y-axis:

$$\mathbf{R}_y = \begin{bmatrix} \cos\beta & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{2.33}$$

Then, create the rotation matrix rotating a vector an angle $\theta_x = \gamma$ about the X-axis:

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \tag{2.34}$$

Multiplying these matrices together we get the single rotation matrix:

$$\mathbf{R} = \mathbf{R}_z\mathbf{R}_y\mathbf{R}_x = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \tag{2.35}$$

Letting r_{ij} denote the individual elements of this matrix, we can now derive the Euler angles as follows:

$$\theta_z = \alpha = \arctan\frac{r_{21}}{r_{11}} \tag{2.36}$$

$$\theta_y = \beta = \arctan\frac{-r_{31}}{\sqrt{r_{11}^2 + r_{21}^2}} \tag{2.37}$$

$$\theta_x = \gamma = \arctan\frac{r_{32}}{r_{33}} \tag{2.38}$$

Remark: *In a computer implementation of the above equations, one should use the intrinsic math function* `atan2(y,x)` *when evaluating* $\arctan\frac{y}{x}$*, such that the sign of the computed angles are correctly set according to which quadrant the arguments x and y implies.*

Chapter 3 Model Reduction

Dynamic analysis is generally more expensive than static analysis, because the solution involves repeated computations of the same form, whereas static analysis requires only a single computation. The introduction of Component Mode Synthesis (CMS) model reduction and superelement techniques, however, reduces the cost of dynamic computations.

The CMS reduction technique reduces the cost of the analysis by decreasing the number DOFs used. The fundamental assumption is that the low-frequency modes of vibration are the most important; in other words, the higher the frequency mode, the less significant it is in the dynamic analysis. It is also true that for any numerical model, and especially an FE model, the high-frequency modes are less accurate in comparison to underlying differential equation solutions.

The form of applied loads is important when deciding upon the required complexity of the FE models. If a harmonic load is applied and the steady-state response is required, a peak response is achieved if the load has a frequency component close to a resonant frequency. In this case, the accuracy of the FE model must be sufficient to represent the resonant frequencies over the entire range of the load's frequencies so that no significant resonant frequencies are missed.

If the loads are not periodic −as is the case, for example, in deployment and latching simulations−then there is no simple relationship between the structural resonant frequencies and the frequency content of the loads. However, there is an effective frequency within the loads defined by the rate of change from one state to another, that is, the rise time of the input. If the force changes rapidly, it has a short rise time and the input has a high-frequency content. This requires a relatively large number of DOFs in the model. If the force input changes relatively slowly (it has a long rise time), it will have a lower frequency content, and fewer modes are therefore required in the analysis.

The number of nodes in a mesh basically controls the number of DOFs in the model. When a dynamic model is constructed, the mesh used can reflect the results. For instance, if only the displacement or the first few resonant frequencies are required, a coarse mesh can be used. This mesh can be much coarser than the mesh used in an equivalent static stress analysis. If, however, dynamic stresses are required, a finer mesh density of the kind used for static

stressing is necessary. In this case, the mesh can be graduated in regions of stress concentration in exactly the same manner as in static stress analysis.

The most expensive dynamic computation is performed for impact and wave propagation models. Here, the short rise times of the forcing function require a relatively large number of DOFs. In addition, the large stress and displacement changes at the front of the wave need a fine mesh to model the wave front accurately. However, the wave front moves through the complete model as the wave propagates, so a uniformly fine mesh is necessary throughout the model to accurately address wave propagation problems.

There are various methods of condensing static models into smaller dynamic models automatically, but for the various reasons outlined above, these methods must be used with care because they are only applicable in certain cases, depending upon the required response. In some cases, typically when calculating dynamic stresses, the degree to which a model can be condensed depends partly upon the reduction process used, and the method of recovering stresses after the displacements have been determined. These two factors must be considered together when determining the degree of condensation to be employed within the analysis.

3.1 Review of model reduction methods

All reduction methods are based on the same principle: to reduce the effective number of DOFs in order to reduce computational costs. However, because these methods are approximate and different approximations can be used, the range of dynamic reduction methods is extensive. In addition, each of these methods has sub-variants.

Most of the dynamic model reduction techniques available can be classified into one of the following four categories:

- *Modal Reduction* −only a small number of the possible modes of vibration−typically those with lowest eigenfrequency−are used in the dynamic analysis.
- *Static Condensing (Guyan reduction)* −this method of reducing the effective number of DOFs is based entirely upon static considerations. Fedem uses this classic FE reduction technique when no additional component modes are specified. In static reduction, the nodes (and corresponding DOFs) are split into master/external and slave/internal nodes. Only master/external nodes are retained after the static reduction.

- *Dynamic Condensing* −the method of reducing the effective number of DOFs is based upon both static and dynamic considerations. The CMS reduction implemented in Fedem is one example of dynamic condensing.
- *Dynamic Substructuring* −the complete structure is formed by establishing a reduced set of equations for a set of components, or substructures, that comprises the total structure or mechanism. One of the techniques above is used for each substructure.

All reduction (or condensing) methods can be written in terms of a coordinate transformation of the form

$$\mathbf{v} = \mathbf{H}\,\mathbf{q} \tag{3.1}$$

where $\mathbf{v}$ is the full set of displacements of size n, $\mathbf{q}$ is the reduced or condensed set of displacements of size m where $m < n$, and $\mathbf{H}$ is a transformation matrix relating the two sets of coordinates. The various reduction techniques can then be viewed as different methods for constructing a suitable transformation matrix.

There are many ways to define the transformation matrix $\mathbf{H}$, but they are not all equally accurate for dynamic analysis. Various factors must be considered, and most methods of condensing the equations result in a compromise in the requirements. The first and most important considerations are that dynamic behavior is largely controlled by the lowest static and component vibration modes, and that any condensation method should allow the lower modes to be recovered with a smaller error than the higher ones.

Secondly, the higher vibration modes still contribute to the dynamic response. However, if the maximum frequency content of the acting loads is significantly lower than a particular resonant frequency, the contribution of that high resonance to the overall response will be in terms of a constant (static) value. This is illustrated in Figure 3.1. Mode 2 here has a substantially constant response up to the frequency ω, and its contribution to the overall response is defined almost entirely by its stiffness over this range. This is not the case for mode 1, which requires dynamic behavior because it varies considerably over the frequency range 0 to ω. If a frequency of 2ω or higher is necessary, mode 2 must also be modeled dynamically.

In general, the transformation matrix, $\mathbf{H}$, must be constructed in such a way that the high-frequency modes can be defined by their static response. Once the transformation has been defined, the equation of motion can be condensed from the full set of n equations to the smaller set, m, by means of the transformation matrix $\mathbf{H}$.

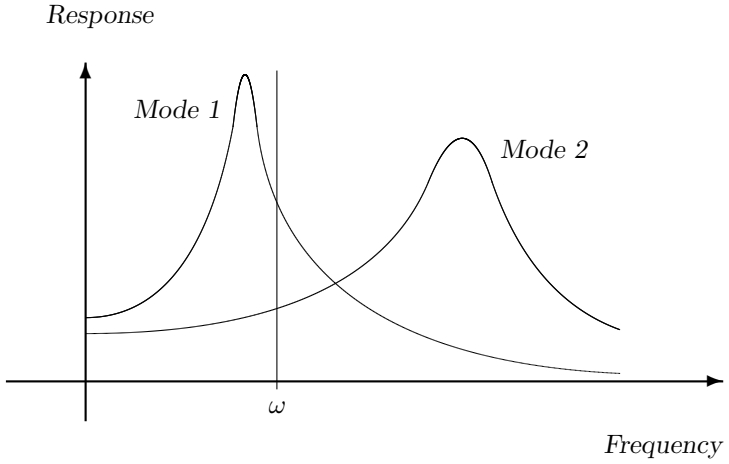


Figure 3.1: Individual modal response

In static analysis, a solution achieved by means of substructuring is exact. If the full structure is analyzed directly and then as a series of substructures, it is found that apart from round-off error the two solutions are identical. There is no approximation involved in static substructuring.

This is not the case with dynamic analysis. When the equations of any individual substructure are reduced in size, information regarding the dynamic response −that is, information regarding the higher resonant frequencies and mode shapes−is lost. The solution can be organized to preserve the stiffness but not the mass characteristics associated with the high-frequency modes. Thus, a static substructure solution is exact, but a dynamic substructure solution is only approximate.

3.1.1 Modal reduction

In general, modal reduction is only carried out when the set of modes used in dynamic analysis is reduced. In this case, each column in the transformation matrix, **H**, is an eigenvector of the system. The transformation to the reduced set of equations has the added computational advantage that the reduced stiffness and mass matrices are both diagonal. For the assumption of either proportional damping or modal damping, the reduced damping matrix is also diagonal. This is used to solve the forced response of the equations, as the reduced equations are a set of m uncoupled single-DOF equations. Proportional damping is described in Section 7.5.

In modal reduction, the contribution of all the modes included in the transformation is exact. The response, however, is not exact, even for a low-frequency input, because the static contributions of the high-frequency modes have not been included. This is not important if the excitation is near one of the low resonance frequencies, because the response of this mode will then dominate. However, at a distance from a resonance it can lead to significant error.

3.1.2 Static condensing (Guyan reduction)

A common form of dynamic reduction is static condensing or Guyan reduction. In this form of reduction, the stiffness properties of the structure are preserved at the expense of its dynamic properties. The nodes (and corresponding DOFs) are split into master/external and slave/internal nodes; only master DOFs are retained after reduction. This form of reduction is widely used, being simple and relatively inexpensive to carry out; however, it should be used with some caution.

For static condensing, a set of master/external DOFs are defined that become the **q** displacements in the transformation equations. The remaining displacements are called the slave/internal DOFs (and nodes), and these are eventually eliminated from the solution process. There are many more slave than master DOFs. The transformation matrix, **H**, is constructed by applying a unit displacement to each master DOF in turn, while setting all the other master DOFs to zero. The resulting slave displacements are used to define a column in **H**. This is repeated for each master in turn to construct the complete transformation matrix.

In Fedem, the master DOFs are defined implicitly by the selection of nodes for use as connection points between component FE models; these points are called Triads.

This process of constructing the transformation matrix preserves rigid body inertias and the static behavior of the structure. However, it only approximates the dynamic behavior, since the process implicitly assumes that inertia forces on the slave freedoms are zero. This has the effect of redistributing the mass of slave freedoms onto the masters (Triads), thereby altering the distribution of structural mass. It is this assumption and its consequences that introduce the errors into the Guyan reduction process. Errors can be minimized, but not eliminated, by selecting an optimum set of master DOFs. The selection process and resulting problems are the subjects of the following sections.

Choice of master DOFs

Within a static/Guyan reduction, the slave freedoms should be those that do not contribute much to the inertia forces in the dynamic response. Inertia forces are the product of mass and acceleration; slave freedoms, therefore, should be those points of low mass or low acceleration. Expressed differently, the master freedoms should be defined at points where the inertia forces are greatest; these correspond to points of high mass and low stiffness.

Problems of static reduction

It is very difficult to quantify the errors associated with static reduction. Generally, the error in a given resonant frequency or mode shape increases with eigenvalue number: the lower eigenvalues are predicted more accurately than the higher ones. However, this rule is not absolute, and occasionally some higher modes are predicted more accurately than lower ones. Any predicted mode shape can only be a linear combination of the columns of the transformation matrix, **H**. If **H** is deficient in the pattern of displacement required by a given mode shape, that mode will not be predicted accurately. If this is the case, the eigenvalues of the condensed system will be greater than those of the original uncondensed system, resulting in a stiffer system than that of the original.

The fewer the number of master freedoms (Triads), the greater the potential error. To minimize error, the user must choose a sufficient number of masters −preferably between two and ten times the number of required modes. However, even with a factor of ten it is very difficult to predict more than general consequences of the reduction. A choice of too many masters can increase computational costs.

The original finite element equation system is heavily banded, and its sparse nature enables efficient handling of large sets of equations. However, the reduced, transformed set is fully populated, so that although fewer equations are involved, they are much more densely packed and therefore more costly to solve. If the original set of equations were of size n and average bandwidth b, then the number of master freedoms, m, should satisfy $m^2 < bn$. If not, it is less expensive computationally to work with the original set of equations without any reduction.

3.1.3 Dynamic condensing

Fedem uses a dynamic condensing technique that combines modal reduction and static condensing. The disadvantage of modal reduction (apart from the cost of determining the eigenvectors) is that it does not preserve the static behavior of the structure, because the stiffness contribution from the modes that have been ignored is never reinstated (refer to Figure 3.1). Conversely, static condensing preserves static stiffness at the expense of dynamic effects. In dynamic condensing, the transformation matrix, **H**, is largely based on the eigenvectors associated with the lowest frequencies, as is the case with modal reduction; but these eigenvectors are augmented by the static modes associated with master freedoms, as in static condensation.

In dynamic condensing, static modes must be manipulated so that the stiffness associated with lower modes (stiffness accounted for by the presence of these modes) is not included twice. In Fedem, static modes are complemented by "fixed interface normal modes," and these normal modes (eigenvectors) are never included in the preserved static modes. Fixed interface modes are computed by eigenvector analysis of the structure with all the nodal DOFS (triad DOFS) fixed. A fixed DOF means defining its value as zero and (usually) removing it from the equation system.

3.1.4 Dynamic substructuring

Another method of improving the computational efficiency of a dynamic analysis is to use dynamic substructuring. However, its use also compromises accuracy. The structure or mechanism is divided into a series of smaller structures or substructures; in Fedem, these are called Links. The division into Links in Fedem is based on physical identification of the various mechanism components.

The transformation matrix in Fedem is defined in such a way as to preserve the original freedoms and takes the following form for each Link:

- $\mathbf{v}_e$ represents the freedoms that connect to other Links or applied springs, dampers, and loads.
- $\mathbf{v}_i$ represents the internal freedoms that are internal to a single Link.
- $\mathbf{q}$ represents the generalized freedoms, internal to the reduced Link, that are included to augment and improve the dynamic response (fixed interface normal/component modes).

It is important to note that with this definition of **H** the connection freedoms are preserved across the transformation, making assembly of the complete

mechanism relatively simple. However, this approach does place restrictions upon the method of transformation.

The simplest way of forming the subcomponent is to base the transformation upon Guyan reduction. Here the master freedoms are not chosen arbitrarily, but are defined at the connection nodes (Triads) between Links and the nodes at which springs, dampers, and loads are applied. There are no augmenting freedoms **q**, and the transformation matrix is identical to that of a Guyan reduction with the connection nodes/freedoms selected as the master freedoms.

Internal component modes in dynamic substructuring

The accuracy of the dynamic behavior of a substructure can be improved considerably by including extra terms in the augmenting freedoms, **q**, of the transformation matrix. These terms are usually based on some of the system eigenvectors, thus the process becomes very similar to that of the dynamic condensing method described above. There are many ways of including these terms, all of which lead to different variations of the dynamic substructure method.

It is difficult to say which is the best method, as they all produce similar results and their relative accuracy is dependent upon the specific problem being analyzed. The various forms of dynamic substructuring can be classified in one of two categories, depending on the way in which the eigenvectors are formed. The most direct method is to calculate the substructure eigenvectors with the connection nodes/freedoms fixed; this is the method employed in Fedem. The alternative method is to calculate the eigenvectors of the substructure with the connection freedoms free.

Fixed boundary dynamic substructure

If the eigenvectors of the substructure are calculated while the connection freedoms are held fixed, the coordinate transformation matrix **H** contains the first few eigenvectors of the fixed boundary substructure −the $\mathbf{\Phi}$ term in equation (3.13). The number of augmenting freedoms is then equal to the number of these fixed interface modes. The static modes can be taken directly from the Guyan reduction process with the connection freedoms as masters −the **B** term in equation (3.13). This means that the behavior of the substructure is defined by the static behavior of its connection freedoms and the internal dynamic behavior of its augmenting freedoms. The representation of the dynamic behavior of the substructure still includes some

approximations, but these become progressively less significant as more eigenvectors are added to the augmenting set.

This combined approach, which is the method used in Fedem, is referred to as the Component Mode Synthesis (CMS) method. The user can also add extra Triads, which need not be connection nodes, for the purpose of improving the dynamic behavior of the substructure.

Free boundary dynamic substructure

The alternative to the fixed boundary substructure is the free boundary form. In this case the connection freedoms, $\mathbf{v}_1$, are left free, and the transformation matrix is defined in terms of the eigenvectors. If the substructure has no natural supports, the eigenvectors will include the rigid body modes to represent the rigid body mass of the structure. The free boundary substructure generally requires more calculations to form the transformation matrix. This form of dynamic substructure ensures that the mass description of the component is correct; the stiffness description, however, will only be approximate. To recover correct stiffness behavior, at least for the connection freedoms, the dynamic modes can be augmented by static modes[1].

3.1.5 Summary

Fedem uses a state-of-the-art dynamic condensing method, Component Mode Synthesis (CMS), which combines static and fixed-interface normal modes. Fedem can also increase accuracy by including an additional distribution of master DOFs (Triads). The CMS method is also well suited to flexible mechanism analysis because it preserves the effective masses and inertias.

Depending on complexity and modeling detail, the finite element model of a substructure can consist of a large or small number of FE nodes. These nodes are divided into internal and external nodes. The external nodes are defined during mechanism modeling as the connection points for joints, springs, dampers, external loads, control input, points of interest, and so on. These nodes are retained as so-called "supernodes" during the reduction of the substructure to a superelement.

[1] There are some indications that free boundary dynamic substructure methods are more accurate than fixed boundary methods, depending upon the manner of implementation for each method and the type of problem being solved.

3.2 Component mode synthesis reduction

For each substructure modeled, the status codes for the substructure DOFs are set to 1 for internal DOFs and 2 for external DOFs. With this control information, the system mass and stiffness matrices are assembled from the corresponding element matrices. The first n_1 DOFs of the system matrices are internal substructure DOFs, and the next n_2 DOFs are external.

In symbolic form, the substructure matrices for mass and stiffness can be divided into submatrices, with index i for internal and e for external as follows:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{ii} & \mathbf{M}_{ie} \\ \mathbf{M}_{ei} & \mathbf{M}_{ee} \end{bmatrix} \quad \text{and} \quad \mathbf{K} = \begin{bmatrix} \mathbf{K}_{ii} & \mathbf{K}_{ie} \\ \mathbf{K}_{ei} & \mathbf{K}_{ee} \end{bmatrix} \tag{3.2}$$

where $\mathbf{M}_{xx}$ are mass and $\mathbf{K}_{xx}$ are stiffness submatrices.

The stiffness relation for the substructure may now be expressed as:

$$\begin{bmatrix} \mathbf{K}_{ii} & \mathbf{K}_{ie} \\ \mathbf{K}_{ei} & \mathbf{K}_{ee} \end{bmatrix} \begin{bmatrix} \mathbf{v}_i \\ \mathbf{v}_e \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_i \\ \mathbf{Q}_e \end{bmatrix} \tag{3.3}$$

where $\mathbf{v}_x$ is the displacement vector and $\mathbf{Q}_x$ is the load vector.

3.2.1 Static modes

The first line of equation (3.3) may be written:

$$\begin{aligned} \mathbf{v}_i &= \mathbf{K}_{ii}^{-1}\mathbf{Q}_i - \mathbf{K}_{ii}^{-1}\mathbf{K}_{ie}\mathbf{v}_e \\ &= \mathbf{v}_i^i + \mathbf{v}_i^e \end{aligned} \tag{3.4}$$

where $\mathbf{v}_i^i$ represents internal displacements with fixed external DOFs, and $\mathbf{v}_i^e$ represents internal displacements as a function of external displacements. Examination of each term provides

$$\mathbf{v}_i^i = \mathbf{K}_{ii}^{-1}\mathbf{Q}_i \tag{3.5}$$

and

$$\mathbf{v}_i^e = -\mathbf{K}_{ii}^{-1}\mathbf{K}_{ie}\mathbf{v}_e = \mathbf{B}\mathbf{v}_e \tag{3.6}$$

where

$$\mathbf{B} = -\mathbf{K}_{ii}^{-1}\mathbf{K}_{ie} \tag{3.7}$$

The set of nodal displacements $[\ \mathbf{v}_i^e \ \ \mathbf{v}_e\ ]$ represents the "exact" solution of static loads and boundary conditions acting on the external nodes ("exact" in the sense that the model reduction has not in any way changed the solution in relation to the solution of the full system).

3.2.2 Constrained dynamic modes

The substructure matrices can be reduced by CMS transformation to produce a set of modes called *Craig Bampton modes*. This is done by eliminating the internal DOFs as system DOFs, and replacing them with a limited number of substructure vibration modes called generalized DOFs. The CMS transformation starts with an eigenvalue analysis of the substructure system matrices with the external DOFs fixed at zero. The equation for free, undamped vibration of the substructure's internal DOFs is written:

$$\mathbf{M}_{ii}\ddot{\mathbf{v}}_i^i + \mathbf{K}_{ii}\mathbf{v}_i^i \;=\; \mathbf{0} \tag{3.8}$$

Considering simple harmonic motion, the displacement $\mathbf{v}_i^i$ may be expressed as:

$$\mathbf{v}_i^i \;=\; \boldsymbol{\phi}\,\sin\omega t \tag{3.9}$$

where the eigenvector $\boldsymbol{\phi}$ is defined by the eigenvalue problem.

$$\left(\mathbf{K}_{ii} - \omega^2\mathbf{M}_{ii}\right)\boldsymbol{\phi} \;=\; \mathbf{0} \tag{3.10}$$

In a substructure with n active DOFs of which p are external DOFs, the internal displacements v_i^i can be expressed as:

$$\mathbf{v}_i^i \;=\; \sum_{k=1}^{S} \boldsymbol{\phi}_k\, y_k \;=\; \boldsymbol{\Phi}\mathbf{y} \tag{3.11}$$

where $s < n - p$, and

$$\boldsymbol{\Phi} \;=\; \left[\begin{array}{cccc} \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_s \end{array}\right] \tag{3.12}$$

is the eigenvector matrix and has dimensions $(n - p) \times s$.

3.2.3 Reduced system

The superelement displacements are now expressed by the external DOFs $\mathbf{v}_e$ and by the new generalized DOFs $\mathbf{y}$:

$$\mathbf{v} \;=\; \left[\begin{array}{c} \mathbf{v}_e \\ \mathbf{v}_i \end{array}\right] = \left[\begin{array}{cc} \mathbf{I} & \mathbf{0} \\ \mathbf{B} & \boldsymbol{\Phi} \end{array}\right] \left[\begin{array}{c} \mathbf{v}_e \\ \mathbf{y} \end{array}\right] = \mathbf{H} \left[\begin{array}{c} \mathbf{v}_e \\ \mathbf{y} \end{array}\right] \tag{3.13}$$

Usually only a few of the lowest modes of vibration need to be included to achieve good results, and this may reduce the size of the problem substantially. If all eigenmodes are included, $s = n - p$, the CMS transformation is exact.

Chapter 4 Co-rotated Formulation

During mechanism simulation, elements may undergo significant translational and rotational displacements. Elements in this context may be superelements, generic parts, or more "standard" elements as described in Chapter A. These large displacements can have both rigid body and deformational components. To obtain these components from the total displacements, a method called *Shadow Element Configuration* is used, see Figure 4.1. In this figure, the configurations are denoted as follows:

C_0 – initial (reference) configuration

C_{0n} – co-rotated shadow element configuration

C_n – current (deformed) configuration

C_0 is the initial, undeformed configuration, and C_n is the deformed position at step n. The shadow element configuration, C_{0n}, is obtained by translating and rotating the initial configuration C_0 as a rigid body in such a way that the "distance" between C_{0n} and C_n is minimized.

This rigid body motion moves element nodes from their initial position within the undeformed configuration, C_0, to their position in the shadow

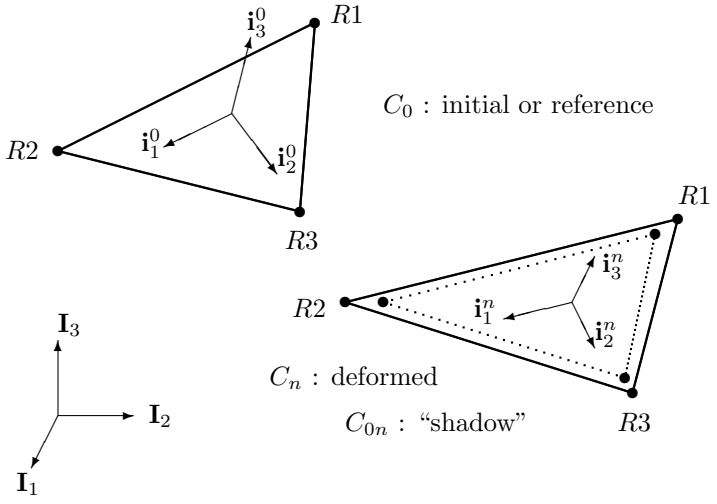


Figure 4.1: Coordinate systems and configurations for an element

element configuration, C_{0n}, whereas the deformational displacement moves the nodes from their position in the shadow element configuration, C_{0n}, to their positions in the deformed configuration C_n.

The local coordinate system of configuration C_0 moves and rotates with the element and becomes the co-rotated coordinate system of C_{0n}. In addition, because the superelement matrices established in Chapter 3 were defined in the element's local coordinate system, the matrices remain constant with respect to this co-rotated coordinate system. Thirdly, the co-rotated coordinate system is common to both C_{0n} and C_n.

When establishing the system matrices as described in Chapter 7, the superelement matrices defined in Chapter 3 must be transformed to the global coordinates for each new position of the mechanism during simulation. This transformation to global coordinates is the subject of the next section.

4.1 Local element coordinate system

The element matrices are all expressed in a local element coordinate system. For superelements this is the coordinate system in which the finite element mesh was modeled at substructure level, and for generic parts this is simply the local element coordinate system of the underlying rigid spider. In the system analysis, this local coordinate system becomes the *Element Co-rotated Coordinate System* (ECCS) because it moves with the element.

Regarding the nonlinear behavior of an element, the co-rotational formulation is independent of whether the element is a reduced superelement, resulting from the Component Mode Synthesis reduction described in Section 3.2, a generic part element as described in Section A.16, or a more traditional element as the other ones described in Appendix A. In the following description of the co-rotational formulation, element and superelement are thus interchangeable terms.

To calculate the transformations of the elements at each new position, the directions for the corresponding ECCS must first be calculated based on the positions of the element nodes.

4.1.1 Method 1: Best fit of max sized triangle

In this method, the ECCS is connected to three nodal points for each element. The goal is to select among all nodal points within an element, those three that form the largest triangle. This is achieved by the following algorithm:

1. The first reference point, R_1, is selected as the node furthest away from the centroid of all element nodes. This first reference point is always at the node itself, i.e., zero offset.
2. The second reference point, R_2, is selected as the node furthest away from the point R_1. If the element only has one node the same node is used again, but with the second reference point defined by an offset of unit length along the global x-axis from the nodal position.
3. The third reference point, R_3, is selected as the node furthest from a line through the points R_1 and R_2. If the element has only one or two nodes, or the third node is on (or close to) this straight line[1], the reference point is defined using an offset from the third node so that the three points form a proper triangle[2]. The direction of the third-node offset vector is obtained by rotating the vector $\vec{R}_{12}$ from R_1 to R_2 90 degrees about either the global x-, y- or z-axis, depending on along which global axis direction the $\vec{R}_{12}$ vector has its smallest component.
4. A temporary coordinate system is now formed based on the positions of the three reference points. The origin is defined at R_1, the x-axis is defined in the direction from R_1 to R_2, the z-axis is next calculated from the cross product of the x-axis and the axis extending from R_1 to R_3, and finally the y-axis is calculated from the cross product of the z- and x-axes. During the analysis, this coordinate system is recalculated for each new position of the three reference points.
5. The position of the triangle system is always fixed in relation to the ECCS during the analysis.

[1]The criterion for whether the third node is on the straight line between R_1 and R_2 can be configured. Originally, this tolerance was absolute and equal to 0.001 regardless of the model dimensions. Later, it was changed to half the distance between the first two points, i.e., $0.5|\vec{R}_{12}|$ but it is now configurable by setting the relative tolerance parameter ϵ in the absolute tolerance $\epsilon|\vec{R}_{12}|$. The default value on ϵ is now 0.05.

[2]The offset on the third node (when needed) was originally selected simply as a vector of unit length away from the straight line. This is clearly not a good strategy if the distance between the first two nodes is much greater (or much less) than 1, since the resulting triangle then will be far from even-sided. These shortcomings of the unit offset spurred an improved method where the offset is scaled with respect to the distance between the first two points. This results in a more even sided triangle, and a more robust coordinate system. The original method is available for backward compatibility of older models.

Using the 4×4 position matrix notation defined in equation (2.9), the position matrix, $\mathbf{P}_{TE}$, relates the triangle coordinate system to the element coordinate system:

$$\mathbf{P}_{TE} = \begin{bmatrix} \mathbf{R}_{TE} & \mathbf{p}_1 \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.1}$$

where $\mathbf{R}_{TE}$ is the rotation or orientation of the triangle system measured in the element system, and $\mathbf{p}_1$ is the position of the triangle system relative to the element system ($\mathbf{p}_1$ is the position of R_1 measured in the element coordinate system).

It follows that the positions of the reference points measured in global coordinates establish $\mathbf{P}_{TG}$, the position of the triangle coordinate system in relation to the global system. The following relationship then relates the triangle, element and global coordinate systems:

$$\mathbf{P}_{TG} = \mathbf{P}_{EG}\mathbf{P}_{TE} \tag{4.2}$$

where $\mathbf{P}_{EG}$ is the element coordinate system relative to the global system, and is stated

$$\mathbf{P}_{EG} = \mathbf{P}_{TG}\mathbf{P}_{TE}^{-1} \tag{4.3}$$

4.1.2 Method 2: Mass based weighted average

This method ultimately gives an expression requiring a weighted average of the nodal discrepancies between the actual deformed element and the undeformed shadow element to be zero. This expression can be established from an equilibrium point of view.

One assumes that the (rigid) shadow element is attached to the deformed element with springs, rotational and translational, at each node. The spring stiffnesses at each node is selected to be, for instance, proportional to the mass for translation and proportional to the inertia for the rotation of each node. When the nodes move, the shadow element will reposition itself according to equilibrium based on the redistribution of nodal displacements. The rigid body motion of the shadow element is then represented by the motion of a fictitious node at the Center of Gravity (CG) of the shadow element.

The equilibrium equations for the shadow element can be established about CG. This yields six equations for the movement of the shadow element as a function of the nodal displacements. The equations end up being non-linear and have to be solved iteratively.

With k_t as the mass-proportional translational stiffness in all three directions at each node, and k_r as inertia-proportional rotational stiffness in

all three directions, the stiffness of a spring element with two co-located nodes i and j is given by

$$\begin{bmatrix} \mathbf{f}_i \\ \mathbf{m}_i \\ \mathbf{f}_j \\ \mathbf{m}_j \end{bmatrix} = \begin{bmatrix} k_t\mathbf{I} & \mathbf{0} & -k_t\mathbf{I} & \mathbf{0} \\ \mathbf{0} & k_r\mathbf{I} & \mathbf{0} & -k_r\mathbf{I} \\ -k_t\mathbf{I} & \mathbf{0} & k_t\mathbf{I} & \mathbf{0} \\ \mathbf{0} & -k_r\mathbf{I} & \mathbf{0} & k_r\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{v}_i \\ \boldsymbol{\theta}_i \\ \mathbf{v}_j \\ \boldsymbol{\theta}_j \end{bmatrix} \tag{4.4}$$

With node j being a dependent node rigidly attached to the node at CG, one can establish the virtual displacement equation

$$\mathbf{v}_j \;=\; \mathbf{E}_i\mathbf{v}_{CG} \tag{4.5}$$

or

$$\begin{bmatrix} \mathbf{v}_j \\ \boldsymbol{\theta}_j \end{bmatrix} = \begin{bmatrix} \mathbf{1} & -\widehat{\mathbf{e}}_i \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{v}_{CG} \\ \boldsymbol{\theta}_{CG} \end{bmatrix} \quad \text{where} \quad \mathbf{e}_i = \begin{bmatrix} x_i - x_{CG} \\ y_i - y_{CG} \\ z_i - z_{CG} \end{bmatrix} \tag{4.6}$$

Using the kinematic relationship above in a virtual work expression, one can establish the stiffness matrix for the spring between node i and the CG node as

$$\begin{bmatrix} \mathbf{f}_i \\ \mathbf{m}_i \\ \mathbf{f}_{CG} \\ \mathbf{m}_{\mathrm{CG}} \end{bmatrix} = \begin{bmatrix} \mathbf{k}_t & \mathbf{0} & -\mathbf{k}_t & \mathbf{k}_t\widehat{\mathbf{e}} \\ \mathbf{0} & \mathbf{k}_r & \mathbf{0} & -\mathbf{k}_r \\ -\mathbf{k}_t & \mathbf{0} & \mathbf{k}_t & -\mathbf{k}_t\widehat{\mathbf{e}} \\ \widehat{\mathbf{e}}^T\mathbf{k}_t & -\mathbf{k}_r & -\widehat{\mathbf{e}}^T\mathbf{k}_t & (\mathbf{k}_r + \widehat{\mathbf{e}}^T\mathbf{k}_T\widehat{\mathbf{e}}) \end{bmatrix} \begin{bmatrix} \mathbf{v}_i \\ \boldsymbol{\theta}_i \\ \mathbf{v}_{CG} \\ \boldsymbol{\theta}_{\mathrm{CG}} \end{bmatrix} \tag{4.7}$$

where $\mathbf{k}_t = k_t\mathbf{I}$ and $\mathbf{k}_r = k_r\mathbf{I}$.

The full stiffness matrix for the shadow element is then formed by the assembly of all nodal contributions. Rewriting the equation above with a slightly different notation:

$$\begin{bmatrix} \mathbf{f}_1^i \\ \mathbf{f}_2^i \end{bmatrix} = \begin{bmatrix} \mathbf{k}_{11}^i & \mathbf{k}_{12}^i \\ \mathbf{k}_{21}^i & \mathbf{k}_{22}^i \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^i \\ \mathbf{v}_2^i \end{bmatrix} \tag{4.8}$$

the assembly for all nodes from 1 to n (where $n+1$ is the CG node) gives the shadow element stiffness matrix:

$$\begin{bmatrix} \mathbf{f}_1^1 \\ \vdots \\ \mathbf{f}_1^i \\ \vdots \\ \mathbf{f}_{CG} \end{bmatrix} = \begin{bmatrix} \mathbf{k}_{11}^1 & \cdots & \mathbf{0} & \cdots & \mathbf{k}_{12}^1 \\ \vdots & \ddots & & & \vdots \\ \mathbf{0} & & \mathbf{k}_{11}^i & & \mathbf{k}_{12}^i \\ \vdots & & & \ddots & \vdots \\ \mathbf{k}_{21}^1 & \cdots & \mathbf{k}_{21}^i & \cdots & \left(\sum_{i=1}^{n} \mathbf{k}_{22}^i\right) \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^1 \\ \vdots \\ \mathbf{v}_1^i \\ \vdots \\ \mathbf{v}_{CG} \end{bmatrix} \tag{4.9}$$

The equilibrium equation at the CG node is seen as the last 6 rows of the assembled shadow element stiffness matrix, which can be rewritten as

$$\mathbf{f}_{CG} = \sum_{i=1}^{n} \mathbf{k}_{21}^{i} \mathbf{v}_i \quad \text{or} \quad \begin{bmatrix} \mathbf{f}_{CG} \\ \mathbf{m}_{\mathrm{CG}} \end{bmatrix} = \sum_{i=1}^{n} \begin{bmatrix} -\mathbf{k}_t^i & \mathbf{0} \\ \hat{\mathbf{e}}^T \mathbf{k}_t^i & -\mathbf{k}_r^i \end{bmatrix} \begin{bmatrix} \mathbf{v}_i \\ \boldsymbol{\theta}_i \end{bmatrix} \tag{4.10}$$

If the above equation does not yield zero forces and moments $\mathbf{f}_{CG}$ at the CG node when the deformational displacement vector is used, the shadow element is given a corrective translation and rotation that will again alter the deformation vector for all the nodes in according to equation (4.6), such that the unbalanced forces and moments vanish.

Equilibrium of the CG node gives

$$\mathbf{f}_{CG} = \sum_{i=1}^{n} \mathbf{k}_{21}^{i} \mathbf{v}_i = \sum_{i=1}^{n} \mathbf{k}_{21}^{i} \mathbf{E}_i \, \Delta \mathbf{v}_{CG} \tag{4.11}$$

Solving this with respect to increment in Shadow element position $\Delta \mathbf{v}_{CG}$ gives

$$\Delta \mathbf{v}_{CG} = \left(\sum_{i=1}^{n} \mathbf{k}_{21}^{i} \mathbf{E}_i \right)^{-1} \sum_{i=1}^{n} \mathbf{k}_{21}^{i} \mathbf{v}_{d_i} \tag{4.12}$$

The equation above is solved repeatedly until the deformational displacement vector $\mathbf{v}_d$ produces no update to the shadow element position given by $\Delta \mathbf{v}_{CG}$.

Chapter 5 Flexible Connections

5.1 Spring elements

A spring may be linear with a spring constant, or it may be nonlinear with the spring stiffness explicitly or implicitly defined by a reference to a function. The stiffness function is then a function of the spring deflection. The spring stiffness and associated force can also be scaled by another function that can be a function of any model variable. Axial springs are specified between two supernodes in the mechanism model, while joint springs are specified directly on the joint DOFs.

The actual stiffness expressions depend on whether the stiffness function is defined explicitly or implicitly, i.e.,

$$K_{el}(\delta,\cdot) \;=\; \begin{cases} S(\cdot)\,[\,k_0 + f(\delta)\,] & : f(\delta) \text{ is explicit (stiffness function)} \\ S(\cdot)\left[k_0 + \frac{df}{d\delta}\right] & : f(\delta) \text{ is implicit (force function)} \end{cases} \tag{5.1}$$

where

K_{el} : Elastic spring stiffness

S : Scale function

k_0 : Constant stiffness

f : Explicit or implicit stiffness function

δ : Spring deflection, $\delta = l - l_0$, where l_0 is the stress-free length

The associated spring force is defined by the expression

$$F_{el}(\delta,\cdot) \;=\; \begin{cases} S(\cdot)\left[k_0\delta + \int_0^{\delta} f(\hat{\delta})d\hat{\delta}\right] & : f(\delta) \text{ is explicit} \\ S(\cdot)\,[\,k_0\delta + f(\delta)\,] & : f(\delta) \text{ is implicit} \end{cases} \tag{5.2}$$

where F_{el} is the elastic spring force, and the other quantities are as above.

Remark: *Although the scale function $S(\cdot)$ in equations* (5.1) *and* (5.2) *in principle may be a function of any model variable, you should avoid using one that directly or indirectly depends on the spring deflection, δ, such as the spring velocity, $\dot{\delta}$, for instance. If such a variable is used, there might be an inconsistency between the spring stiffness and associated force that may lead to instabilities in the solution, or poor convergence. The best is just to let it be some explicit function of time, t.*

For joint springs, the stiffness is added directly to the diagonal of the system matrix on the actual DOF involved. For axial springs, the stiffness and force act in the direction between the two end nodes. Since an axial spring may be rotating, geometric stiffness terms must also be included, such that the total stiffness matrix of the axial spring becomes

$$\mathbf{k} \;=\; K\,\boldsymbol{i}_1 \otimes \boldsymbol{i}_1 \;+\; \frac{F}{l}\,(\boldsymbol{i}_2 \otimes \boldsymbol{i}_2 + \boldsymbol{i}_3 \otimes \boldsymbol{i}_3) \tag{5.3}$$

where K and F are the current stiffness and force in the spring, defined through equations (5.1) and (5.2) respectively, and l is the current spring length. The local coordinate system of the spring is defined through the basis $\{\boldsymbol{i}_1, \boldsymbol{i}_2, \boldsymbol{i}_3\}$ where $\boldsymbol{i}_1$ goes in the length direction. The operator $\otimes$ denotes a dyadic product (outer product) between two vectors. The 3×3 stiffness matrix (5.3) is then added to the global system stiffness matrix on the DOFs associated with the two connected nodes.

A spring may be used as a passive element with fixed stress-free length/angle. However, it may also be used as an active drive element in the mechanism where the stress-free length varies as a function of time in accordance with an explicitly defined function.

By specifying springs on joint DOFs, special constraint effects may be imposed, for instance the free joint may be used in this way to model nonlinear bearings or rubber bushings. Effects like end stops, backlash in gears, switches, temperature-dependent stiffness or other nonlinear effects may easily be modeled in this way.

5.1.1 Failure and yield properties

The nonlinear behavior in a spring may also be modeled by specifying certain failure and/or yield criteria. Failure is modeled by using equations (5.1) and (5.2), only as long as the following conditions are satisfied

$$\begin{aligned} \delta_{f-} &< \delta \;\; < \delta_{f+} \\ F_{f-} &< F_{el} < F_{f+} \end{aligned} \tag{5.4}$$

where δ_{f-} and δ_{f+} define the maximum spring deflection before the spring fails, on compression and tension, respectively, whereas F_{f-} and F_{f+} are similar failure limits on the spring force.

When any of the conditions (5.4) are no longer fulfilled, the spring state is changed to *failed* and it will no longer contribute to the system stiffness matrix and force vector of the mechanism. This has the same effect as

choosing a scale function in the stiffness and force expression (5.1) and (5.2) equal to a step function

$$S(t) = \begin{cases} 1 & : \quad t < t_f \\ 0 & : \quad t \geq t_f \end{cases} \tag{5.5}$$

where t_f is the time at which the failure occurred.

When a joint spring reaches a failure state, this would normally cause an abrupt redistribution of the forces in the joint, and in many cases one would like to signal the failure in the other springs in the same joint too, when one of its joint springs detects failure.

Hysteresis behavior and permanent deflection after unloading can be introduced in a spring by specifying a yield criterion, which limits the spring force to a specified maximum value F_{y+} (on tension), or minimum value F_{y-} (on compression). The actual stiffness and force in the spring will then be given by

$$K(\delta, \cdot) = \begin{cases} K_{el}(\delta, \cdot) & : F_{y-} \leq F_{el} \leq F_{y+} \\ 0 & : F_{el} < F_{y-} \quad \text{or} \quad F_{el} > F_{y+} \end{cases} \tag{5.6}$$

$$F(\delta, \cdot) = \min\left\{\max\left\{F_{el}(\delta, \cdot), F_{y-}\right\}, F_{y+}\right\} \tag{5.7}$$

where K_{el} and F_{el} are the elastic stiffness and force, as defined by equations (5.1) and (5.2), respectively.

When the spring is yielding, i.e., $F_{el} \notin [F_{y-}, F_{y+}]$, the "elastic" deflection of the spring (the deflection used when evaluating the stiffness function) is taken as $\delta = l - (l_0 + \delta_y)$, where δ_y is the *yield deflection* defined through

$$\delta_y(t) = \int_{t_y}^{t} \frac{F_{el} - F}{K_{el}} d\hat{t} \tag{5.8}$$

where t_y is the time at which the yielding first started, and F is the actual spring force defined by equation (5.7). When the spring re-enters an elastic state due to unloading, the integrand of equation (5.8) becomes zero such that δ_y then remains constant until the yield limit is reached again.

The yield limits, F_{y-1} and F_{y+1} may be either constant values or functions of another model variable. The latter may be used to to model clutch-like behavior in joint springs, where you can smoothly (or abruptly) engage/disengage the motion constraint enforced by the spring.

5.1.2 Interconnected spring elements

Springs with a nonlinear force–displacement curve can be used to model certain typical joint characteristics. A "trailer-hitch" connection has typically a certain play where it has no stiffness or force resisting displacement. This can be modeled using a spring with a stiffness–displacement curve where we have a zero stiffness "play" area around zero displacement. Alternatively, this is achieved with a corresponding force–displacement curve. However, we seek to have this play and nonlinear force–displacement behavior to be independent of the displacement direction, when the displacement is taken in an arbitrary radial direction. With constant stiffness and independent springs in, for instance, the X and Y directions we will achieve this, but when the stiffness is nonlinear the force–displacement response gets a distinct "square" characteristic when subjecting the hitch or joint to a rotating force. This can be circumvented by interconnecting the displacements and force–displacement curves in the two directions.

The interconnected displacement is defined as

$$\bar{\delta}^i = \sqrt{\sum_j B_{ij}\delta_j^2} \quad \text{where } B_{ij} = \begin{cases} 1 : i = j \\ 1 : \text{DOF } i \text{ and } j \text{ are interconnected} \\ 0 : \text{otherwise} \end{cases} \tag{5.9}$$

The interpolated displacement $\bar{\delta}^i$ is used for calculating an interpolated force based on all the individual DOF springs defined according to equation (5.2)

$$\bar{F}^i = \sum_j \text{sign}(\delta_j) N_{ij} F_j(\bar{\delta}^i \,\text{sign}(\delta_j)) \tag{5.10}$$

where

$$N_{ij} = \left(\frac{\delta_j}{\bar{\delta}^i}\right)^2 \qquad \text{and} \qquad \text{sign}(\delta_j) = \begin{cases} 1 & : \quad \delta_j \geq 0 \\ -1 & : \quad \delta_j < 0 \end{cases} \tag{5.11}$$

We are using an interpolation of the individual spring functions in order to handle oval characteristics defined by different stiffness functions for, for instance, the X and Y DOFs. The interpolated force value $\bar{F}^i$ is then decomposed along each of the interconnected DOFs as

$$\bar{F}_j^i = c_{ij}\bar{F}^i \qquad \text{with direction cosines} \qquad c_{ij} = \frac{\delta_j}{\bar{\delta}^i} \tag{5.12}$$

Remark: *You should avoid using the interconnectivity functionality between springs with constant stiffness. No additional cylindrical or spherical behavior is achieved by this, while reduced stability can be experienced due to* 0/0 *terms in equation* (5.12) *when displacements are close to zero and springs are interconnected.*

The (material) stiffness terms for the interconnected springs are obtained in a similar fashion as the force components, i.e.

$$\bar{K}^{Mi} = \sum_j N_{ij} K_j (\bar{\delta}^i \operatorname{sign}(\delta_j)) \tag{5.13}$$

where N_{ij} is defined in equation (5.11). The components of the material stiffness matrix $\bar{\mathbf{K}}^{Mi}$ can then be written as

$$\bar{K}^{Mi}_{jk} = c_{ij} c_{ik} \bar{K}^{Mi} \tag{5.14}$$

Since the direction of the interpolated spring force, $\bar{F}^i$, depends on the current solution state through the direction cosines c_{ij}, geometric stiffness terms are needed in addition to the material stiffness (5.14), for the translational components. Let $\boldsymbol{i}_1$ denote the current direction for the interconnected spring, i.e., $\boldsymbol{i}_1 = \{c_{i1}, c_{i2}, c_{i3}\}^T$ and $\boldsymbol{i}_2$ and $\boldsymbol{i}_3$ being two perpendicular unit vectors such that $\{\boldsymbol{i}_1, \boldsymbol{i}_2, \boldsymbol{i}_3\}$ forms an orthonormal basis. Assuming the stress-free length of the spring is zero, the total tangent stiffness matrix for an interconnected translatory spring is then given as

$$\bar{\mathbf{K}}^i = \bar{\mathbf{K}}^{Mi} + \bar{\mathbf{K}}^{Gi} \quad \text{where} \quad \bar{\mathbf{K}}^{Gi} = \frac{\bar{F}^i}{\bar{\delta}^i} (\boldsymbol{i}_2 \otimes \boldsymbol{i}_2 + \boldsymbol{i}_3 \otimes \boldsymbol{i}_3) \tag{5.15}$$

The equation (5.15) holds for translatory springs coupled in all three coordinate directions (flexible ball joint behavior). If only two of the three coordinate directions are coupled (flexible revolute joint behavior), we have

$$\bar{\mathbf{K}}^{Gi} = \frac{\bar{F}^i}{\bar{\delta}^i} \boldsymbol{i}_2 \otimes \boldsymbol{i}_2 \tag{5.16}$$

where $\boldsymbol{i}_2 = \boldsymbol{i}_3 \times \boldsymbol{i}_1$, and $\boldsymbol{i}_3$ is the unit vector in the coordinate direction that is not coupled (rotation axis of the flexible revolute joint).

5.1.3 Global spring elements

Instead of using an axial spring which has the important property that it contributes to stress stiffening in the overall system through its geometric stiffness, an alternative is to specify independent spring properties on one or several of the global DOFs between two nodes in the system, or between one node and ground. This is much the same as using a free joint with all DOFs spring-constrained. However, in the global spring element the joint variable transformation outlined in Section 6.2.6 is not performed and the resulting

stiffness coefficients are added directly into the system stiffness matrix for the DOFs associated with the two nodes in question.

The updated spring lengths in a global spring element, that are needed for the computation of the stiffness and force in each DOF, are defined as

$$\mathbf{l} = [l_x\ l_y\ l_z]^T = \mathbf{x}_2 - \mathbf{x}_1 \tag{5.17}$$

$$\boldsymbol{\alpha} = [\alpha_x\ \alpha_y\ \alpha_z]^T = \text{EulerZYX}(\mathbf{R}_1^T\mathbf{R}_2) \tag{5.18}$$

where $\mathbf{x}_i$ is the updated global position vector of node i, and $\mathbf{R}_i$ is the orthonormal transformation matrix defining its orientation. The operator EulerZYX$(\cdot)$ extracts the Euler Z-Y-X angles from the provided orthonormal transformation matrix, as outlined in Section 2.3.6. The corresponding spring deflections are then taken as $\boldsymbol{\delta} = \mathbf{l} - \mathbf{l}_0$ and $\boldsymbol{\theta} = \boldsymbol{\alpha} - \boldsymbol{\alpha}_0$, with $\mathbf{l}_0$ and $\boldsymbol{\alpha}_0$ being the initial stress free lengths.

> **Remark:** *When rotational stiffness is assigned to a global spring, it is imperative that the orientations of the two nodes are initially identical or close to each other, and that they are unlikely to exhibit large relative rotations. The background for this is that the Euler angle extraction operator is valid for small rotations only. Actually, the operator is singular for certain rotation states (i.e., when one have 0/0-situations in some of the equations* (2.36)–(2.38)*. For the same reason, using soft global springs with rotational stiffness may also render the solution unstable and should be avoided.*

Especially in situations when the distance between the two nodes is large, global spring elements may yield a more stable solution than using a regular free joint, since a small rotation change in one end then may correspond to a large translation change at the other end, due to the long arm. The global spring element is a system level equivalent to the BUSH element on the FE link level (see Section A.10) but with the important distinction that a global spring may account for nonlinear spring properties.

5.2 Damper elements

A damper may be linear with a damping constant, or it may be nonlinear with the damping coefficient defined by a reference to a function. The damping coefficient is a function of the damper velocity. The damping coefficient and associated force can also be scaled by another function that can be a function of any model variable. A wide range of energy dissipation effects may be modeled by damper elements. Axial dampers are specified between two supernodes in the mechanism model, while joint dampers are specified directly on the joint DOFs.

The actual damping coefficient expression depend on whether the damping function is defined explicitly or implicitly

$$C(v,\cdot) \;=\; \begin{cases} S(\cdot)\,[\,C_0 + f(v)] & : f(v) \text{ is explicit (damping function)} \\ S(\cdot)\left[C_0 + \frac{df}{dv}\right] & : f(v) \text{ is implicit (force function)} \end{cases} \tag{5.19}$$

where

C : Damping coefficient

S : Scale function

C_0 : Constant damping

f : Explicit or implicit damping coefficient function

v : Damper velocity

The associated damper force is defined by the expression

$$F(v,\cdot) \;=\; \begin{cases} S(\cdot)\left[C_0 v + \int\limits_0^v f(\hat{v})d\hat{v}\right] & : f(v) \text{ is explicit} \\ S(\cdot)\,[\,C_0 v + f(v)\,] & : f(v) \text{ is implicit} \end{cases} \tag{5.20}$$

where F is the damper force or moment, and the other quantities are as above.

5

Remark: *Although the scale function $S(\cdot)$ in equations (5.19) and (5.20) in principle may be a function of any model variable, you should avoid using one that directly or indirectly depends on the damper velocity, c, such as the damper deflection, for instance. If such a variable is used, there will be an inconsistency between the damping coefficient and associated force that may lead to solution instabilities or poor convergence. The best is just to let it be some explicit function of time, t.*

For joint dampers, the damping coefficient is added directly to the diagonal of the system damping matrix on the actual DOF involved. Axial dampers are handled in a similar way as axial springs, that is, they result both in an damping matrix and geometric stiffness matrix to be added into the corresponding system matrices for the nodal DOFs involved:

$$\mathbf{c} \;=\; C\,\boldsymbol{i}_1 \otimes \boldsymbol{i}_1 \tag{5.21}$$

$$\mathbf{k}_g \;=\; \frac{F}{l}\,(\boldsymbol{i}_2 \otimes \boldsymbol{i}_2 + \boldsymbol{i}_3 \otimes \boldsymbol{i}_3) \tag{5.22}$$

where C and F are the current damping coefficient and damper force defined through equations (5.19) and (5.20) respectively, and l is the damper length.

By specifying dampers on joint DOFs, special constraint effects may be imposed, for instance the free joint may be used in this way to model damping in nonlinear bearings or rubber bushings. Effects like switches, temperature dependent damping or other nonlinear effects may be modeled in this way.

5.3 Spring-constrained joints

All joint variables for the master-slave based joints, such as revolute joints, ball joints, prismatic joints, etc., may be constrained by springs. The free joint has no degree of freedom constraint and is usually the basis for modeling spring-constrained joints. The reason for making joints spring-constrained can be to model the flexibility of an actual bearing or model nonlinear effects like clearings, rubber bushings, etc. To be able to easily switch between master-slave constrained joints and spring-constrained joints, the position and orientation of triads for spring-constrained joints should be the same as for the corresponding master-slave constrained joint.

The *flexible revolute joint* is usually modeled from a free joint with relative constraining springs on the x, y and z translational variables and on the x and y rotational joint variables. The rotation about the z-axis will be the joint variable. The stiffness of the springs on the different joint variables should comply with the actual translational and rotational stiffness of the bearing. Nonlinear stiffness like in rubber bushings or in bearings with clearing should be modeled by nonlinear springs. The translatory springs in the local x and y directions may need to be interconnected, as described in Section 5.1.2, in order to achieve equal behavior for any load direction in the joint.

The *flexible ball joint* will also usually be based on a free joint and with constraining springs for the x, y and z translational variables. The stiffness for the springs will be set according to joint stiffness and possible nonlinearities as for the revolute joint above. All three translatory springs may need to be interconnected, as described in Section 5.1.2, in order to achieve equal behavior for any load direction in the joint.

The *flexible prismatic joint* will usually be modeled from one or more free joints. The z-axis of the joint triads should be along the direction of the joint variable, the joint axis. If the flexible prismatic joint is based on only one free joint, all joint variables should have constraining springs except the variable along the joint axis, that is the z direction of the joint. If two or more free joints are used, the orientation of the corresponding triads should be the same and with the z-direction along the joint axis. For the free joints constituting the flexible prismatic joint, the joint variables in x and y directions and rotation about z should have constraining springs. The rotation variables about the x and y directions for the actual free joints will usually have no constraining springs, see the master-slave based prismatic joint. Translational clearing in the joint could be modeled by nonlinear constraining springs for x and/or y translation and rotational clearing about the z-axis is modeled by nonlinear constraining springs for the z rotation.

The *flexible cylindric joint* is modeled in a very similar way as the flexible prismatic joint described above. However, there is no constraining spring for the z rotation. This joint will have two joint variables, the translation along and the rotation about the joint axis that is the z-axis of the triads.

Besides the options for modeling constraints mentioned above, you have almost unlimited options for modeling different constrained motions. However, you should be aware of what effects could result from the fact that rotations in space do not commute, see Section 2.3.5.

Chapter 6 Modeling of Joints

6.1 Master and Slave based Joint Modeling

A joint is a way of specifying a constrained relative motion between two bodies or links in a mechanism. In the finite element modeling of mechanism joints, superelement nodes are used to specify the joint constraints. A joint between two moving links, denoted 1 and 2, will in general be defined by one node on link 1, called the slave triad, and one or more nodes on link 2, called the master triad(s). The constraints are then modeled by making all DOFs of the slave triad dependent on all DOFs of the master triad(s), and DOFs of the joint itself.

6.2 Single-master Joints

The position of the slave triad relative to the master triad is given by

$$\mathbf{P}_{SG} = \mathbf{P}_{MG}\mathbf{P}_{JT}\mathbf{P}_{N}\cdots\mathbf{P}_{1}\mathbf{P}_{SJ} \tag{6.1}$$

where the interpretation of each term is as follows:

- $\mathbf{P}_{SG}$ Position of Slave triad measured in the Global coordinate system. The position is ultimately a function of the DOFs of the master triad and the joint DOFs.
- $\mathbf{P}_{MG}$ Position of Master triad measured in the Global coordinate system. This matrix is a function of the master triad DOFs and is thus subject to variation.
- $\mathbf{P}_{JT}$ Position of the Joint measured in the master Triad coordinate system. This matrix handles a possible offset of the joint relative to the master triad, as well as joint orientation possibly different from that of the master triad. The matrix is constant (carries no DOFs) and is thus not subject to variation.
- $\mathbf{P}_{i}$ Joint variable matrices, numbered from 1 to N. Each matrix can have from 1 to 6 DOFs, or joint variables (three translations and three rotations). When all joint variables are zero, each of the matrices $\mathbf{P}_{i}$ becomes unity, and equation (6.1) simplifies to $\mathbf{P}_{SG} = \mathbf{P}_{MG}\mathbf{P}_{JT}\mathbf{P}_{SJ}$.

$\mathbf{P}_{SJ}$ Position of the Slave triad measured in the Joint coordinate system, when all joint variables are zero. This matrix handles a possible offset of the joint relative to the slave triad, as well as joint orientation possibly different from that of the slave triad. This matrix is also constant, and is thus not subject to variation.

Note that the position of the joint itself, in the global system, is given by

$$\mathbf{P}_{JG} = \mathbf{P}_{MG}\mathbf{P}_{JT} \tag{6.2}$$

The N joint variable matrices $\mathbf{P}_i$ can be thought of as describing the relative motion between $N+1$ bodies. With one single joint variable matrix the two bodies will be the master- and slave link, respectively (slave link as body 1, and master link as body 2). With two or more joint variable matrices the intermediate bodies are of course purely fictitious, although a very useful mental concept. The joint variables associated with joint variable matrix $\mathbf{P}_i$ are expressed in the coordinate system of body $i+1$. According to (6.1), this gives the following coordinate system, $\delta\tilde{\mathbf{u}}_1 = \mathbf{0}$, for these joint variables:

$$\mathbf{R}_{C_i} = \mathbf{R}_{MG}\mathbf{R}_{JT}\mathbf{R}_N \cdots \mathbf{R}_{i+1} \quad \text{and} \quad \mathbf{R}_{C_N} = \mathbf{R}_{MG}\mathbf{R}_{JT} \tag{6.3}$$

Variation of equation (6.1) provides an expression for the slave DOFs as a linear combination of the free master DOFs, i.e.

$$\begin{aligned} \delta\mathbf{P}_{SG} &= \delta\mathbf{P}_{MG}\mathbf{P}_{JT}\mathbf{P}_N \cdots \mathbf{P}_1\mathbf{P}_{SJ} \\ &+ \mathbf{P}_{MG}\mathbf{P}_{JT}\delta\mathbf{P}_N \cdots \mathbf{P}_1\mathbf{P}_{SJ} \\ &\vdots \\ &+ \mathbf{P}_{MG}\mathbf{P}_{JT}\mathbf{P}_N \cdots \delta\mathbf{P}_1\mathbf{P}_{SJ} \end{aligned} \tag{6.4}$$

since $\delta\mathbf{P}_{JT} = \mathbf{0}$ and $\delta\mathbf{P}_{SJ} = \mathbf{0}$ according to the definitions above.

The first term of equation (6.4) provides a linear coupling to the master triad DOFs, and can be expressed as

$$\begin{bmatrix} \delta\mathbf{u}_s \\ \delta\boldsymbol{\omega}_s \end{bmatrix}_M = \begin{bmatrix} \mathbf{I} & \widehat{\mathbf{e}}_{SM} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \delta\mathbf{u}_m \\ \delta\boldsymbol{\omega}_m \end{bmatrix} \tag{6.5}$$

where $\mathbf{e}_{SM}$ is the position of the slave triad relative to the master triad, measured in global system. For simplicity, it is here assumed that both the master and slave triad have DOFs in global directions.

The variation with respect to the joint variables of a given joint variable

matrix $\mathbf{P}_i$, can be expressed as

$$\begin{aligned}\begin{bmatrix} \delta\mathbf{u}_s \\ \delta\boldsymbol{\omega}_s \end{bmatrix}_i &= \begin{bmatrix} \mathbf{I} & \widehat{\mathbf{e}}_{Si} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{C_i} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{C_i} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_i \end{bmatrix} \begin{bmatrix} \delta\tilde{\mathbf{u}}_i \\ \delta\tilde{\boldsymbol{\theta}}_i \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_{C_i} & \widehat{\mathbf{e}}_{Si}\mathbf{R}_{C_i}\mathbf{H}_i \\ \mathbf{0} & \mathbf{R}_{C_i}\mathbf{H}_i \end{bmatrix} \begin{bmatrix} \delta\tilde{\mathbf{u}}_i \\ \delta\tilde{\boldsymbol{\theta}}_i \end{bmatrix}\end{aligned} \tag{6.6}$$

where $\mathbf{e}_{Si}$ is the position of the slave triad relative to body $i+1$, measured in global system. The matrix $\mathbf{R}_{C_i}$ defines the coordinate system of joint variable matrix i according to equation (6.3). The rotation gradient matrix $\mathbf{H} = \frac{\partial\boldsymbol{\omega}}{\partial\boldsymbol{\theta}}$ is given by equation (2.20). Note that in case of only one rotational DOF for joint variable matrix $\mathbf{P}_i$, this simplifies (in effect) to $\mathbf{H}_i = \mathbf{I}$.

6.2.1 Revolute Joint

A revolute joint is modeled from two supernodes located on different links in a mechanism, see Figure 6.1. The joint position matrix $\mathbf{P}_{JT}$ orients the joint coordinate system to have local z-axis along the axis of relative rotation. The joint has one joint variable matrix with a single DOF, $\tilde{\theta}_z$:

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{MG}\mathbf{P}_{JT}\mathbf{P}_1(\tilde{\theta}_z)\mathbf{P}_{SJ} \tag{6.7}$$

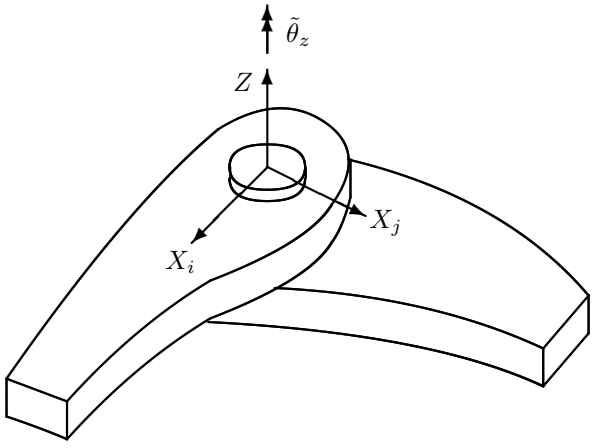


Figure 6.1: Coordinate axis and variable of revolute joint.

Optionally, the joint may also have free motion in the z-direction, in which case it has the additional DOF $\tilde{w}$:

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{MG}\mathbf{P}_{JT}\mathbf{P}_1(\tilde{w},\tilde{\theta}_z)\mathbf{P}_{SJ} \tag{6.8}$$

The joint produces six constraint equations, one for each slave triad DOF. Variation of the default revolute joint, equation (6.7), will be that of equations (6.5) and (6.6) with the simplifications $\mathbf{H}_1 = \mathbf{I}$, $\delta\tilde{\mathbf{u}}_1 = \mathbf{0}$, and $\delta\tilde{\boldsymbol{\theta}}_1 = [0\ 0\ \delta\tilde{\theta}_z]^T$. With the additional freedom in z as in equation (6.8), the only change is $\delta\tilde{\mathbf{u}}_1 = [0\ 0\ \delta\tilde{w}]^T$.

6.2.2 Universal Joint

A universal joint consists of three bodies: Input Shaft, Center Cross, and Output Shaft. It is thus modeled with two joint variable matrices to describe the relative motion between the three bodies (the center cross body is only implicitly defined through the joint formulation):

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{MG}\mathbf{P}_{JT}\mathbf{P}_2(\tilde{\theta}_z)\mathbf{P}_1(\tilde{\theta}_y)\mathbf{P}_{SJ} \tag{6.9}$$

The matrix $\mathbf{P}_2(\tilde{\theta}_z)$ here describes the relative motion (rotation) between the master body and the center cross. The rotation axis is the z-axis of the joint coordinate system. The matrix $\mathbf{P}_1(\tilde{\theta}_y)$ describes the rotation between the center cross an the slave body. The rotation axis is here the y-axis of the cross coordinate system. The y-axis of the cross coordinate system is initially coincident with the y-axis of the joint coordinate system. The joint coordinate system will normally be that of the master triad.

6.2.3 Constant Velocity Joint

A constant velocity joint is modeled using the fact that two universal joints connected via an intermediate shaft will give a constant rotational velocity, provided that each joint absorbs half of the total angle between the input and output shaft. This gives a joint formulation with 4 joint variable matrices, and two constraint equations that ensure the equal shaft-axis angle between the two fictitious universal joints:

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{MG}\mathbf{P}_{JT}\mathbf{P}_4(\tilde{\theta}_{z_4})\mathbf{P}_3(\tilde{\theta}_{y_3})\mathbf{P}_2(\tilde{\theta}_{y_2})\mathbf{P}_1(\tilde{\theta}_{z_1})\mathbf{P}_{SJ} \tag{6.10}$$

with the additional linear constraint equations

$$\tilde{\theta}_{z_4} = \tilde{\theta}_{z_1} \qquad \text{and} \qquad \tilde{\theta}_{y_3} = \tilde{\theta}_{y_2} \tag{6.11}$$

6.2.4 Ball Joint

A ball joint is modeled from two supernodes located on different links in a mechanism, see Figure 6.2. The default ball joint formulation has Euler angles $Z - Y - X$ as joint DOFs:

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{MG}\mathbf{P}_{JT}\mathbf{P}_3(\tilde{\theta}_z)\mathbf{P}_2(\tilde{\theta}_y)\mathbf{P}_1(\tilde{\theta}_x)\mathbf{P}_{SJ} \tag{6.12}$$

In addition, the ball joint is available with components of the rotation axis as joint DOFs:

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{MG}\mathbf{P}_{JT}\mathbf{P}_1(\tilde{\theta}_z, \tilde{\theta}_y, \tilde{\theta}_x)\mathbf{P}_{SJ} \tag{6.13}$$

The formulation with the rotation axis parameterization may be used if the default formulation runs into singularities with respect to the Euler angles. If singularity arises for the Euler angles, one may also try rotating the joint (different initial orientation).

6.2.5 Rigid Joint

A rigid joint (see Figure 6.3) contains no joint variable matrices since there is no relative motion between the master and slave bodies. The joint allows a rigid coupling between two nodes that need not be coincident, nor is there a limitation on common directions in space:

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{MG}\mathbf{P}_{JT}\mathbf{P}_{SJ} \tag{6.14}$$

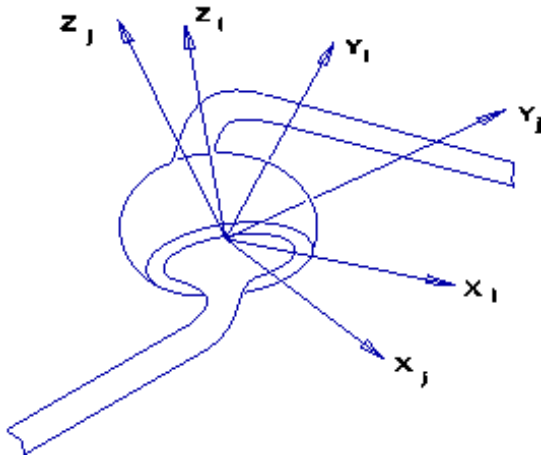


Figure 6.2: Coordinate system i and j of a ball joint

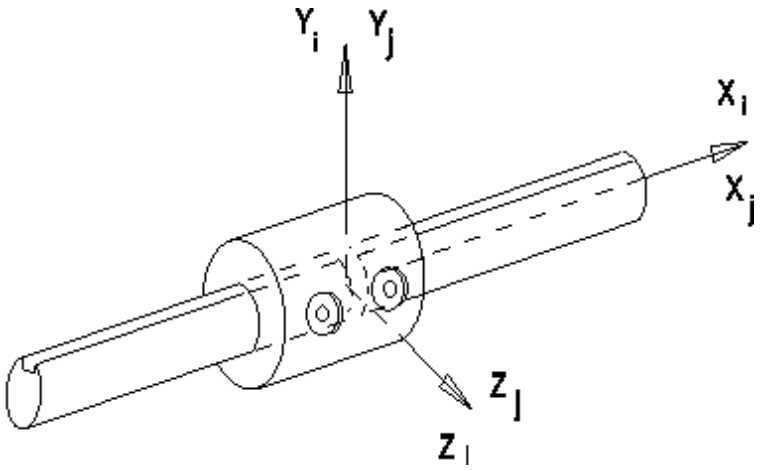


Figure 6.3: Coordinate axis of rigid joint.

The rigid joint is often used for accessing internal forces through the joint at system level during simulation.

6.2.6 Free Joint

The free joint (see Figure 6.4) is modeled from two supernodes located on ground or different links in a mechanism. There is full freedom of motion between the links, provided by six joint variables. The three translational joint variables are in the coordinate system of the joint itself, and the three rotational joint variables are Euler angles $Z - Y - X$. This gives the joint formulation as

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{MG}\mathbf{P}_{JT}\mathbf{P}_3(\tilde{u},\tilde{v},\tilde{w},\tilde{\theta}_z)\mathbf{P}_2(\tilde{\theta}_y)\mathbf{P}_1(\tilde{\theta}_x)\mathbf{P}_{SJ} \tag{6.15}$$

The free joint is also available with components of the rotation axis as joint DOFs:

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{MG}\mathbf{P}_{JT}\mathbf{P}_1(\tilde{u},\tilde{v},\tilde{w},\tilde{\theta}_z,\tilde{\theta}_y,\tilde{\theta}_x)\mathbf{P}_{SJ} \tag{6.16}$$

The reason for having this joint is to facilitate introduction of constraints, prescribed motions and spring/damper properties on the DOFs represented by the joint variables rather than the global triad DOFs. Note that an entirely spring-based free joint formulation, without the joint variable transformations given by equations (6.15) and (6.16), also is available for this joint type, see Section 5.1.3.

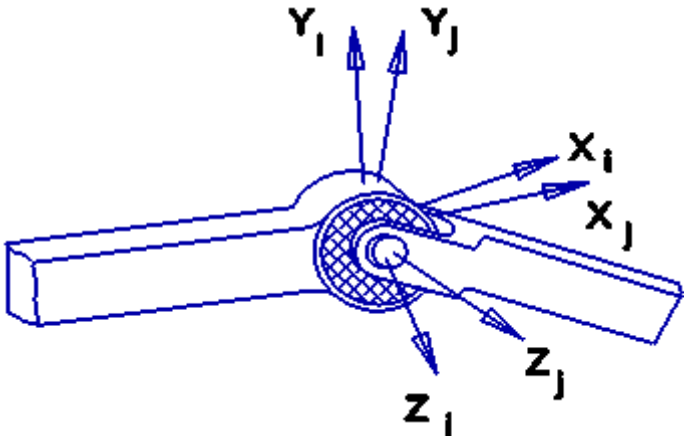


Figure 6.4: Coordinate system i and j for a free joint.

6.2.7 Axial Joint

An axial joint is modeled from two supernodes located on two different links. The joint position matrix $\mathbf{P}_{JT}$ orients the joint coordinate system to have its local x-axis pointing from the master triad to the slave triad. The joint has one joint variable matrix with a single DOF, $\tilde{u}$, the displacement along the local x-axis:

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{MG}\mathbf{P}_{JT}\mathbf{P}_1(\tilde{u})\mathbf{P}_{SJ} \tag{6.17}$$

6.3 Multi-master Joints

The multi-master joints have the same formulation with respect to the relative motion between the joint itself and the slave triad, as single-master joints. These joints have an additional variable, called the slider variable s, which expresses the position of the joint (or follower) along a curve defined by N master triads, i.e, $\mathbf{P}_{JG}(s)$. For each master triad, the joint position relative to the triad itself is defined through $\mathbf{P}_{JT_i}$, such that the path the follower is to travel along does not need to go through the triads themselves. This can be expressed as

$$\mathbf{P}_{JG} \;=\; \mathbf{P}(s, \mathbf{P}_{JG_1}, \cdots, \mathbf{P}_{JG_N}) \quad \text{where} \quad \mathbf{P}_{JG_i} = \mathbf{P}_{MG_i}\mathbf{P}_{JT_i} \tag{6.18}$$

which resembles the similar expression for the single master joint in equation (6.2).

The variation of equation (6.18) with respect to the N master triads is

$$\begin{bmatrix} \delta\mathbf{u}_s \\ \delta\boldsymbol{\omega}_s \end{bmatrix}_M = \sum_{i=1}^{N} f(s)_i \begin{bmatrix} \mathbf{I} & \widehat{\mathbf{e}}_{SM_i} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \delta\mathbf{u}_{m_i} \\ \delta\boldsymbol{\omega}_{m_i} \end{bmatrix} \tag{6.19}$$

which closely resembles the similar expression for the single master joints, equation (6.5). If the follower is positioned between node k and l, the interpolation functions $f(s)_i$ will be

$$\begin{aligned} f(s)_k &= (-s - s_l)/(s_l - s_k) \\ f(s)_l &= (s - s_k)/(s_l - s_k) \\ f(s)_i &= 0 \quad \text{for } i \neq k \text{ and } i \neq l \end{aligned} \tag{6.20}$$

6.3.1 Prismatic Joint

The prismatic joint is modeled from three or more supernodes where one node, the slave triad, is located on the first link while two or more nodes, the master triads, are located on the second link. The master triads define a straight line along which the sliding occur, see Figure 6.5. The slave link can rotate about the x- and y-axes of the joint, which gives the following parameterization:

$$\mathbf{P}_{SG} = \mathbf{P}_{JG}(s, \cdots)\mathbf{P}_2(\tilde{\theta}_y)\mathbf{P}_1(\tilde{\theta}_x)\mathbf{P}_{SJ} \tag{6.21}$$

The z-axis of the joint is always pointing in the positive curve direction.

To model telescopic motion, two parallel prismatic joints of this type is used between the same two links of the mechanism.

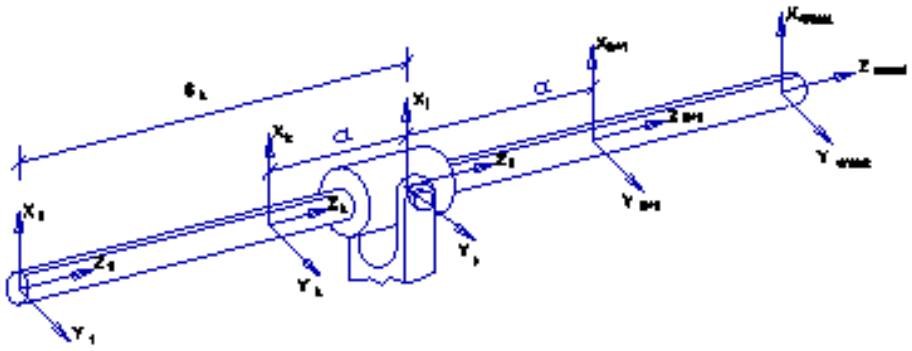


Figure 6.5: Prismatic joint with slave and master nodes.

6.3.2 Cylindric Joint

The cylindric joint closely resembles the prismatic joint, but the slave link is also free to rotate about the master line itself. The joint coordinate system has, as for the prismatic joint, the local z-axis along the master curve in positive direction, which gives the parameterization:

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{JG}(s,\cdots)\mathbf{P}_3(\tilde{\theta}_z)\mathbf{P}_2(\tilde{\theta}_y)\mathbf{P}_1(\tilde{\theta}_x)\mathbf{P}_{SJ} \tag{6.22}$$

The rotation can also be parameterized using rotation axis components:

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{JG}(s,\cdots)\mathbf{P}_1(\tilde{\theta}_z,\tilde{\theta}_y,\tilde{\theta}_x)\mathbf{P}_{SJ} \tag{6.23}$$

6.3.3 Cam Joint

The cam surface is defined by a curve through an ordered set of master triads. If the first and last master are identical, the cam has a closed surface. The curve tangent defines the local z-axis of the joint coordinate system. Its local x-axis corresponds with the surface normal direction, and is defined from the local x-axis direction of the master triads along the curve.

Currently, the curve defining the cam surface can consist of straight lines and circle arcs only, each made up of three nodes, see Figure 6.6. A straight line is simply an arc with zero curvature ($1/R \approx 0$).

Remark: *It is worth noticing that a sudden change in radii of the curve gives a discontinuity in the second derivative, resulting in sudden forces in the cam.*

The default master-slave based cam parameterization has full rotational freedom of the slave link relative to the follower, as well as translational DOFs in the local x- and y-direction of the follower, which gives

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{JG}(s,\cdots)\mathbf{P}_3(\tilde{u},\tilde{v},\tilde{\theta}_z)\mathbf{P}_2(\tilde{\theta}_y)\mathbf{P}_1(\tilde{\theta}_x)\mathbf{P}_{SJ} \tag{6.24}$$

The translational DOFs can be suppressed entirely, and individually, which give the following three alternative formulations for the cam joint:

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{JG}(s,\cdots)\mathbf{P}_3(\tilde{u},\tilde{\theta}_z)\mathbf{P}_2(\tilde{\theta}_y)\mathbf{P}_1(\tilde{\theta}_x)\mathbf{P}_{SJ} \tag{6.25}$$

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{JG}(s,\cdots)\mathbf{P}_3(\tilde{v},\tilde{\theta}_z)\mathbf{P}_2(\tilde{\theta}_y)\mathbf{P}_1(\tilde{\theta}_x)\mathbf{P}_{SJ} \tag{6.26}$$

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{JG}(s,\cdots)\mathbf{P}_3(\tilde{\theta}_z)\mathbf{P}_2(\tilde{\theta}_y)\mathbf{P}_1(\tilde{\theta}_x)\mathbf{P}_{SJ} \tag{6.27}$$

The rotation can also be parameterized using rotation axis components as for the ball-, free- and cylindric joints, i.e., instead of equation (6.24) we have:

$$\mathbf{P}_{SG} \;=\; \mathbf{P}_{JG}(s,\cdots)\mathbf{P}_1(\tilde{u},\tilde{v},\tilde{\theta}_z,\tilde{\theta}_y,\tilde{\theta}_x)\mathbf{P}_{SJ} \tag{6.28}$$

and similarly for equations (6.25)–(6.27).

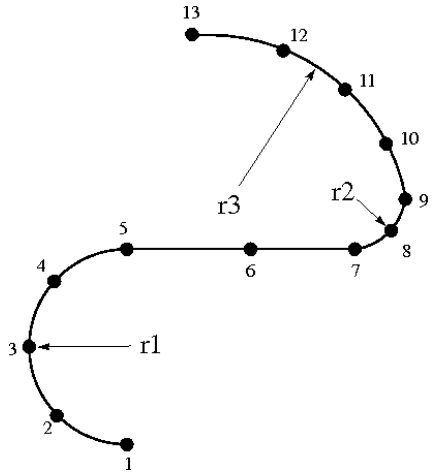


Figure 6.6: Cam curve composed of a circle arc, a straight line, a circle arc and another circle arc. The first and last arcs consist of two curve segments (5 nodes). The straight line and the second arc consist of one curve segment each.

6.3.4 Spring-based cam joint formulation

Contact between the slave triad and the cam surface can be modeled using the formulation described above in Section 6.3.3, by introducing non-linear springs in the two translational joint variables $\tilde{u}$ and $\tilde{v}$ of equation (6.24), or in one of them only by using equation (6.25) or (6.26). However, in situations with large relative separation between the slave triad and the cam curve, or if the cam curve has sharp corners (infinite curvature), the solution might become unstable using this formulation.

A more robust alternative is to use a purely spring-based formulation for such contact problems[1]. This is actually a multi-master equivalent to the *global spring* element described in Section 5.1.3. That is, it does not involve a transformation of the free variables according to equations (6.24)–(6.28). Instead, the stiffness and force contributions from the springs are added directly to the slave and (some of) the master triad DOFs of the cam joint.

A set of (up to) six orthogonal joint springs (three translational and three rotational springs) connect the slave node to the follower location on the cam curve, i.e., the projection of the slave location onto the cam curve along the

[1]This spring-based formulation has actually been the default formulation in Fedem since version R2.5m3.

local x-direction, see Figure 6.7. A spring must always be present in the local x-direction (the surface normal direction), otherwise no contact would be established. In the other five directions, spring-constraining is optional.

Remark: *The remark on rotational stiffnesses in global spring elements given in Section 5.1.3, also applies to the spring-based cam joint. Therefore, when using rotational springs, make sure they are stiff enough to avoid large relative rotations.*

The springs are assumed located at the contact point, C, with rigid arms to the slave node (S) and the master secant point (M), respectively. Let $\mathbf{k}_c = \lceil k_i \rfloor$ and $\mathbf{f}_c = \{f_i\}$, $i = 1 \ldots 6$, denote the diagonal stiffness matrix and the associated force vector of the compound joint spring in its local system. The contributions to the slave node are then found as

$$\mathbf{k}_s \;=\; \mathbf{T}_s \mathbf{k}_c \mathbf{T}_s^T \quad \text{and} \quad \mathbf{f}_s \;=\; \mathbf{T}_s \mathbf{f}_c \quad \text{with} \quad \mathbf{T}_s \;=\; \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \widehat{\mathbf{e}}_s & \mathbf{I} \end{bmatrix} \tag{6.29}$$

where $\mathbf{e}_s = \mathbf{x}_c - \mathbf{x}_s$ is the eccentricity vector from the slave node to the contact point, C. Similarly, the contributions to master node i are found as

$$\mathbf{k}_i \;=\; -f_i(\xi)\mathbf{T}_m \mathbf{k}_c \mathbf{T}_m^T \,,\; \mathbf{f}_i \;=\; -f_i(\xi)\mathbf{T}_m \mathbf{f}_c \quad \text{with} \quad \mathbf{T}_m \;=\; \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \widehat{\mathbf{e}}_m & \mathbf{I} \end{bmatrix} \tag{6.30}$$

where $\mathbf{e}_m = \mathbf{x}_c - \mathbf{x}_m$ is the eccentricity vector from the master secant point (M) to the contact point, and the function $f_i(\xi)$ is a linear interpolation function distributing the contributions to the two closest master nodes, and is given by equation (6.20).

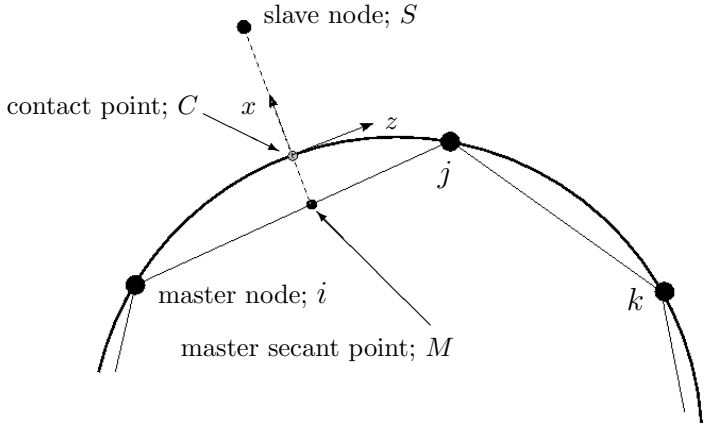


Figure 6.7: The geometry of the spring-based cam joint.

The updated spring lengths in the spring-based cam joint are computed similarly as for the global spring element of Section 5.1.3. However, for the rotational springs we now always assume zero initial angle and deflection, regardless of the modeling configuration, such that the computations become

$$\mathbf{l} = [\, l_x \;\; l_y \;\; 0 \,]^T = \mathbf{R}_c^T(\mathbf{x}_s - \mathbf{x}_c) \tag{6.31}$$

$$\boldsymbol{\alpha} = [\alpha_x \; \alpha_y \; \alpha_z]^T = \text{EulerZYX}(\mathbf{R}_c^T \mathbf{R}_s^T \mathbf{R}_{s0}^T \mathbf{R}_{c0}) \tag{6.32}$$

Here, $\mathbf{x}_s$ and $\mathbf{x}_c$ denote the current global position vectors of the slave node and the contact point, respectively, whereas $\mathbf{R}_s$ and $\mathbf{R}_c$ denote the associated transformation matrices representing their current orientation, and $\mathbf{R}_{s0}$ and $\mathbf{R}_{c0}$ are the corresponding orientations of the initial (modeling) configuration. It also possible to assign spring properties to the local z-direction, and the spring length l_z is then identical to the current curve length position, s.

Radial contact spring

By assigning stiffness functions that have zero stiffness in a certain deflection range around zero, and a high stiffness on both sides outside this range, it is possible to model that the slave should remain inside a cylindric surface along the cam curve. However, for this cylinder to have a circular cross section (like a pipe), the contact spring(s) should be effective in a radial coordinate system (r, θ), rather than the local Cartesian system (x, y).

The polar spring lengths (l_r, l_θ) are obtained from equation (6.31) through

$$l_r = \sqrt{l_x^2 + l_y^2} \quad \text{and} \quad l_\theta = \arctan\left(\frac{l_y}{l_z}\right) \tag{6.33}$$

The corresponding spring stiffnesses and forces are then applied in the radial r-direction, and in the circular θ-direction defined such that the θ-axis is orthogonal to both the r- and z-directions, and (r, θ, z) forms a right-handed system. In addition, we get a geometric stiffness contribution in θ-direction of magnitude F_r/l_r where F_r is the computed spring force in r-direction, due to the changing direction of the radial spring.

Remark: *The radial contact spring must have zero stiffness for small spring lengths or deflections. Otherwise, the geometric stiffness term, F_r/l_r, will go to infinity and cause solution instabilities.*

6.4 Master and Slave based Transmissions

Master and slave based transmissions are expressed using linear dependencies between selected joint variables of the various joint types described above.

6.4.1 Gear Joint

The gear joint is based on two revolute joints, one for the input shaft and one for the output shaft. The master triads for these two joints must be placed on the same link—the gear housing. Setting the gear ratio to N, the linear dependency of the rotational joint DOFs becomes

$$\theta_{z_O} = N\,\theta_{z_I} \tag{6.34}$$

where subscript O designates the output shaft, and I the input shaft.

6.4.2 Rack and Pinion

The rack–and–pinion is based on a revolute joint and a prismatic joint. The master triads of the revolute joint and prismatic joint should be placed on the same link—the joint housing. The linear dependency of the rack–and–pinion becomes

$$s_O = N\,\theta_{z_I} \tag{6.35}$$

when using subscript O for the slider variable of the prismatic joint, and subscript I for the rotational DOF of the input shaft.

6.4.3 Screw Joint

The screw joint is based on a cylindric joint. The slider DOF is now connected to the rotational DOF about the joint z-axis, which gives the linear dependency

$$s_O = N\,\theta_{z_I} \tag{6.36}$$

6.5 Joint Friction

Friction is calculated from forces, moments, and relative velocity in a joint. The friction properties consist of viscous friction, Coulomb dry friction, modified Stribeck friction, and friction force caused by prestressed components. The Stribeck friction yields a continuous description of the friction from static to sliding movement.

Forces and moments in the joint give an equivalent load, which is the basis for computing the acting friction force in the joint. The equivalent load for the various joint types is given in the Sections 6.5.5–6.5.8 below.

6.5.1 Viscous friction

Viscous friction can act between any two supernodes or on any joint variable. The damper force or torque is equal to the damper's viscous coefficient multiplied by the damper velocity.

$$F_{viscous} \;=\; c\,V \tag{6.37}$$

where c is the damper coefficient.

6.5.2 Coulomb friction

In classical Coulomb friction models, there is a constant friction force opposing the motion when the velocity is non-zero. In the case of zero velocity, the friction opposes all motions as long as the force is smaller in magnitude than the friction force (see Figure 6.8)

$$F_{Coulomb} \;=\; \mu_{Coulomb}\,F_e\,\mathrm{sgn}(V) \tag{6.38}$$

where

$F_{Coulomb}$: Coulomb friction force
$\mu_{Coulomb}$: Coefficient of friction
F_e : Equivalent normal load
V : Velocity
$\mathrm{sgn}(V)$: The direction of movement (± 1)

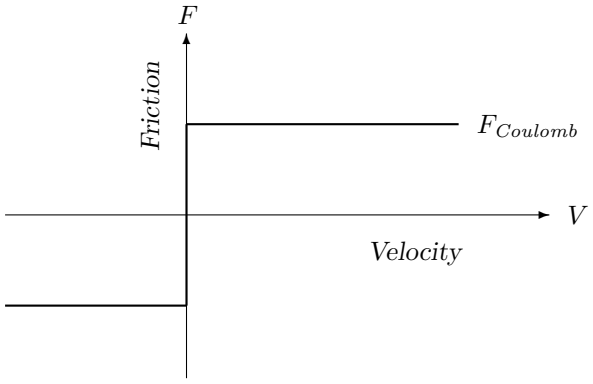


Figure 6.8: Coulomb friction

Reduction gears and bearing set-ups are often prestressed to avoid backlash. This prestress produces friction even when the external load is zero. This friction component, which is added to the Coulomb friction term. is defined as (see Figure 6.9):

$$F_{prestress} = F_0 \, \mathrm{sgn}(V) \tag{6.39}$$

where F_0 is the friction force or torque caused by prestressed components or other constant friction effects.

6.5.3 Modified Stribeck friction

Modified Stribeck friction defines friction as a constant at extremely low velocities, then making a smooth transition from higher static friction to the lower kinetic, or kinetic plus viscous, friction. A model using modified Stribeck friction predicts a steady motion at extremely low velocities, instability through a range of velocities, and stable motion above a certain threshold velocity.

Experiments have verified that after the stiction force has been surmounted, the friction decreases exponentially reaching approximately 60% of the breakaway force. These bends occur at velocities close to zero. Detailed experiments carried out in industrial manipulators and reduction gears have confirmed this negative velocity dependence at low velocity. The hysteresis effect is also added into the Stribeck friction. It has been observed that the friction curve is not single-valued. There is a static friction force that is higher than the kinetic friction, and the friction does not return to the higher static

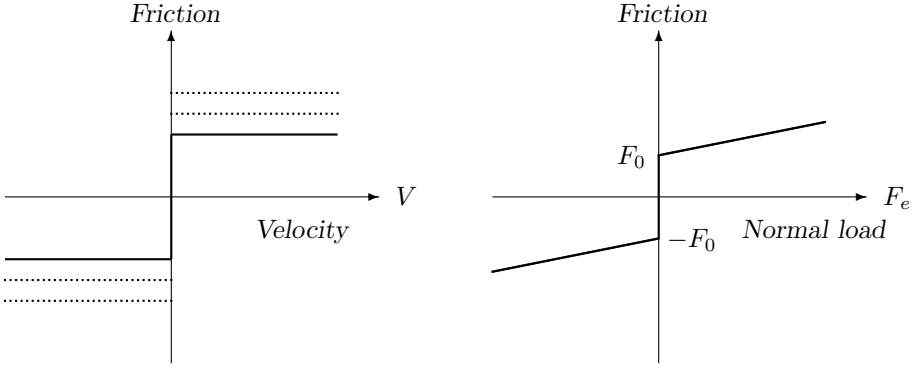


Figure 6.9: Friction caused by prestressed components

value when the sliding velocity decreases. This means that the friction remains at the low Coulomb friction value until the velocity has changed sign or is zero (see Figure 6.10):

$$F_{Stribeck} \;=\; F_{Coulomb}(1 + Se^{-(\frac{V}{V_{slip}})^2})\,\mathrm{sgn}(V) \tag{6.40}$$

where

$F_{Stribeck}$: Stick-slip friction as a function of velocity

$F_{Coulomb}$: Coulomb friction including friction from prestressed components

S : Magnitude of Stribeck effect (stick-slip factor):

$$S = \frac{F_{static} - F_{Coulomb}}{F_{Coulomb}}$$

V_{slip} : Critical velocity for Stribeck effect

$\mathrm{sgn}(V)$: The direction of movement (± 1)

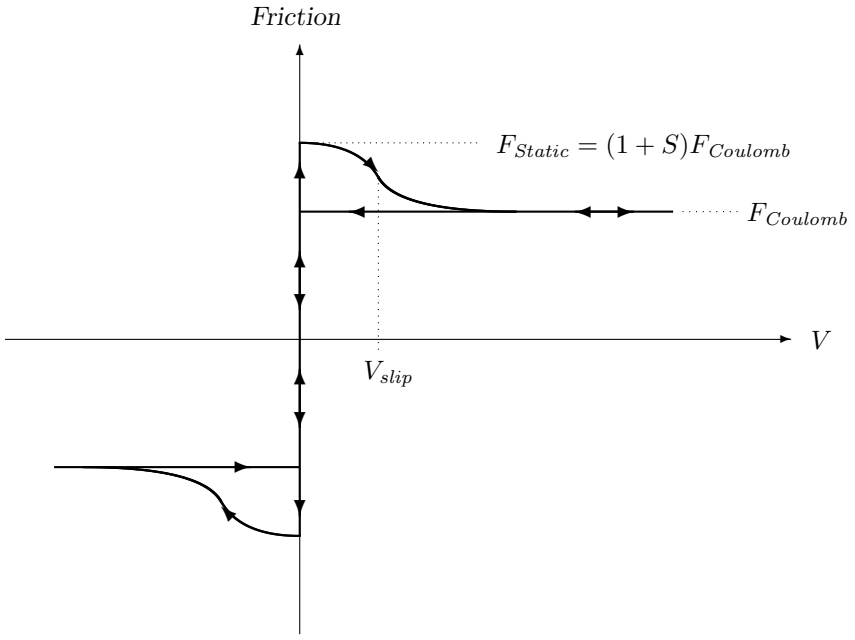


Figure 6.10: Modified Stribeck friction including hysteresis

6.5.4 Total friction

The total friction model is a function resulting from the combination of the three components described above:

$$F_{total} = F_{viscous} + \left[F_0 + \mu_{Coulomb}\, F_e \left(1 + Se^{-\left(\frac{V}{V_{slip}}\right)^2}\right)\right] \text{sgn}(V) \qquad (6.41)$$

6.5.5 Equivalent load in revolute joint

Friction in revolute joints depends on bearing design and joint loads. The revolute joint transmits the forces F_x, F_y, and F_z and the bending moments M_x and M_y. All these forces and moments are carried by the joint bearings. The first step is to calculate the axial and radial bearing load based on these joint forces and moments.

The revolute joint can be designed and modeled in many ways. In some cases the joint has only one bearing. The bending capacity is then calculated by the bearing constant a. This is a standard constant given in bearing catalogs. If the joint has two bearings as shown in Figure 6.11, the joint can also be modeled by using two separate revolute joints, one at each bearing. The next step is to calculate the equivalent bearing load as a function of axial and radial bearing load and bearing geometry. Different types of bearings are

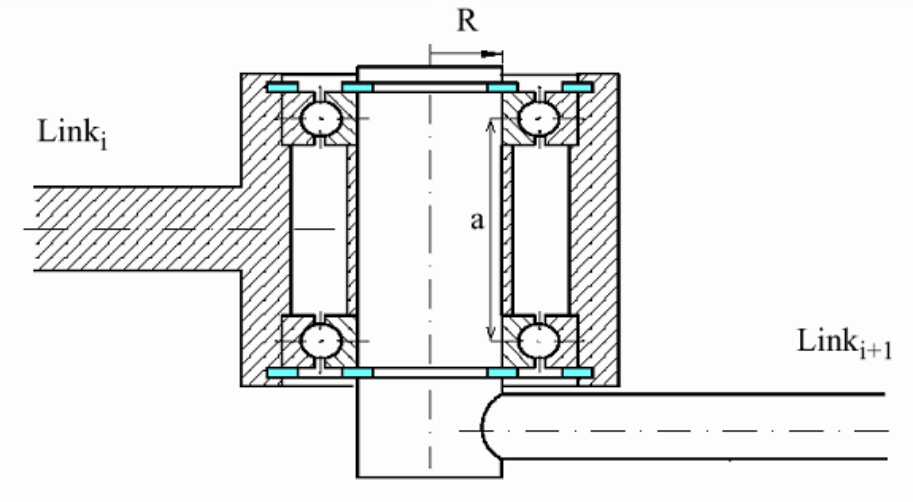


Figure 6.11: Revolute joint

designed to carry either radial or axial load or a combination of these loads. The equivalent load is then calculated as follows:

$$\begin{aligned} \text{Bearing 1:} \quad F_{rx1} &= \frac{F_x}{2} - \frac{M_y}{a} \\ F_{ry1} &= \frac{F_y}{2} - \frac{M_x}{a} \\ F_{r1} &= \sqrt{F_{rx1}^2 + F_{ry1}^2} \end{aligned} \tag{6.42}$$

$$\begin{aligned} \text{Bearing 2:} \quad F_{rx2} &= \frac{F_x}{2} + \frac{M_y}{a} \\ F_{ry2} &= \frac{F_y}{2} + \frac{M_x}{a} \\ F_{r2} &= \sqrt{F_{rx2}^2 + F_{ry2}^2} \end{aligned} \tag{6.43}$$

$$\text{Radial bearing load:} \quad F_r = F_{r1} + F_{r2} \tag{6.44}$$

$$\text{Axial bearing load:} \quad F_a = F_z \tag{6.45}$$

$$\text{Equivalent bearing load:} \quad F_e = F_r + F_a Y \tag{6.46}$$

where Y is a constant depending on bearing type and geometry, and is given in the bearing catalogs. The friction force is calculated as a function of the equivalent bearing load and the angular velocity. The friction torque in the joint is calculated by multiplying the friction force by the bearing radius R.

A simplified friction representation in revolute joints, where the effects of the bending moments and the axial load are ignored can also be used. The coefficients a and Y are then not needed and the equivalent normal load given by equation (6.46), reduces to

$$F_e = \sqrt{F_x^2 + F_y^2} \tag{6.47}$$

6.5.6 Equivalent load in ball and free joints

Ball and free joints can both be assigned friction properties to one of its three rotational DOFs. The equivalent normal load is then given by equation (6.47), where the indices x and y now represent the two local joint directions that are orthogonal to the local axis of the chosen friction DOF. The actual friction torque is then calculated in the same way as for a revolute joint, based on a specified ball radius R and the angular velocity in the chosen friction DOF.

For free joints, friction can also be assigned to a translational DOF instead of a rotational DOF. The equivalent normal load is then

$$F_e = \begin{cases} F_{e1} = \sqrt{F_y^2 + F_z^2} & \text{for friction in the Tx DOF} \\ F_{e2} = \sqrt{F_z^2 + F_x^2} & \text{for friction in the Ty DOF} \\ F_{e3} = \sqrt{F_x^2 + F_y^2} & \text{for friction in the Tz DOF} \end{cases} \quad (6.48)$$

where F_x, F_y and F_z are the joint forces in its local directions. The actual friction force is then calculated as a function of F_e and the linear velocity in the chosen friction direction.

6.5.7 Equivalent load in prismatic joint

The friction in a prismatic joint depends on forces normal to the sliding axis (z-axis) and forces caused by torque around the z-axis. The joint rotation around the z-axis is fixed. This is usually accomplished by means of a sliding key or similar mechanism (see Figure 6.12).

The friction coefficient in the locking device can in some cases be much higher than in the linear guidance. This linear bearing is often made of low-friction rolling elements. Forces in the locking device and normal load in the guidance are therefore separated and can be weighted by the factor Y.

Normal Force: $$F_N = \sqrt{F_y^2 + (F_x - \frac{M_z}{R})^2} \quad (6.49)$$

Force in locking device: $$F_M = \frac{M_z}{R} \quad (6.50)$$

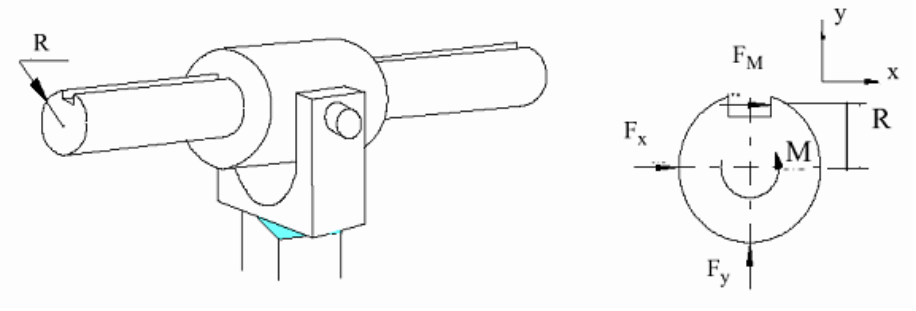


Figure 6.12: Prismatic joint

Here (see Figure 6.12), F_y and F_x are joint forces in the y and z direction, M_z is the torque around the sliding z-axis, and R is the radius or distance to the locking device. The equivalent load is then given by

$$F_e \;=\; F_N + F_M Y \tag{6.51}$$

A simplified friction representation in prismatic joints, where the effects of the locking device is ignored can also be used. The coefficients R and Y are then not needed and the equivalent normal load, equation (6.51), reduces to F_{e3} of equation (6.48).

The actual friction force in the prismatic joint is calculated as a function of the equivalent normal load and the linear velocity of the slider DOF.

6.5.8 Equivalent load in cam joint

The equivalent load in a cam joint is the force between the cam and the follower in the normal (x) direction. No effect from any translation in the y-direction is accounted for. The equivalent force F_e is thus equal to the force of the contact spring in the x-direction, and the actual friction force is calculated similarly as for the prismatic joint.

Chapter 7 Dynamics Simulation

7.1 Dynamics equation on incremental form

The equation of dynamic equilibrium at time t may be written

$$\mathbf{R}\left(t, \mathbf{r}, \dot{\mathbf{r}}, \ddot{\mathbf{r}}\right) \;=\; \mathbf{0} \tag{7.1}$$

or

$$\mathbf{F}^{\mathrm{I}}\left(t, \mathbf{r}, \dot{\mathbf{r}}, \ddot{\mathbf{r}}\right) \;+\; \mathbf{F}^{\mathrm{D}}\left(t, \mathbf{r}, \dot{\mathbf{r}}, \ddot{\mathbf{r}}\right) \;+\; \mathbf{F}^{\mathrm{S}}\left(t, \mathbf{r}, \dot{\mathbf{r}}, \ddot{\mathbf{r}}\right) \;-\; \mathbf{Q}\left(t, \mathbf{r}, \dot{\mathbf{r}}, \ddot{\mathbf{r}}\right) \;=\; \mathbf{0} \tag{7.2}$$

where

- $\mathbf{F}^{\mathrm{I}}$ represents inertia forces,
- $\mathbf{F}^{\mathrm{D}}$ represents damping forces
- $\mathbf{F}^{\mathrm{S}}$ represents internal elastic forces
- $\mathbf{Q}$ represents input loads (forces and torques) and gravitational forces

Equation (7.2) is integrated in time with a time increment length of h. At time t_k, this equation of equilibrium may be written

$$\mathbf{F}^{\mathrm{I}}_k + \mathbf{F}^{\mathrm{D}}_k + \mathbf{F}^{\mathrm{S}}_k \;=\; \mathbf{Q}_k \tag{7.3}$$

Since equation (7.3) has to be satisfied at all times, we can subtract this equation from the same equation at time $t_{k+1} = t_k + h$ to produce the equation of motion on incremental form, as follows

$$\left[\mathbf{F}^{\mathrm{I}}_{k+1} - \mathbf{F}^{\mathrm{I}}_k\right] \;+\; \left[\mathbf{F}^{\mathrm{D}}_{k+1} - \mathbf{F}^{\mathrm{D}}_k\right] \;+\; \left[\mathbf{F}^{\mathrm{S}}_{k+1} - \mathbf{F}^{\mathrm{S}}_k\right] \;=\; \mathbf{Q}_{k+1} - \mathbf{Q}_k \tag{7.4}$$

or

$$\Delta\mathbf{F}^{\mathrm{I}}_k + \Delta\mathbf{F}^{\mathrm{D}}_k + \Delta\mathbf{F}^{\mathrm{S}}_k \;=\; \Delta\mathbf{Q}_k \tag{7.5}$$

In the following, the different terms of equation (7.5) are expanded.

The inertia, damping and stiffness relationships are estimated by a linear approximation around the starting position for each time increment. Incremental system matrices are then generated for that configuration. Equilibrium iterations are next used to reduce the error in this approximation. The exact incremental system matrices, called tangent matrices, are functions of the unknown displacement increments and cannot be generated in advance.

The incremental inertia forces from equation (7.5) may be written

$$\Delta \mathbf{F}_k^{\mathrm{I}} \;=\; \mathbf{F}_{k+1}^{\mathrm{I}} - \mathbf{F}_k^{\mathrm{I}} \;=\; \mathbf{M}_k \Delta \ddot{\mathbf{r}}_k \tag{7.6}$$

where $\mathbf{M}_k$ is the system mass matrix at the beginning of time increment k, and $\Delta \ddot{\mathbf{r}}_k = \ddot{\mathbf{r}}_{k+1} - \ddot{\mathbf{r}}_k$ represents the change in acceleration during that time increment. The system mass matrix may be constant or a function of the displacement $\mathbf{r}$, depending on the element mass representation used. The element mass matrices are constant, but undergo a geometric transformation before they are added into the system matrix. If a lumped mass representation is chosen, the element mass matrix is diagonal and the geometric transformations have no effect. Then the system mass matrix will be diagonal and constant during integration.

The incremental damping forces from equation (7.5) may be written

$$\Delta \mathbf{F}_k^{\mathrm{D}} \;=\; \mathbf{F}_{k+1}^{\mathrm{D}} - \mathbf{F}_k^{\mathrm{D}} \;=\; \mathbf{C}_k \Delta \dot{\mathbf{r}}_k \tag{7.7}$$

where $\mathbf{C}_k$ is the system damping matrix at the beginning of time increment k, and $\Delta \dot{\mathbf{r}}_k = \dot{\mathbf{r}}_{k+1} - \dot{\mathbf{r}}_k$ represents the change in velocity during that time increment. The system damping matrix may be constant or a function of the displacement $\mathbf{r}$. If the damping matrix is diagonal, the damping matrix will not be affected by the geometric transformation and remains constant.

The incremental elastic forces from equation (7.5) may be written

$$\Delta \mathbf{F}_k^{\mathrm{S}} \;=\; \mathbf{F}_{k+1}^{\mathrm{S}} - \mathbf{F}_k^{\mathrm{S}} \;=\; \mathbf{K}_k \Delta \mathbf{r}_k \tag{7.8}$$

where $\mathbf{K}_k$ is the system stiffness matrix at the beginning of time increment k, and $\Delta \mathbf{r} = \mathbf{r}_{k+1} - \mathbf{r}_k$ represents the associated displacement increment. The system stiffness matrix is in general a function of the displacement vector.

The incremental dynamic equation of motion (7.5) can now be written on the linearized form

$$\mathbf{M}_k \Delta \ddot{\mathbf{r}}_k + \mathbf{C}_k \Delta \dot{\mathbf{r}}_k + \mathbf{K}_k \Delta \mathbf{r}_k \;=\; \Delta \mathbf{Q}_k \tag{7.9}$$

The matrices $\mathbf{M}_k$, $\mathbf{C}_k$ and $\mathbf{K}_k$ may be recalculated for each time increment and iteration of the solution process. Solving equation (7.9) using a time integration algorithm, such as the Newmark method, gives $\Delta \mathbf{r}_k, \Delta \dot{\mathbf{r}}_k$, and $\Delta \ddot{\mathbf{r}}_k$. Therefore, the total solution at the end of the increment is

$$\begin{aligned} \mathbf{r}_{k+1} &= \mathbf{r}_k + \Delta \mathbf{r}_k \\ \dot{\mathbf{r}}_{k+1} &= \dot{\mathbf{r}}_k + \Delta \dot{\mathbf{r}}_k \\ \ddot{\mathbf{r}}_{k+1} &= \ddot{\mathbf{r}}_k + \Delta \ddot{\mathbf{r}}_k \end{aligned} \tag{7.10}$$

This solution may be used to calculate the forces $\mathbf{F}^{\mathrm{I}}_{k+1}$, $\mathbf{F}^{\mathrm{D}}_{k+1}$ and $\mathbf{F}^{\mathrm{S}}_{k+1}$. However, due to the linear approximation there will be unbalanced forces at the end of the increment, given by

$$\hat{\mathbf{R}}_{k+1} \;=\; \mathbf{Q}_{k+1} - \left[\mathbf{F}^{\mathrm{I}}_{k+1} + \mathbf{F}^{\mathrm{D}}_{k+1} + \mathbf{F}^{\mathrm{S}}_{k+1}\right] \tag{7.11}$$

These residual forces may be added to the load increment for the next step

$$\Delta\hat{\mathbf{Q}}_k \;=\; \Delta\mathbf{Q}_k + \hat{\mathbf{R}}_k \;=\; \mathbf{Q}_{k+1} - \left[\mathbf{F}^{\mathrm{I}}_{k} + \mathbf{F}^{\mathrm{D}}_{k} + \mathbf{F}^{\mathrm{S}}_{k}\right] \tag{7.12}$$

Replacing $\Delta\mathbf{Q}_k$ by $\Delta\hat{\mathbf{Q}}_k$, equation (7.9) may then be written

$$\mathbf{M}_k\Delta\ddot{\mathbf{r}}_k + \mathbf{C}_k\Delta\dot{\mathbf{r}}_k + \mathbf{K}_k\Delta\mathbf{r}_k \;=\; \mathbf{Q}_{k+1} - \left[\mathbf{F}^{\mathrm{I}}_{k} + \mathbf{F}^{\mathrm{D}}_{k} + \mathbf{F}^{\mathrm{S}}_{k}\right] \tag{7.13}$$

7.2 Newmark time integration

The Newmark β-family of algorithms is used for time integration in Fedem. The basis for the Newmark method is the following update scheme

$$\mathbf{r}_{k+1} \;=\; \mathbf{r}_k + h\dot{\mathbf{r}}_k + \left(\frac{1}{2} - \beta\right) h^2\ddot{\mathbf{r}}_k + \beta h^2\ddot{\mathbf{r}}_{k+1} \tag{7.14}$$

$$\dot{\mathbf{r}}_{k+1} \;=\; \dot{\mathbf{r}}_k + (1-\gamma)h\ddot{\mathbf{r}}_k + \gamma h\ddot{\mathbf{r}}_{k+1} \tag{7.15}$$

where β and γ are integration parameters (see Section 7.2.1) and h is the time increment length. These equations may be rewritten into the incremental forms

$$\mathbf{r}_{k+1} \;=\; \mathbf{r}_k + \Delta\mathbf{r}_k \quad \text{where} \quad \Delta\mathbf{r}_k \;=\; h\dot{\mathbf{r}}_k + \frac{h^2}{2}\ddot{\mathbf{r}}_k + \beta h^2\Delta\ddot{\mathbf{r}}_k \tag{7.16}$$

$$\dot{\mathbf{r}}_{k+1} \;=\; \dot{\mathbf{r}}_k + \Delta\dot{\mathbf{r}}_k \quad \text{where} \quad \Delta\dot{\mathbf{r}}_k \;=\; h\ddot{\mathbf{r}}_k + \gamma h\Delta\ddot{\mathbf{r}}_k \tag{7.17}$$

The increment in velocity and acceleration can now be expressed as functions of the displacement increment and known quantities at time t_k. With $\Delta\ddot{\mathbf{r}}_k = \ddot{\mathbf{r}}_{k+1} - \ddot{\mathbf{r}}_k$, equation (7.16) produces

$$\begin{aligned}\ddot{\mathbf{r}}_{k+1} \;=\; \ddot{\mathbf{r}}_k + \Delta\ddot{\mathbf{r}}_k \;&=\; \ddot{\mathbf{r}}_k + \frac{1}{\beta h^2}\Delta\mathbf{r}_k - \frac{1}{\beta h}\dot{\mathbf{r}}_k - \frac{1}{2\beta}\ddot{\mathbf{r}}_k \\ &=\; \frac{1}{\beta h^2}\Delta\mathbf{r}_k - \mathbf{a}_k\end{aligned} \tag{7.18}$$

where

$$\mathbf{a}_k \;=\; \frac{1}{\beta h}\dot{\mathbf{r}}_k + \left(\frac{1}{2\beta} - 1\right)\ddot{\mathbf{r}}_k \tag{7.19}$$

Combining equation (7.17) with equation (7.18) yields

$$\begin{aligned}\dot{\mathbf{r}}_{k+1} &= \dot{\mathbf{r}}_k + \Delta\dot{\mathbf{r}}_k &= \dot{\mathbf{r}}_k + \frac{\gamma}{\beta h}\Delta\mathbf{r}_k - \frac{\gamma}{\beta}\dot{\mathbf{r}}_k - \left(\frac{\gamma h}{2\beta} - h\right)\ddot{\mathbf{r}}_k \\ &= \frac{\gamma}{\beta h}\Delta\mathbf{r}_k - \mathbf{d}_k\end{aligned} \tag{7.20}$$

where

$$\mathbf{d}_k = \left(\frac{\gamma}{\beta} - 1\right)\dot{\mathbf{r}}_k + \left(\frac{\gamma}{2\beta} - 1\right)h\ddot{\mathbf{r}}_k \tag{7.21}$$

By inserting equations (7.18) and (7.20) into equation (7.13), we now get

$$\mathbf{N}_k\Delta\mathbf{r}_k = \hat{\mathbf{R}}_k \tag{7.22}$$

where the Newton matrix is

$$\mathbf{N}_k = \mathbf{K}_k + \frac{\gamma}{\beta h}\mathbf{C}_k + \frac{1}{\beta h^2}\mathbf{M}_k \tag{7.23}$$

and the right-hand-side vector is

$$\hat{\mathbf{R}}_k = \mathbf{Q}_{k+1} - \left[\mathbf{F}_k^{\mathrm{I}} + \mathbf{F}_k^{\mathrm{D}} + \mathbf{F}_k^{\mathrm{S}}\right] + \mathbf{M}_k\left[\mathbf{a}_k + \ddot{\mathbf{r}}_k\right] + \mathbf{C}_k\left[\mathbf{d}_k + \dot{\mathbf{r}}_k\right] \tag{7.24}$$

The evaluation of the system Newton matrix, $\mathbf{N}_k$, is covered by Section 7.6, whereas the external loading, $\mathbf{Q}_{k+1}$, and the internal inertia, damping, and elastic forces, $\mathbf{F}_k^{\mathrm{I}}$, $\mathbf{F}_k^{\mathrm{D}}$ and $\mathbf{F}_k^{\mathrm{S}}$, are covered by Section 7.7.

7.2.1 Stability and accuracy

The integration parameters γ and β are selected for controlling the stability, accuracy and efficiency of the integration process. For the linear case, the method is unconditionally stable for

$$\gamma \geq \frac{1}{2} \qquad \text{and} \qquad \beta \geq \frac{1}{4}\left(\gamma + \frac{1}{2}\right)^2 \tag{7.25}$$

For smaller β-values, the method is only conditionally stable. The stability criterion is

$$h_{\mathrm{cr}} = \frac{T}{2\pi}\left(\frac{1}{4}\left(\gamma + \frac{1}{2}\right)^2 - \beta\right)^{-\frac{1}{2}} \tag{7.26}$$

where h_{cr} is the critical time increment size and T is the period for the highest frequency in the model.

The parameter γ may be selected to introduce artificial damping into the integration process. $\gamma > \frac{1}{2}$ gives positive artificial damping; in other words, the amplitude will decay with increasing k. $\gamma < \frac{1}{2}$ gives negative artificial damping, i.e., the amplitude will increase with increasing k. $\gamma = \frac{1}{2}$ gives no artificial damping. Unfortunately, numerical (algorithmic) damping can not be introduced without loss off accuracy, and using a value of $\gamma \neq \frac{1}{2}$ will make the algorithm only first order accurate. Consequently, $\gamma = \frac{1}{2}$ is often the selection. In this case, the Newmark β-family includes the following methods

- $\beta = 0$ The second central difference method with $h_{\text{cr}} = 0.318T$
- $\beta = \frac{1}{12}$ Fox-Goodwins method with $h_{\text{cr}} = 0.389T$
- $\beta = \frac{1}{6}$ Linear acceleration with $h_{\text{cr}} = 0.551T$
- $\beta = \frac{1}{4}$ Constant average acceleration (trapezoid method), which is unconditionally stable for linear systems

Fedem is using $\gamma = \frac{1}{2}$ and $\beta = \frac{1}{4}$ for its Newmark integration algorithm. This constitutes a second order accurate integration with no numerical damping for any frequencies. Because of the lack of numerical damping, we must include structural damping to the model to obtain stable integration with the Newmark algorithm. We will usually include (mainly) stiffness proportional damping in order to introduce dissipation of high-frequency modes, see Section 7.5.

7.3 Newton–Raphson iteration

Equation (7.13) is an approximation of the equilibrium equation at time t_{k+1}. To achieve equilibrium at the end of the increment, so-called Newton–Raphson iterations are used to minimize the error from the solution of this equation.

An iteration is formally conducted by replacing the right hand side of equation (7.13) by the residual from the previous iteration, ${}^{i-1}\hat{\mathbf{R}}_{k+1}$, and then solving for the correction $\delta\mathbf{r}_k$ to $\Delta\mathbf{r}_k$ from

$$\begin{aligned} {}^{i}\mathbf{M}_k\,{}^{i}\delta\ddot{\mathbf{r}}_k + {}^{i}\mathbf{C}_k\,{}^{i}\delta\dot{\mathbf{r}}_k + {}^{i}\mathbf{K}_k\,{}^{i}\delta\mathbf{r}_k &= {}^{i-1}\mathbf{Q}_{k+1} \\ &\quad - \left[{}^{i-1}\mathbf{F}^{\text{I}}_{k+1} + {}^{i-1}\mathbf{F}^{\text{D}}_{k+1} + {}^{i-1}\mathbf{F}^{\text{S}}_{k+1}\right] \end{aligned} \tag{7.27}$$

The superindex to the left of the symbols indicates the iteration number. The displacement, velocity and acceleration increments are then corrected from

$$\begin{aligned} {}^{i}\Delta\mathbf{r}_k &= {}^{i-1}\Delta\mathbf{r}_k + {}^{i}\delta\mathbf{r}_k \\ {}^{i}\Delta\dot{\mathbf{r}}_k &= {}^{i-1}\Delta\dot{\mathbf{r}}_k + {}^{i}\delta\dot{\mathbf{r}}_k \\ {}^{i}\Delta\ddot{\mathbf{r}}_k &= {}^{i-1}\Delta\ddot{\mathbf{r}}_k + {}^{i}\delta\ddot{\mathbf{r}}_k \end{aligned} \tag{7.28}$$

The total displacement, velocity and acceleration vectors are updated through

$$\begin{aligned} {}^{i}\mathbf{r}_{k+1} &= {}^{i-1}\mathbf{r}_{k+1} + {}^{i}\delta\mathbf{r}_{k} \\ {}^{i}\dot{\mathbf{r}}_{k+1} &= {}^{i-1}\dot{\mathbf{r}}_{k+1} + {}^{i}\delta\dot{\mathbf{r}}_{k} \\ {}^{i}\ddot{\mathbf{r}}_{k+1} &= {}^{i-1}\ddot{\mathbf{r}}_{k+1} + {}^{i}\delta\ddot{\mathbf{r}}_{k} \end{aligned} \tag{7.29}$$

The improved displacement increment, given by equation $(7.28)_1$, is next inserted into equations (7.18) and (7.20), respectively, yielding

$${}^{i}\ddot{\mathbf{r}}_{k+1} = \frac{1}{\beta h^2}{}^{i}\Delta\mathbf{r}_{k} - \mathbf{a}_{k} = \frac{1}{\beta h^2}\left[{}^{i-1}\Delta\mathbf{r}_{k} + {}^{i}\delta\mathbf{r}_{k}\right] - \mathbf{a}_{k} \tag{7.30}$$

$${}^{i}\dot{\mathbf{r}}_{k+1} = \frac{\gamma}{\beta h}{}^{i}\Delta\mathbf{r}_{k} - \mathbf{d}_{k} = \frac{\gamma}{\beta h}\left[{}^{i-1}\Delta\mathbf{r}_{k} + {}^{i}\delta\mathbf{r}_{k}\right] - \mathbf{d}_{k} \tag{7.31}$$

From this it follows that the iterative acceleration and velocity corrections are given by, respectively

$${}^{i}\delta\ddot{\mathbf{r}}_{k} = \frac{1}{\beta h^2}{}^{i}\delta\mathbf{r}_{k} \qquad \text{and} \qquad {}^{i}\delta\dot{\mathbf{r}}_{k} = \frac{\gamma}{\beta h}{}^{i}\delta\mathbf{r}_{k} \tag{7.32}$$

Combining equation (7.32) with equation (7.27) yields

$${}^{i}\mathbf{N}_{k}{}^{i}\delta\mathbf{r}_{k} = {}^{i-1}\hat{\mathbf{R}}_{k} \tag{7.33}$$

with

$${}^{i}\mathbf{N}_{k} = {}^{i}\mathbf{K}_{k} + \frac{\gamma}{\beta h}{}^{i}\mathbf{C}_{k} + \frac{1}{\beta h^2}{}^{i}\mathbf{M}_{k} \tag{7.34}$$

and

$${}^{i-1}\hat{\mathbf{R}}_{k} = {}^{i-1}\mathbf{Q}_{k+1} - \left[{}^{i-1}\mathbf{F}^{\mathrm{I}}_{k+1} + {}^{i-1}\mathbf{F}^{\mathrm{D}}_{k+1} + {}^{i-1}\mathbf{F}^{\mathrm{S}}_{k+1}\right] \tag{7.35}$$

After updating the solution state through the equations (7.28) and (7.29), the internal forces ${}^{i}\mathbf{F}^{\mathrm{I}}_{k+1}$, ${}^{i}\mathbf{F}^{\mathrm{D}}_{k+1}$ and ${}^{i}\mathbf{F}^{\mathrm{S}}_{k+1}$ can be calculated from the equations (7.6)–(7.8) and substituted into equation (7.27) to solve for the correction ${}^{i+1}\delta\mathbf{r}_{k}$ of the next iteration, and so on.

If the matrices $\mathbf{M}_k$, $\mathbf{C}_k$ and $\mathbf{K}_k$ are updated in each iteration, this iteration process is called Newton–Raphson iteration. If they all are left constant during the iterations, or only updated after some iterations, the process is called modified Newton–Raphson iteration. The coefficient matrix of the equation system (7.33) is then unchanged for those iterations and new triangularizations are avoided. The iterations are repeated until the chosen convergence test is satisfied (refer to Section 7.3.1).

7.3.1 Convergence criteria

Some criteria for terminating the equilibrium iterations governed by equations (7.27)–(7.29), must be established. They are typically based on some *norms* of the vector quantities involved. When computing the norms, we have the problem of dealing with different units. There are rotational and translational DOFs (typically measured in radians and meters), as well as generalized DOFs associated with the component modes.

Changing the modeling units of length and translation from [m] to [mm] changes the size of the translations with 3 orders of magnitude, whereas the rotations remain the same. Such a change of units should not affect the relative contribution of rotation and translations to the norm of a displacement vector.

Scaled vector norms

In order to make the vector norms unit independent, we choose to use rotations as the "base" displacement quantity since they are always measured in radians. With a rigid rotation of unit magnitude as guiding displacement, an appropriate scaling of the translations is then assumed to be $\frac{1}{L_{\text{model}}}$, where L_{model} is the largest dimension of the model itself.

Based on this, we define the scaled vector norm as

$$\|\mathbf{v}\|_{\text{scaled}} = \sqrt{\frac{\sum_i (w_i v_i)^2}{\sum_i w_i^2}} \tag{7.36}$$

$$\text{where} \quad \begin{cases} w_i = 1 & \text{if } i \text{ is rotational DOF} \\ w_i = \frac{1}{L_{\text{model}}} & \text{if } i \text{ is translational DOF} \\ w_i = 1 & \text{if } i \text{ is generalized DOF} \end{cases}$$

Convergence criterias are now defined as

$$\|\mathbf{v}\|_{\text{scaled}} \leq \varepsilon_{tol} \tag{7.37}$$

where the vector $\mathbf{v}$ can be one of the following

${}^i\delta\mathbf{r}_k$ displacement correction defined by equations (7.27) and (7.33)

${}^i\delta\dot{\mathbf{r}}_k$ velocity correction defined by equations (7.27) and (7.32)[1]

${}^i\delta\ddot{\mathbf{r}}_k$ acceleration correction defined by equations (7.27) and (7.32)

${}^i\hat{\mathbf{R}}_k$ residual vector (unbalanced forces) defined by equations (7.27) and (7.35)

The scaling factors are inverted for the force residual vector.

Remark: *Velocity and acceleration corrections are computed from the displacement correction as time step dependent scalings proportional to* $\frac{1}{h}$ *and* $\frac{1}{h^2}$ *respectively, as given by equation* (7.32)*. For small time steps, any noise in the displacement corrections gets greatly magnified in the velocity and acceleration terms, and can lead to convergence problems if the convergence criterion involves testing on the velocity and acceleration corrections.*

DOF type selective infinite norms

The infinite norm of a vector is defined as

$$\|\mathbf{v}\|_\infty \;=\; \lim_{n\to\infty} \sqrt[n]{\sum_i |v_i^n|} \;=\; \max_{\forall i} |v_i| \tag{7.38}$$

Because of the different DOF types (rotation, translation and component modes), we define three different infinite norms of a mixed vector $\mathbf{v}$ as

$$\begin{array}{lcll} \|\mathbf{v}\|_{\infty,\text{tran}} & = & \max|v_i| & : \forall\, i \text{ being a translational DOF} \\ \|\mathbf{v}\|_{\infty,\text{rot}} & = & \max|v_i| & : \forall\, i \text{ being a rotational DOF} \\ \|\mathbf{v}\|_{\infty,\text{gen}} & = & \max|v_i| & : \forall\, i \text{ being a component mode DOF} \end{array} \tag{7.39}$$

Convergence criterias based on these norms are now defined as

$$\|\mathbf{v}\|_{\infty,\text{tran}} \le \varepsilon_{tol}, \quad \|\mathbf{v}\|_{\infty,\text{rot}} \le \varepsilon_{tol}, \quad \|\mathbf{v}\|_{\infty,\text{gen}} \le \varepsilon_{tol} \tag{7.40}$$

where $\mathbf{v}$ can be the correction to the displacements ${}^i\delta\mathbf{r}_k$, velocity ${}^i\delta\dot{\mathbf{r}}_k$, acceleration ${}^i\delta\ddot{\mathbf{r}}_k$, as well as the residual vector ${}^i\hat{\mathbf{R}}_k$.

[1] For the scaled vector norm of the velocity correction we also have the option of "relaxing" the convergence criterion when the overall velocity of the mechanism is large, i.e.,

$$\varepsilon_{tol} \;=\; \varepsilon_{abs} + \varepsilon_{vel}\,\|\dot{\mathbf{v}}\|$$

Energy norms

The product of residual vector ${}^i\hat{\mathbf{R}}_k$ and displacement correction ${}^i\delta\mathbf{r}_k$ form an incremental energy term. The advantage of using this quantity in convergence testing is the automatic same unit of energy for all DOFs regardless of translation/force, rotation/torque, or generalized DOF/generalize force.

We define energy norm analogous to the infinite and scaled vector norms respectively, as the max DOF energy and average energy

$$\text{Max DOF energy:} \quad E_{\max} = \max_{j=1,n} \left| {}^i\delta r_{kj} \, {}^i\hat{R}_{kj} \right| \tag{7.41}$$

$$\text{Average energy:} \quad E_{\text{avg}} = \frac{1}{n} \sum_{j=1,n} \left| {}^i\delta r_{kj} \, {}^i\hat{R}_{kj} \right| \tag{7.42}$$

Remark: *Structures close to a buckling state will often be close to force equilibrium (low residual norm), while the displacements are largely undetermined (high displacement correction norm). Similarly, structures with very stiff members often are close to the correct position (low displacement correction norm), whereas the unbalanced forces are high (high residual norm). Both of these cases can lead to convergence problems when the convergence criterion is based on displacements and/or force equilibrium. In these cases the energy norms often give a more "balanced" picture and can improve stability in passing problem areas in the dynamics simulation.*

7.4 Newmark integration with numerical damping

7.4.1 The Hilber–Hughes–Taylor method

The α-method by Hilber, Hughes and Taylor [8] circumvents the problem of not being able to introduce numerical damping without a loss of accuracy as in the traditional Newmark algorithm. This is achieved by modifying the equilibrium equation (7.3), as follows

$$\mathbf{F}^{\text{I}}_{k+1} + (1+\alpha)\mathbf{F}^{\text{D}}_{k+1} - \alpha\mathbf{F}^{\text{D}}_{k} + (1+\alpha)\mathbf{F}^{\text{S}}_{k+1} - \alpha\mathbf{F}^{\text{S}}_{k} \;=\; (1+\alpha)\mathbf{Q}_{k+1} - \alpha\mathbf{Q}_{k} \tag{7.43}$$

where α adjusts the amount of numerical damping.

The HHT-α method gives efficient high-frequency dissipation and is unconditionally stable for parameters in the range

$$\alpha \in [-\frac{1}{3}, 0], \qquad \gamma = \frac{1}{2}(1-2\alpha), \qquad \text{and} \qquad \beta = \frac{1}{4}(1-\alpha)^2 \tag{7.44}$$

Fedem is using $\alpha = -0.1$, which gives $\gamma = 0.6$ and $\beta = 0.3025$.

Remark: *For simulations involving high-speed rotations, the α-method can give less accurate estimates for the rigid body rotational velocities than the standard Newmark method. The latter method might give better results and allow larger time increments in such cases. Note, however, that the standard Newmark method requires more structural damping than the α-method. With the use of stiffness proportional damping, no artificial damping against the rigid body rotation is introduced.*

Equation (7.43) can also be expressed as

$$\mathbf{F}^{\mathrm{I}}_{k+1} + (1+\alpha)\mathbf{F}^{\mathrm{D}}_{k+1} + (1+\alpha)\mathbf{F}^{\mathrm{S}}_{k+1} = (1+\alpha)\mathbf{Q}_{k+1} - \alpha\mathbf{F}^{\mathrm{I}}_{\mathrm{actual}} \tag{7.45}$$

where $\mathbf{F}^{\mathrm{I}}_{\mathrm{actual}}$ represents the actual inertia force at increment k, computed from the equilibrium equation (7.3), i.e.

$$\mathbf{F}^{\mathrm{I}}_{\mathrm{actual}} = \mathbf{Q}_k - \mathbf{F}^{\mathrm{D}}_k - \mathbf{F}^{\mathrm{S}}_k \tag{7.46}$$

This modified equilibrium equation is integrated by the standard Newmark method. Hence, the finite difference expressions (7.14)–(7.21) are retained.

By inserting the internal force linearizations from equations (7.6)–(7.8) into equation (7.45), we obtain

$$\begin{aligned} \mathbf{F}^{\mathrm{I}}_k + \mathbf{M}_k\Delta\ddot{\mathbf{r}}_k &+ (1+\alpha)\left(\mathbf{F}^{\mathrm{D}}_k + \mathbf{C}_k\Delta\dot{\mathbf{r}}_k\right) \\ &+ (1+\alpha)\left(\mathbf{F}^{\mathrm{S}}_k + \mathbf{K}_k\Delta\mathbf{r}_k\right) \\ &= (1+\alpha)\,\mathbf{Q}_{k+1} - \alpha\mathbf{F}^{\mathrm{I}}_{\mathrm{actual}} \end{aligned} \tag{7.47}$$

or

$$\begin{aligned} &\mathbf{M}_k\Delta\ddot{\mathbf{r}}_k + (1+\alpha)\,\mathbf{C}_k\Delta\dot{\mathbf{r}}_k + (1+\alpha)\,\mathbf{K}_k\Delta\mathbf{r}_k \\ &\quad = (1+\alpha)\left(\mathbf{Q}_{k+1} - \mathbf{F}^{\mathrm{D}}_k - \mathbf{F}^{\mathrm{S}}_k\right) - \mathbf{F}^{\mathrm{I}}_k - \alpha\mathbf{F}^{\mathrm{I}}_{\mathrm{actual}} \end{aligned} \tag{7.48}$$

Inserting the Newmark incremental acceleration and velocity expressions from equations (7.18) and (7.20), resepctively, into equation (7.48) now yields

$$\begin{aligned} &\mathbf{M}_k\left(\frac{1}{\beta h^2}\Delta\mathbf{r}_k - \mathbf{a}_k - \ddot{\mathbf{r}}_k\right) + \\ &(1+\alpha)\mathbf{C}_k\left(\frac{\gamma}{\beta h}\Delta\mathbf{r}_k - \mathbf{d}_k - \dot{\mathbf{r}}_k\right) + \\ &(1+\alpha)\mathbf{K}_k\Delta\mathbf{r}_k = (1+\alpha)\left(\mathbf{Q}_{k+1} - \mathbf{F}^{\mathrm{D}}_k - \mathbf{F}^{\mathrm{S}}_k\right) - \mathbf{F}^{\mathrm{I}}_k - \alpha\mathbf{F}^{\mathrm{I}}_{\mathrm{actual}} \end{aligned} \tag{7.49}$$

which after rearranging the known quantities on the right hand side becomes

$$\mathbf{N}_k\Delta\mathbf{r}_k = \widehat{\mathbf{R}}_k \tag{7.50}$$

with

$$\mathbf{N}_k = \frac{1}{\beta h^2}\mathbf{M}_k + \frac{(1+\alpha)\gamma}{\beta h}\mathbf{C}_k + (1+\alpha)\mathbf{K}_k \tag{7.51}$$

and

$$\begin{aligned}\widehat{\mathbf{R}}_k &= (1+\alpha)\left(\mathbf{Q}_{k+1} + \mathbf{C}_k(\mathbf{d}_k + \dot{\mathbf{r}}_k) - \mathbf{F}_k^{\mathrm{D}} - \mathbf{F}_k^{\mathrm{S}}\right) \\ &\quad + \mathbf{M}_k(\mathbf{a}_k + \ddot{\mathbf{r}}_k) - \mathbf{F}_k^{\mathrm{I}} - \alpha\mathbf{F}_{\mathrm{actual}}^{\mathrm{I}}\end{aligned} \tag{7.52}$$

If we assume that the inertia- and damping forces are linear with respect to the total acceleration and velocity, respectively, we have that $\mathbf{F}_k^{\mathrm{D}} = \mathbf{C}_k\dot{\mathbf{r}}_k$ and $\mathbf{F}_k^{\mathrm{I}} = \mathbf{M}_k\ddot{\mathbf{r}}_k$, and equation (7.52) reduces to the following

$$\widehat{\mathbf{R}}_k = (1+\alpha)\left(\mathbf{Q}_{k+1} + \mathbf{C}_k\mathbf{d}_k - \mathbf{F}_k^{\mathrm{S}}\right) + \mathbf{M}_k\mathbf{a}_k - \alpha\mathbf{F}_{\mathrm{actual}}^{\mathrm{I}} \tag{7.53}$$

The equation (7.53) is used as the default force predictor in Fedem, optionally also replacing $\mathbf{F}_{\mathrm{actual}}^{\mathrm{I}}$ by $\mathbf{F}_k^{\mathrm{I}}$. The linear assumption on the inertia- and damping- forces is a simplification that should not matter much if the mass- and damping characteritiscs are relatively constant during the simulation time. However, for systems consisting of large bodies moving relative to each other, like in contact problems, or systems with significant nonlinear damping, like hydrodynamic drag, etc., it is worth considering using the full expression, equation (7.52).

Remark: *A small error in the predictor force usually has no significance in a non-linear simulation with Newton–Raphson iterations, since then it will iterate towards the correct solution in any case. But if the predictor force is too far off, there is a risk of divergence, or in the worst case, convergence towards an incorrect solution.*

For a linear system, the state at time t_{k+1} given by $\Delta\mathbf{r}_k$, $\Delta\dot{\mathbf{r}}_k$ and $\Delta\ddot{\mathbf{r}}_k$, and the associated inertia, damping and elastic forces forces $\mathbf{F}_{k+1}^{\mathrm{I}}$, $\mathbf{F}_{k+1}^{\mathrm{D}}$ and $\mathbf{F}_{k+1}^{\mathrm{S}}$, respectively, will satisfy the equilibrium equation (7.45). However, for nonlinear systems the increments $\Delta\mathbf{r}_k$, $\Delta\dot{\mathbf{r}}_k$ and $\Delta\ddot{\mathbf{r}}_k$ will in general not satisfy equation (7.45). In order to ensure dynamic equilibrium before advancing to the next increment, the dynamic residual seeks to be minimized through a similar Newton–Raphson procedure as described in Section 7.3. Thus, in each iteration we have to solve the linearized system of equations

$${}^{i}\mathbf{N}_k\,{}^{i}\Delta_k = {}^{i-1}\widehat{\mathbf{R}}_k \tag{7.54}$$

where the Newton matrix ${}^{i}\mathbf{N}_k$ is the same as in equation (7.51), and the right-hand-side vector ${}^{i-1}\widehat{\mathbf{R}}_k$ equals the residual from the previous iteration

$${}^{i-1}\widehat{\mathbf{R}}_k = (1+\alpha)\left({}^{i-1}\mathbf{Q}_{k+1} - {}^{i-1}\mathbf{F}_{k+1}^{\mathrm{D}} - {}^{i-1}\mathbf{F}_{k+1}^{\mathrm{S}}\right) - {}^{i-1}\mathbf{F}_{k+1}^{\mathrm{I}} - \alpha\mathbf{F}_{\mathrm{actual}}^{\mathrm{I}} \tag{7.55}$$

7.4.2 Numerical characteristics of the HHT-α method

The Hilber–Hughes–Taylor α-method introduces numerical damping for the higher frequencies, which is very beneficial for the numerical stability of the time integration. However, "high frequency" is relative to the time step size being used. When using a time step with a sampling rate of 300 Hz, 50-60 Hz is a fairly high frequency compared to the sampling frequency, and thus has a fairly significant numerical damping. With a time step of 800 Hz, the numerical damping at 50-60 Hz is much smaller. The numerical properties of the HHT-α method is described in more detail in [13].

The damping ratio of the HHT-α method is a function of the parameter ωh, see Figure 7.1. Reduction in the time step size, h, will move the damping ratio up in the frequency range with the inverse factor of the time step reduction, e.g., reducing the time step size with a factor of 0.1 will move the numerical damping up in the frequency range with a factor of 10.

The frequency of the response has an error which also is dependent on the frequency parameter ωh. This is shown in Figure 7.2. Note, however, that the error is largely independent of the amount of numerical damping introduced through the parameter α. In order to capture the frequency with sufficient accuracy we use a sufficiently small time step.

Remark: *As a rule of thumb, one should use a time step of one tenth of the highest response frequency one wants capture with a reasonable degree of accuracy.*

7.4.3 The generalized-α method

The generalized-α method by Chung and Hulbert [14] seeks to introduce high frequency dissipation into the numerical solution by interpolating the inertia forces between time increments k and $k+1$ using a factor α_m, and interpolating the elastic, damping and external forces using another factor α_f. The dynamic equilibrium equation (7.3) then takes the form

$$\begin{aligned}(1-\alpha_m)\mathbf{F}^{\mathrm{I}}_{k+1} + \alpha_m \mathbf{F}^{\mathrm{I}}_{k} \; + \\ (1-\alpha_f)\mathbf{F}^{\mathrm{D}}_{k+1} + \alpha_f \mathbf{F}^{\mathrm{D}}_{k} \; + \\ (1-\alpha_f)\mathbf{F}^{\mathrm{S}}_{k+1} + \alpha_f \mathbf{F}^{\mathrm{S}}_{k} \; = \; (1-\alpha_f)\mathbf{Q}_{k+1} + \alpha_f \mathbf{Q}_{k}\end{aligned} \tag{7.56}$$

The Newmark integration of equation (7.56) is a second order accurate algorithm provided that

$$\gamma = \frac{1}{2} - \alpha_m + \alpha_f \tag{7.57}$$

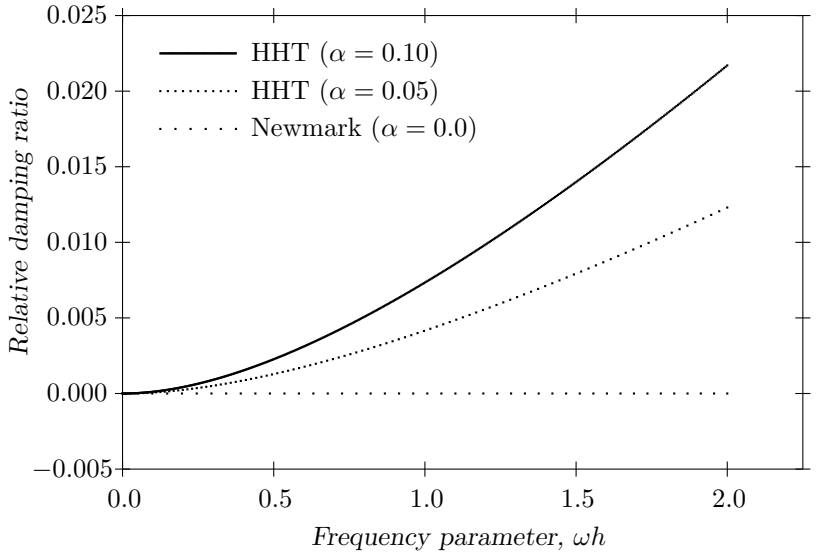


Figure 7.1: Numerical damping ratio for HHT and Newmark algorithm.

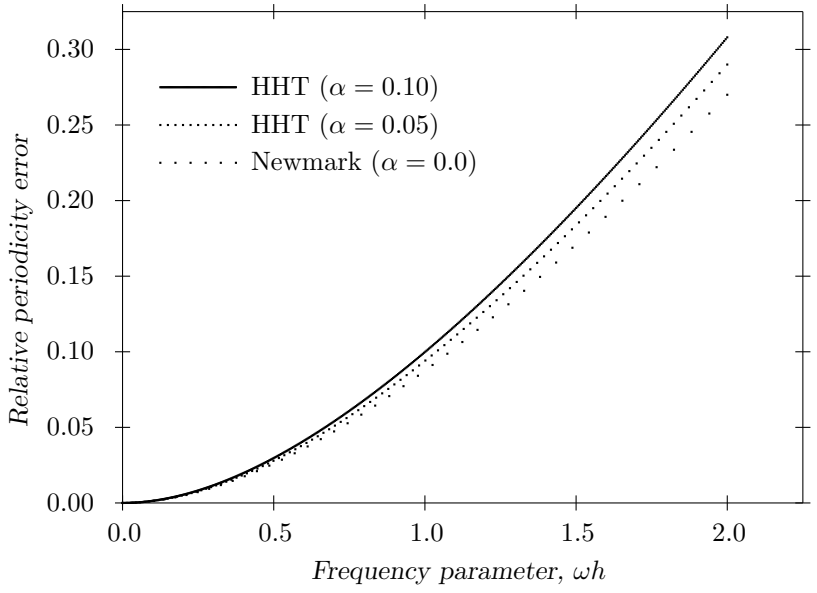


Figure 7.2: Relative periodicity error for HHT and Newmark algorithm.

This algorithm is unconditionally stable if the following conditions are met

$$\alpha_m \leq \alpha_f \leq \frac{1}{2} \quad \text{and} \quad \beta \geq \frac{1}{4} + \frac{1}{2}(\alpha_f - \alpha_m) \tag{7.58}$$

This region of stability is depicted by the shaded area in Figure 7.3, which is bounded by the two lines $\lambda_{1,2}^{\infty} = -1$ corresponding to $\alpha_m \leq \alpha_f$, and $\lambda_3^{\infty} = -1$ corresponding to $\alpha_f \leq \frac{1}{2}$. According to [14], the high frequency modes (relative to the time increment length h) will be damped out if we choose

$$\beta = \frac{1}{2}(1 - \alpha_m + \alpha_f)^2 \tag{7.59}$$

Inserting the internal force linearizations from equations (7.6)–(7.8) into equation (7.56) and using $\mathbf{Q}_{k+1} = \mathbf{Q}_k + \Delta\mathbf{Q}_k$, we obtain the generalized-α

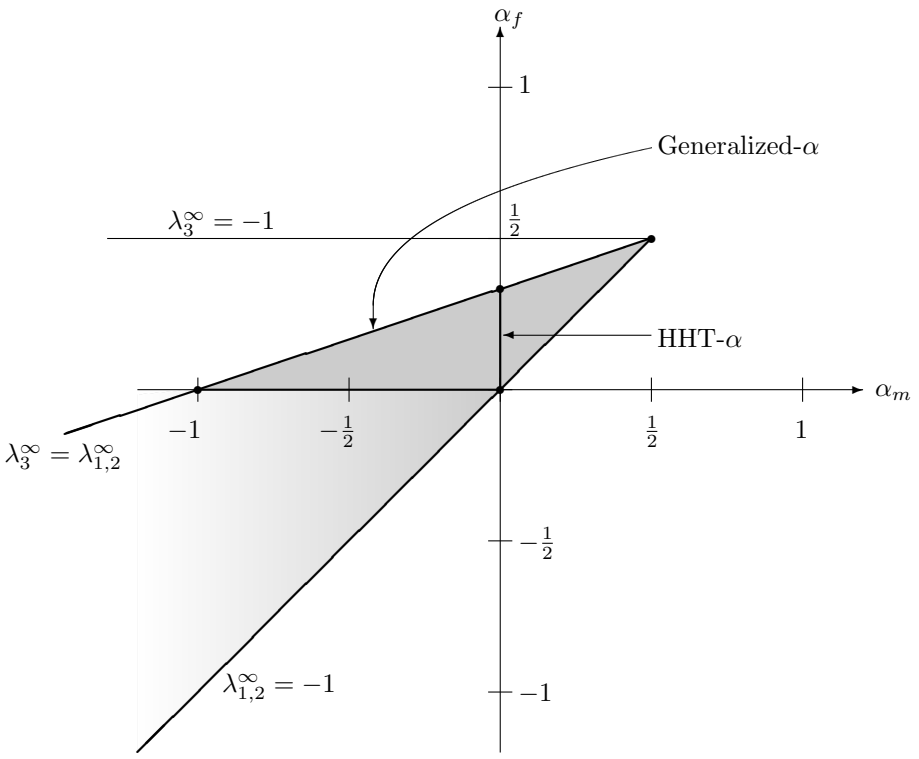


Figure 7.3: Classification of the generalized-α method in the $\alpha_m - \alpha_f$ space. From [14].

equation on incremental form

$$\begin{aligned}&(1-\alpha_m)\mathbf{M}_k\Delta\ddot{\mathbf{r}}_k + \mathbf{F}^{\mathrm{I}}_k + \\ &(1-\alpha_f)\mathbf{C}_k\Delta\dot{\mathbf{r}}_k + \mathbf{F}^{\mathrm{D}}_k + \\ &(1-\alpha_f)\mathbf{K}_k\Delta\mathbf{r}_k + \mathbf{F}^{\mathrm{S}}_k \;=\; (1-\alpha_f)\Delta\mathbf{Q}_k + \mathbf{Q}_k\end{aligned} \tag{7.60}$$

or, by collecting all known quantities of time increment k on the right hand side

$$\begin{aligned}(1-\alpha_m)\mathbf{M}_k\Delta\ddot{\mathbf{r}}_k + (1-\alpha_f)\mathbf{C}_k\Delta\dot{\mathbf{r}}_k \;+\; (1-\alpha_f)\mathbf{K}_k\Delta\mathbf{r}_k \\ = (1-\alpha_f)\Delta\mathbf{Q}_k - \left(\mathbf{F}^{\mathrm{I}}_k + \mathbf{F}^{\mathrm{D}}_k + \mathbf{F}^{\mathrm{S}}_k - \mathbf{Q}_k\right)\end{aligned} \tag{7.61}$$

The last term on the right hand side of equation (7.61) can be recognized as the force residual of time increment k, and as such can be omitted since this should be equal to zero if convergence has been achieved before advancing to time increment $k+1$.

Inserting the Newmark incremental acceleration and velocity expressions from equations (7.18) and (7.20) into equation (7.61) yields

$$\begin{aligned}&(1-\alpha_m)\mathbf{M}_k\left(\frac{1}{\beta h^2}\Delta\mathbf{r}_k - \mathbf{a}_k - \ddot{\mathbf{r}}_k\right) + \\ &(1-\alpha_f)\mathbf{C}_k\left(\frac{\gamma}{\beta h}\Delta\mathbf{r}_k - \mathbf{d}_k - \dot{\mathbf{r}}_k\right) + \\ &(1-\alpha_f)\mathbf{K}_k\Delta\mathbf{r}_k \;=\; (1-\alpha_f)\Delta\mathbf{Q}_k - \left(\mathbf{F}^{\mathrm{I}}_k + \mathbf{F}^{\mathrm{D}}_k + \mathbf{F}^{\mathrm{S}}_k - \mathbf{Q}_k\right)\end{aligned} \tag{7.62}$$

which after rearranging the known quantities on the right hand side becomes

$$\mathbf{N}_k\Delta\mathbf{r}_k \;=\; \widehat{\mathbf{R}}_k \tag{7.63}$$

with

$$\mathbf{N}_k \;=\; \frac{1-\alpha_m}{\beta h^2}\mathbf{M}_k \;+\; \frac{(1-\alpha_f)\gamma}{\beta h}\mathbf{C}_k \;+\; (1-\alpha_f)\mathbf{K}_k \tag{7.64}$$

and

$$\begin{aligned}\widehat{\mathbf{R}}_k \;&= (1-\alpha_f)\Delta\mathbf{Q}_k - \left(\mathbf{F}^{\mathrm{I}}_k + \mathbf{F}^{\mathrm{D}}_k + \mathbf{F}^{\mathrm{S}}_k - \mathbf{Q}_k\right) \\ &\quad + (1-\alpha_m)\mathbf{M}_k(\mathbf{a}_k + \ddot{\mathbf{r}}_k) \;+\; (1-\alpha_f)\mathbf{C}_k(\mathbf{d}_k + \dot{\mathbf{r}}_k)\end{aligned} \tag{7.65}$$

For a linear system, the state at time t_{k+1} given by $\Delta\mathbf{r}_k$, $\Delta\dot{\mathbf{r}}_k$ and $\Delta\ddot{\mathbf{r}}_k$, and the associated inertia, damping and elastic forces forces $\mathbf{F}^{\mathrm{I}}_{k+1}$, $\mathbf{F}^{\mathrm{D}}_{k+1}$ and $\mathbf{F}^{\mathrm{S}}_{k+1}$, respectively, will satisfy the equilibrium equation (7.56). For nonlinear systems, the increments $\Delta\mathbf{r}_k$, $\Delta\dot{\mathbf{r}}_k$ and $\Delta\ddot{\mathbf{r}}_k$ will in general not satisfy equation (7.56). In order to ensure dynamic equilibrium before advancing to

the next time increment, the dynamic residual seeks to be minimized through a similar Newton–Raphson procedure as described in Section 7.3. Thus, in each iteration we have to solve the linearized system of equations

$$ {}^{i}\mathbf{N}_k \, {}^{i}\delta\mathbf{r}_k \;=\; {}^{i-1}\mathbf{R}_k^{\alpha} \tag{7.66} $$

where the Newton matrix ${}^{i}\mathbf{N}_k$ is the same as in equation (7.64), and the right-hand-side vector ${}^{i-1}\mathbf{R}_k^{\alpha}$ equals

$$ \begin{aligned} {}^{i-1}\mathbf{R}_k^{\alpha} \;&= (1-\alpha_f)\, {}^{i-1}\mathbf{Q}_{k+1} + \alpha_f \mathbf{Q}_k \\ &- (1-\alpha_m)\, {}^{i-1}\mathbf{F}^{\mathrm{I}}_{k+1} - \alpha_m \mathbf{F}^{\mathrm{I}}_k \\ &- (1-\alpha_f)\, {}^{i-1}\mathbf{F}^{\mathrm{D}}_{k+1} - \alpha_f \mathbf{F}^{\mathrm{D}}_k \\ &- (1-\alpha_f)\, {}^{i-1}\mathbf{F}^{\mathrm{S}}_{k+1} - \alpha_f \mathbf{F}^{\mathrm{S}}_k \end{aligned} \tag{7.67} $$

7.5 Structural damping

The reduced superelement damping matrix is not developed in Fedem. Instead, proportional damping is used. If we assume that the damping force in a superelement is proportional to the velocity of each mass point, we have

$$ \mathbf{c} \;=\; \alpha_1 \mathbf{m} \tag{7.68} $$

where α_1 is a constant. Similarly, if we assume the damping force is proportional to the strain velocity in each point, we have

$$ \mathbf{c} \;=\; \alpha_2 \mathbf{k} \tag{7.69} $$

where α_2 is another constant. The combination of these two assumptions produces the damping matrix of *Rayleigh-damping* or *proportional damping*

$$ \mathbf{c} \;=\; \alpha_1 \mathbf{m} + \alpha_2 \mathbf{k} \tag{7.70} $$

The damping ratio for the natural frequencies can now be calculated from

$$ \lambda_i \;=\; \frac{1}{2}\left(\frac{\alpha_1}{\omega_i} + \alpha_2 \omega_i\right) \tag{7.71} $$

where α_1 damps out lower vibration modes while α_2 damps out higher modes. If the damping ratios λ_i for two vibration modes are selected, the corresponding constants of proportionality, α_1 and α_2 may be calculated from

$$ \begin{aligned} \alpha_1 &= \frac{2\omega_1\omega_2}{\omega_2^2 - \omega_1^2}\left(\lambda_1\omega_2 - \lambda_2\omega_1\right) \\ \alpha_2 &= \frac{2\left(\omega_2\lambda_2 - \omega_1\lambda_1\right)}{\omega_2^2 - \omega_1^2} \end{aligned} \tag{7.72} $$

where ω_1 and ω_2 are the circle frequencies and λ_1 and λ_2 are the damping ratios for the selected vibration modes, see Figure 7.4.

When using component modes (see Section 3.2), the reduced superelement matrices are partitioned into sub-matrices associated with the retained nodal DOFs and component modes, respectively. It is then possible to assign individual Rayleigh damping factors for each component mode. In this case, exploiting that $\mathbf{k}_{12} = \mathbf{k}_{21}^T = \mathbf{0}$ and $\mathbf{m}_{12} = \mathbf{m}_{21}^T$, equation (7.70) reads

$$\begin{bmatrix} \mathbf{c}_{11} & \mathbf{c}_{12} \\ \mathbf{c}_{21} & \mathbf{c}_{22} \end{bmatrix} = \begin{bmatrix} \alpha_1 \mathbf{m}_{11} & (\boldsymbol{\alpha}_m \mathbf{m}_{21})^T \\ \boldsymbol{\alpha}_m \mathbf{m}_{21} & \boldsymbol{\alpha}_m \mathbf{m}_{22} \end{bmatrix} + \begin{bmatrix} \alpha_2 \mathbf{k}_{11} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\alpha}_k \mathbf{k}_{22} \end{bmatrix} \tag{7.73}$$

where $\boldsymbol{\alpha}_m = \lceil \alpha_{mi} \rfloor$ and $\boldsymbol{\alpha}_k = \lceil \alpha_{ki} \rfloor$ are diagonal matrices containing the component mode damping factors. Note that $\mathbf{m}_{22}$ and $\mathbf{k}_{22}$ both are diagonal matrices, see Section 3.2.3.

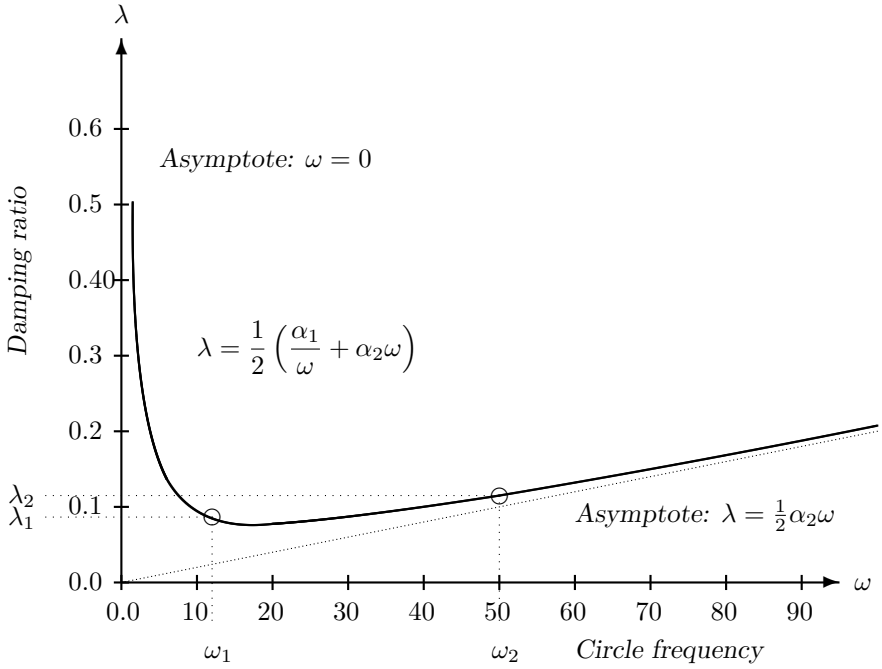


Figure 7.4: A typical relationship between damping and natural frequency arising from the specification of damping ratio at the frequencies. ($\alpha_1 = 1.5$ and $\alpha_2 = 0.004$).

7.6 Evaluation of the Newton matrix

The Newton matrix is a function of the integration parameters γ and β and of the time increment size size h, see equation (7.23). In Section 3.2, mass and stiffness matrices are developed for each of the substructures and reduced to superelement matrices by CMS-reduction techniques. The tangent stiffness matrix of each substructure is then established at the current configuration through the co-rotational formulation outlined in Chapter 4.

A superelement Newton matrix can now be calculated from the following relation (refer to equations (7.23) and (7.70))

$$\mathbf{n}_i \;=\; \mathbf{k}_i + \frac{\gamma}{\beta h}\left(\alpha_1 \mathbf{m}_i + \alpha_2 \mathbf{k}_i\right) + \frac{1}{\beta h^2}\mathbf{m}_i \tag{7.74}$$

where

$\mathbf{n}_i$ reduced superelement Newton matrix

$\mathbf{k}_i$ reduced superelement stiffness matrix

$\mathbf{m}_i$ reduced superelement mass matrix

Each time the integration algorithm requires a new Newton matrix (refer to equations (7.22) and (7.33)), the superelement Newton matrices are transformed to the actual directions of the system level DOFs by

$$\bar{\mathbf{n}}_i \;=\; \mathbf{T}_{SEi}^T \mathbf{n}_i \mathbf{T}_{SEi} \tag{7.75}$$

resulting in the transformed superelement Newton matrix, $\bar{\mathbf{n}}_i$.

The superelement Newton matrices are added into the incremental Newton matrix at system level as indicated by

$$\mathbf{N}_k \;=\; \sum_i \mathbf{a}_i^T \bar{\mathbf{n}}_i \mathbf{a}_i \tag{7.76}$$

where $\mathbf{a}_i$ are incidence matrices that represent superelement topology at system level.

The stiffness for a spring may be a constant or, in the nonlinear case, a variable that is dependent on the spring deflection. For axial springs, the stiffness is transformed to the directions of the connected supernodes and then added directly to the Newton system matrix. For joint springs, stiffness is added to the diagonal of the involved DOF of the system's Newton matrix.

The damping coefficient of a damper may be a constant, or in the nonlinear case a variable that is dependent on the damper velocity. For the Newton matrix, the damping coefficient must be modified by the factor $\frac{\gamma}{\beta h}$

(see equations (7.23) and (7.34)). For axial dampers, the modified damping coefficient is transformed to the actual directions for the connected supernodes and then added to the Newton system matrix. For joint dampers, the modified damping coefficients are added to the diagonal of the involved DOF of the system's Newton matrix.

Additional masses may be regarded as constants during simulation. For the Newton matrix, additional masses must be modified by the factor $\frac{1}{\beta h^2}$ (refer to equations (7.23) and (7.34)). The modified additional masses are added to the diagonal elements for the specified DOFs of the Newton matrix.

7.7 Evaluation of the force vector

The force vector must be evaluated for each time increment and iteration (refer to equations (7.22) and (7.33)).

7.7.1 External forces

The external forces at time increment $k+1$, $\mathbf{Q}_{k+1}$ consist of the following contributions:

- Superelement gravitational forces
- Concentrated external forces in supernodes
- Gravitational forces from additional masses

The gravitational forces are transformed to actual directions of the DOFs at system level by means of the superelement transformation matrix $\mathbf{T}_{SEi}$

$$\bar{\mathbf{g}}_i \;=\; \mathbf{T}_{SEi}^T \mathbf{g}_i \tag{7.77}$$

The transformed superelement's gravitational forces are then added into the force vector by

$$\mathbf{Q}_{k+1} \;=\; \mathbf{Q}_{k+1} + \sum_i \mathbf{a}_i^T \bar{\mathbf{g}}_i \tag{7.78}$$

Direction and magnitude for specified concentrated loads are calculated for the actual position of the mechanism, then transformed to the supernode direction and added into the system vector $\mathbf{Q}_{k+1}$. For loading on specified DOFs at system level, the magnitude at that position is added directly into $\mathbf{Q}_{k+1}$.

For additional masses added to translational DOFs, a gravitational force is calculated by multiplying the component of the gravitational vector along the

actual DOF by the specified mass. These gravitational forces are then added to $\mathbf{Q}_{k+1}$ for the DOFs in question.

7.7.2 Stiffness forces

The superelement stiffness forces are transformed to the actual directions of the DOFs for superelement i, i.e.

$$\bar{\mathbf{S}}_i \;=\; \mathbf{T}_{SEi}^T \mathbf{S}_i \tag{7.79}$$

(see equation (7.77)) and formally added into the force vector by

$$\mathbf{F}^{\mathrm{S}} \;=\; \sum_i \mathbf{a}_i^T \bar{\mathbf{S}}_i \tag{7.80}$$

(see equation (7.78)).

In general, a spring element may have a stress-free length that is a function of time, and it may also have a spring stiffness which is a function of the spring deflection (see Section 5.1). At a given configuration of the mechanism, the specified stress-free length of the spring, l_0, is evaluated and subtracted from the actual spring length, l, to give the actual spring deflection

$$\delta \;=\; l(\mathbf{r}) - l_0(t) \tag{7.81}$$

In a typical nonlinear case, the spring force is found by integrating the stiffness over a change in the deflection. For axial springs, the evaluated spring force is transformed to the direction of the end supernodes and added into the force vector similar to equations (7.79) and (7.80).

For joint springs, the force (or torque) is calculated similarly to that of axial springs; however, the force is added directly to the involved DOF of the load vector without any further transformation.

7.7.3 Inertia and damping forces

Taking into account that

$$\mathbf{F}_k^{\mathrm{I}} \;=\; \mathbf{M}_k \ddot{\mathbf{r}}_k \tag{7.82}$$

$$\mathbf{F}_k^{\mathrm{D}} \;=\; \mathbf{C}_k \dot{\mathbf{r}}_k \tag{7.83}$$

the right-hand-side vector defined by equation (7.24) reduces to

$$\Delta\hat{\mathbf{Q}}_k \;=\; \mathbf{Q}_{k+1} + \mathbf{C}_k \mathbf{d}_k + \mathbf{M}_k \mathbf{a}_k - \mathbf{F}_k^{\mathrm{S}} \tag{7.84}$$

where $\mathbf{a}_k$ and $\mathbf{d}_k$ are the *predicted* acceleration and velocity, defined by equations (7.19) and (7.21), respectively.

For the first iteration within each time increment, the effective inertia forces are calculated by replacing $\ddot{\mathbf{r}}_k$ with $-\mathbf{a}_k$ in equation (7.82) and $\dot{\mathbf{r}}_k$ with $-\mathbf{d}_k$ in equation (7.83). To avoid assembling the system mass and damping matrix explicitly, the corresponding superelement accelerations and velocities are extracted from the system vectors and transformed to the local superelement direction. The local superelement mass and damping matrices are then multiplied by the corresponding local acceleration and velocity vectors to produce the superelement inertia and damping forces, respectively. These forces are then transformed to the actual supernode directions and added into the system vector (compare with equations (7.77) and (7.78)).

In general, a damper element has a damping coefficient that is a function of the damper velocity (see Section 5.2). For a given velocity, the damping coefficient and associated force or torque is calculated. The force/torque is then added directly to the involved DOF of the load vector without any further transformation for joint dampers. For axial dampers, the force is transformed in a similar manner as described above for the axial springs.

The inertia forces (or torques) from additional masses on specified DOFs of the mechanism are calculated by multiplying the DOF acceleration by the magnitude of the mass. These forces are then added to the corresponding DOFs of the load vector.

7.7.4 Forces due to prescribed motion

Prescribed motions in a mechanism can be imposed by introducing stiff springs with a specified stress-free length variation as a function of time, as described in Section 5.1. This is equivalent to a penalty-enforcement of the motion constraint, where the spring stiffness serves as the penalty parameter.

In Fedem, it is also possible of enforce the prescribed motion explicitly, by eliminating the DOF that is prescribed from the system of equations (7.22) and (7.33). This is usually more efficient, since the number of unknowns then is reduced, and it may also yield a more stable solution process since we don't have to deal with penalty parameters. It is possible to prescribe motions in joint DOFs or directly on triads.

For a given DOF n, we assume that the motion in terms of total displacement relative to the initial position is a known function of time, $u_n = u_n(t)$. The right-hand-side vector of the equation system (7.22), given

by equation (7.84) above, is then modified as follows

$$\Delta\hat{\mathbf{Q}}_k = \mathbf{Q}_{k+1} - \mathbf{N}_k\boldsymbol{\delta}_n\Delta u_k + \mathbf{C}_k\mathbf{d}_k + \mathbf{M}_k\mathbf{a}_k - \mathbf{F}_k^{\mathrm{S}} \tag{7.85}$$

where $\Delta u_k = u_n(t_{k+1}) - u_n(t_k)$ is the increment in the prescribed motion from the previous time increment, k, to the next increment, $k+1$, and $\boldsymbol{\delta}_n$ is a constant vector with the value 1 at the location corresponding to the prescribed DOF n, and zero elsewhere. The size of the equation system (7.22) is then reduced by 1, by removing the n'th column and row of the Newton matrix and the right-hand-side vector, since this DOF no longer is an unknown.

The equation system of the iterations (7.33) is modified in a similar manner, except that there is no additional force term here since the iterative correction on the prescribed DOF, ${}^i\Delta u_k$, by definition always is zero.

One can also use a prescribed velocity or acceleration in the same manner. We then use the Newmark integration parameters to derive the equivalent prescribed displacement, which then is inserted into equation (7.85). With a prescribed velocity function $v_n(t)$, we find the equivalent displacement increment to impose from equation (7.20) as

$$\Delta u_k \;=\; \frac{\beta h}{\gamma}\left(v_n(t_{k+1}) - v_n(t_k)\right) + h\, v_n(t_k) + \left(\frac{1}{2} - \frac{\beta}{\gamma}\right) h^2 \ddot{r}_k \tag{7.86}$$

Similarly, with a prescribed acceleration function $a_n(t)$, we can find an equivalent displacement from equation (7.18) as

$$\Delta u_k \;=\; \beta h^2\left(a_n(t_{k+1}) - a_n(t_k)\right) + h\,\dot{r}_k + \frac{h^2}{2} a_n(t_k) \tag{7.87}$$

7.8 Quasi-static equilibrium

Dynamic simulation of mechanism motion should start from an equilibrium position to avoid that unbalanced forces in the mechanism from the initial configuration cause a false superimposed vibration in the simulation results. This can happen when the simulation starts from both a stationary position and a position with initial velocities and acceleration. Unbalanced forces for the stationary position arise from gravitation forces, initially loaded springs and initial external loading, as well as from positioning inaccuracies in the mechanism modeling. Equilibrium for the initial positions of mechanisms with initial velocities and accelerations also includes the unbalanced inertia and damping forces.

The inclusion of possible inertia and damping effects in the initial equilibrium iteration justifies the term *quasi-static equilibrium* iteration. Since the initial velocities and accelerations are either zero or specified, their correction in the inertia and damping terms on the left-hand side of the iteration equation (7.27) cancel out. The quasi-static iteration equation therefore reduces to

$$ {}^{i}\mathbf{K}_0 {}^{i}\delta\mathbf{r}_0 \;=\; {}^{i-1}\mathbf{Q}_0 - \left[{}^{i-1}\mathbf{F}_0^{\mathrm{I}} + {}^{i-1}\mathbf{F}_0^{\mathrm{D}} + {}^{i-1}\mathbf{F}_0^{\mathrm{S}}\right] \tag{7.88} $$

Here, the subindices have been changed to 0, indicating the initial position. In the case of prescribed motion, the first iteration is instead governed by

$$ {}^{1}\mathbf{K}_0 {}^{1}\delta\mathbf{r}_0 \;=\; {}^{0}\mathbf{Q}_0 - \left[{}^{1}\mathbf{K}_0\boldsymbol{\delta}_n u_n(t_0) + {}^{0}\mathbf{F}_0^{\mathrm{I}} + {}^{0}\mathbf{F}_0^{\mathrm{D}} + {}^{0}\mathbf{F}_0^{\mathrm{S}}\right] \tag{7.89} $$

If the rigid body mechanism motions all are either constrained by springs—as in a car suspension system—or fixed, the stiffness matrix is nonsingular and equation (7.88) may be used to iterate for the equilibrium position. When mechanism motion is controlled by force input, equation (7.88) may be singular and *additional boundary conditions* must be introduced into the equation to eliminate the rigid body DOFs of the mechanism. In this case, extra boundary conditions that are effective for the quasi-static equilibrium iterations only, are usually specified for DOFs where the external forces are applied. This modifies equation (7.88) so that it becomes nonsingular, and the iteration can proceed until equilibrium is reached within a specified tolerance.

7.8.1 Equilibrium iteration procedure

Starting from equation (7.88), possibly modified by additional boundary conditions, the iteration will follow the same algorithm as in the dynamic equilibrium iteration described above, with the exception that the stiffness matrix $\mathbf{K}_0$ replaces the Newton matrix from the dynamic case. For each iteration the initial position of the mechanism is improved from

$$ {}^{i}\mathbf{r}_0 \;=\; {}^{i-1}\mathbf{r}_0 + {}^{i}\delta\mathbf{r}_0 \tag{7.90} $$

The updating of the velocity and acceleration vector in the dynamic iteration procedure is omitted in the quasi-static case. The stiffness matrix is evaluated in the same way as the Newton matrix described in Section 7.6. All the stiffness terms from superelement matrices and springs are kept, while the mass and damping terms from superelement matrices, additional masses, and dampers are skipped.

The external force vector ${}^{i-1}\mathbf{Q}_0$ in equation (7.88) is evaluated in the same way as in the dynamic case for the initial time. The inertia and damping forces, ${}^{i-1}\mathbf{F}_0^{\mathrm{I}}$ and ${}^{i-1}\mathbf{F}_0^{\mathrm{D}}$, are evaluated based on constant velocities and accelerations during iteration. However, these vectors may also change during iteration due to updated positions of the mechanism. The stiffness forces ${}^{i-1}\mathbf{F}_0^{\mathrm{S}}$ are evaluated in exactly the same way as in the dynamic case.

The iteration based on equation (7.88) is called Newton–Raphson iteration, or modified Newton–Raphson if the stiffness matrix is kept constant during some or all iterations. If the stiffness matrix is kept constant during iteration, the last evaluated and triangularized stiffness matrix is kept for new iterations. Only the right-hand-side vector of equation (7.88) is then evaluated for each iteration, and the increments for improving the mechanism position are evaluated through backward substitution. When many iterations are necessary, this can save computational effort.

7.9 Frequency Response Analysis

Frequency response analysis is an alternative approach to compute the structural response due to steady state excitation given in the frequency domain. The excitations are applied forces and/or motions, like displacements, velocities or accelerations. Two types of analysis are available [15]:

- *Direct frequency response* −The response will be computed by solving a set of coupled equations by using complex arithmetics.
- *Modal frequency response* −This method uses the decomposition based on eigenmodes. A certain number of modes, called the eigenspace, will be used for the response calculation and reduces the overall system size.

In the end, the modal frequency response will yield exactly the same answer as the direct frequency response, provided that all modal degrees of freedom are included in the analysis. However, the strength of the modal approach comes from the idea that the solution is very close to the direct approach by using significantly fewer modal degrees of freedom than physical degrees of freedom.

7.9.1 Direct frequency response analysis

In direct frequency response analysis, the response is computed at discrete excitation frequencies by solving a set of coupled matrix equations. The equation of damped forced vibration with harmonic excitation is given by:

$$\mathbf{M}\ddot{\mathbf{r}}(t) + \mathbf{C}\dot{\mathbf{r}}(t) + \mathbf{K}\mathbf{r}(t) \;=\; \mathbf{P}(\omega)e^{i\omega t} \tag{7.91}$$

The load in equation (7.91) is introduced as a complex vector, which is more convenient to solve for. From the physical point of view, the load can be real or imaginary, or both.

For harmonic motion (which is the basis of a frequency response analysis), a harmonic solution of the following form will be assumed:

$$\mathbf{r}(t) \;=\; \mathbf{u}(\omega)e^{i\omega t} \tag{7.92}$$

where $\mathbf{r}(t)$ is the complex displacement vector, and ω is the circle frequency of the periodic load and response. Taking the first and second derivatives of equation (7.92), the following is obtained:

$$\dot{\mathbf{r}}(t) \;=\; i\omega\,\mathbf{u}(\omega)e^{i\omega t} \tag{7.93}$$

$$\ddot{\mathbf{r}}(t) \;=\; -\omega^2\mathbf{u}(\omega)e^{i\omega t} \tag{7.94}$$

When the above expressions are substituted into equation (7.91), we obtain:

$$-\omega^2\mathbf{M}\mathbf{u}(\omega)e^{i\omega t} + i\omega\mathbf{C}\mathbf{u}(\omega)e^{i\omega t} + \mathbf{K}\mathbf{u}(\omega)e^{i\omega t} \;=\; \mathbf{P}(\omega)e^{i\omega t} \tag{7.95}$$

which simplifies to

$$-\omega^2\mathbf{M}\mathbf{u}(\omega) + i\omega\mathbf{C}\mathbf{u}(\omega) + \mathbf{K}\mathbf{u}(\omega) \;=\; \mathbf{P}(\omega) \tag{7.96}$$

The expression (7.96) represents a system of equations with complex coefficients if damping is included or the applied loads have phase angles. This equation of motion is solved for given forcing frequencies, ω, in the same manner as for linear static problems, but using complex arithmetics.

7.9.2 Modal frequency response analysis

The modal frequency response analysis method uses the mode shapes of the structure to reduce the size of the equation system. It uncouples the equations of motion thereby making the numerical solution more efficient. Since the mode shapes typically are computed as part of the characterization of the structure, modal frequency response is a natural extension of a normal mode analysis.

As a first step in the formulation, the variables are transformed from physical coordinates $\mathbf{u}(\omega)$ to modal coordinates by assuming

$$\mathbf{u}(\omega) \;=\; \boldsymbol{\phi}\boldsymbol{\xi}(\omega)e^{i\omega t} \tag{7.97}$$

The equation (7.97) represents an equality if all modes are used. However, since that rarely is the case the equation represents an approximation. Substituting the modal coordinates in equation (7.97) for the physical coordinates in equation (7.96) and simplifying, the following is obtained:

$$-\omega^2\mathbf{M}\boldsymbol{\phi}\boldsymbol{\xi}(\omega) + i\omega\mathbf{C}\boldsymbol{\phi}\boldsymbol{\xi}(\omega) + \mathbf{K}\boldsymbol{\phi}\boldsymbol{\xi}(\omega) \;=\; \mathbf{P}(\omega) \tag{7.98}$$

which represents the equation of motion in modal coordinates.

At this point the equations remain coupled. To uncouple the equations, premultiply by $\boldsymbol{\phi}^T$ to obtain

$$-\omega^2\boldsymbol{\phi}^T\mathbf{M}\boldsymbol{\phi}\boldsymbol{\xi}(\omega) + i\omega\boldsymbol{\phi}^T\mathbf{C}\boldsymbol{\phi}\boldsymbol{\xi}(\omega) + \boldsymbol{\phi}^T\mathbf{K}\boldsymbol{\phi}\boldsymbol{\xi}(\omega) \;=\; \boldsymbol{\phi}^T\mathbf{P}(\omega) \tag{7.99}$$

where the expressions represent:

$\boldsymbol{\phi}^T\mathbf{M}\boldsymbol{\phi}$	:	modal mass matrix
$\boldsymbol{\phi}^T\mathbf{C}\boldsymbol{\phi}$	:	modal damping matrix
$\boldsymbol{\phi}^T\mathbf{K}\boldsymbol{\phi}$	:	modal stiffness matrix
$\boldsymbol{\phi}^T\mathbf{P}$	:	modal load vector

The final step uses the orthogonality property of the mode shapes to formulate the equation of motion in terms of the generalized mass, damping and stiffness matrices, which are diagonal matrices (damping as long as it is defined as a linear combination between the stiffness and mass matrix). Therefore, in this form the modal equations of motion are uncoupled. In this uncoupled form, the equations of motion are written as a set of scalar equations, like

$$-\omega^2 m_i \xi_i(\omega) + i\omega c_i \xi_i(\omega) + k_i \xi_i(\omega) = p_i(\omega) \tag{7.100}$$

where

m_i : i^{th} modal mass
c_i : i^{th} modal damping
k_i : i^{th} modal stiffness
p_i : i^{th} modal load

The modal form is much faster to solve than the direct method because it is a series of uncoupled single-degree-of-freedom equations. Once the individual modal responses are computed, physical responses are recovered as the summation of the modal responses using equation (7.97). These responses are in complex form (magnitude/phase or real/imaginary).

7.9.3 Modal vs. direct frequency response

Some general guidelines can be used when selecting modal frequency response analysis versus direct frequency response analysis. These guidelines are summarized in table 7.1.

In general, larger models may be solved more efficiently in modal frequency response because the numerical solution is a solution of a smaller

Table 7.1: Modal vs. Direct Frequency Response

	Modal	Response
Small model		x
Large model	x	
Few excitation frequencies		x
Many excitation frequencies	x	
High frequency excitation		x
Nonmodal damping		x
Higher accuracy		x

system of uncoupled equations. The modal method is particularly advantageous if the natural frequencies and mode shapes were computed during a previous stage of the analysis. In that case, you simply perform a recover/restart. Using the modal approach to solve the uncoupled equations is very efficient, even for very large numbers of excitation frequencies.

On the other hand, the major portion of the effort in a modal frequency response analysis is the calculation of the modes. For large systems with a large number of modes, this operation can be as costly as a direct solution. This result is especially true for high-frequency excitation. To capture high frequency response in a modal solution, less accurate, high-frequency modes must be computed. For small models with a few excitation frequencies, the direct method may be the most efficient because it solves the equations without first computing modes. The direct method is also more accurate than the modal method because the direct method is not concerned with mode truncation.

7.9.4 Sampling and windowing

The sampling rate defines the upper limit on the frequency that can be used for analysis of the input data, i.e., the forcing function. It describes the number of data samples acquired per unit time. The sampling rate is also denoted as sampling frequency. The upper limit of the frequency band is given by the Nyquist frequency, f_q

$$f_q = \frac{f_s}{2} \quad \text{and} \quad f_s = \frac{1}{\Delta t_s} \tag{7.101}$$

where f_s is the sampling frequency and Δt_s is the sampling time increment. E.g., if the sampling frequency is 100 Hz, then the investigations are limited to 50 Hz and any information beyond this frequency can not be determined. This means that the sampling frequency must be chosen large enough to get the desired information from the input data.

Handling large amounts of input data can be done via *segmenting*, which reduces the leakage in subsequent fourier transformations while minimizing discontinuities between the data segments. The duration of each segment is defined by

$$T_w \;=\; N_w \, \Delta t \tag{7.102}$$

where N_w denotes the number of samples in each window. The segmenting procedure consists of multiplying the input data by a finite-length windowing function with an amplitude that varies smoothly and gradually toward zero at the edges. In Fedem, the Hanning window (raised-cosine window) is used.

This windowing function, which is depicted in Figure 7.5, can be seen as one period of a cosine 'raised' so that negative peaks just touch zero.

Figure 7.6 shows a sample input data sequence, for simplicity just a sine function. Each data segment will then be captured by overlapping and window tapering such that the sum of all data segments is equal to the original, except at the ends where the window is still present. An easy way to accomplish this is to use the Hanning window with 50% overlap, as showed in Figure 7.7. The additional dashed black line represents the superposition of the overlapped windows, which results in unity except for at the ends of the sequences where the window is still present. Figure 7.8 shows the tapered data segments for three Hanning windows, which are obtained by multiplying the window function with the input data-sequence. The product is zero-valued outside the interval. All that is left is the part where they coincide, the "view through the window". This input data isolation (tapering) is the main purpose of window functions.

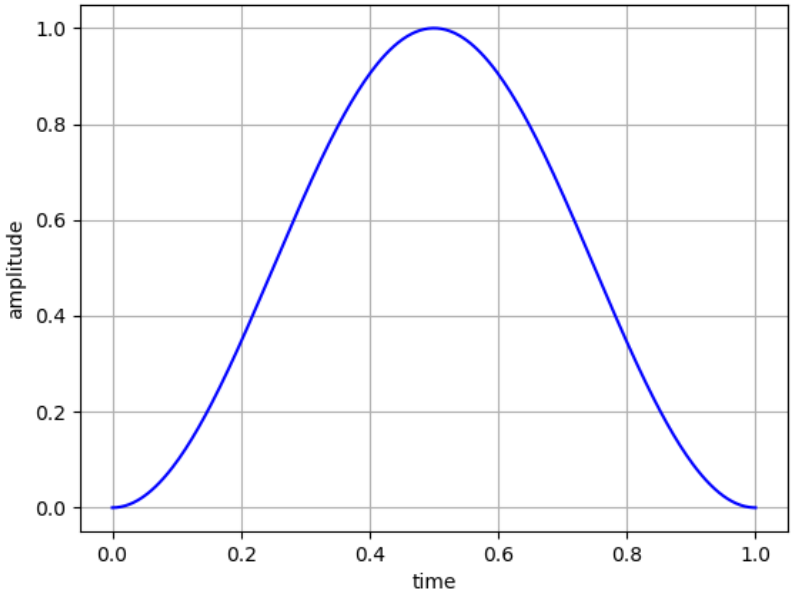


Figure 7.5: The Hanning window.

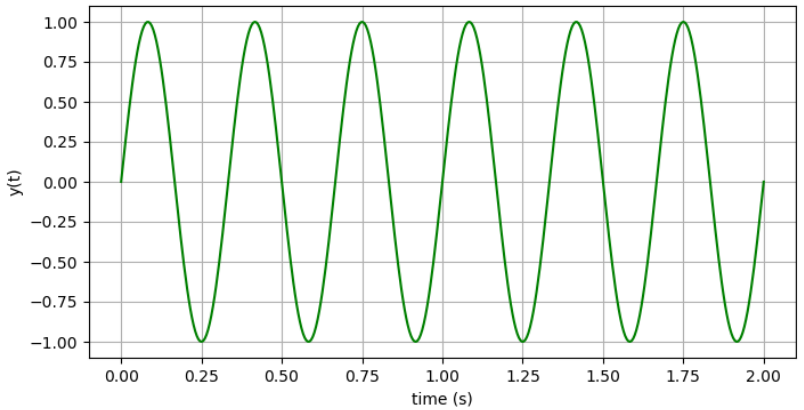


Figure 7.6: Sample input data.

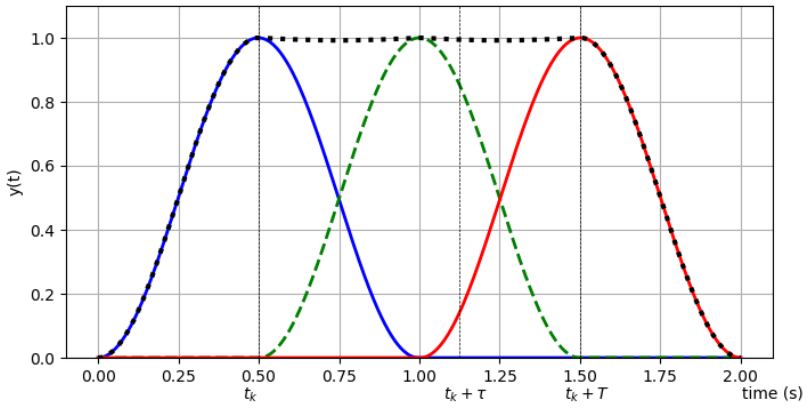


Figure 7.7: Three Hanning windows with 50% overlap.

It is convenient to perform the windowing and overlapping between segments in such a way that the windowed data segments are defined in terms of the absolute time, i.e.,

$$y_k(t) = \begin{cases} y(t)\,\omega(t - t_k) & t \subset [t_k, t_k + T] \\ 0 & t \not\subset [t_k, t_k + T] \end{cases} \tag{7.103}$$

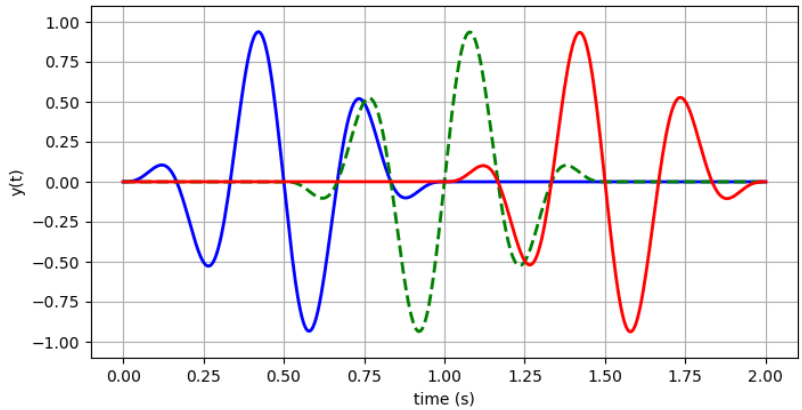


Figure 7.8: Tapered windows.

For each of these tapered windows, a Fast Fourier Transform (FFT) into the frequency domain, and an inverse FFT back to time domain will be applied. After assembling, the 'tapered system response' (see Figure 7.9) will be obtained. As mentioned above, at the ends the window lobes are still active. The systems response in Figure 7.9 is based on an undamped system. The transfer function reduces the amplitude, and there is no phase shift because the system is undamped and not in resonance.

The window size N_w represents the number of samples and herefrom the duration T_w (see equation (7.102)). It depends on the fundamental frequency, intensity and change. In general: *The lower the frequency, the bigger the window size should be.* The default behaviour in Fedem is to not use windowing and to treat the entire time series of the simulation in one go.

7.9.5 Fast fourier transformation (FFT)

A Fourier transform takes a signal in the time domain and switches it into the frequency domain, e.g., it transforms a time series $f(t)$ of N equally- or uniformly spaced points in time $t_n = n\Delta t$, $n = 0, \ldots, N-1$, from the discrete time (or spatial) domain to the discrete frequency domain. The inverse Fourier transform does the inverse transformation from the frequency domain back to the time (or spatial) domain.

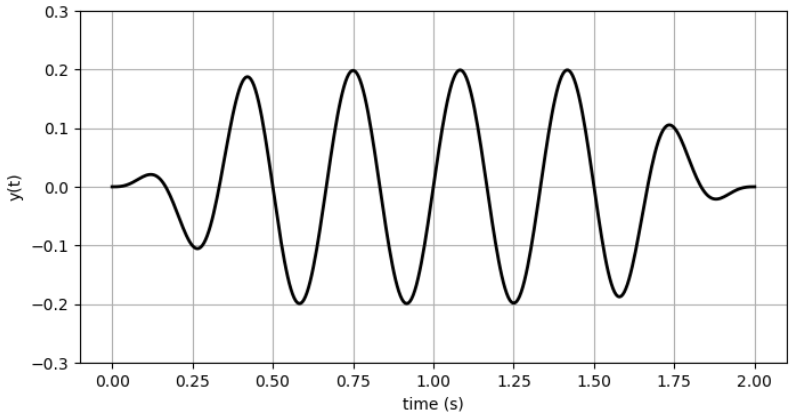


Figure 7.9: Tapered system response

The discrete Fourier transform (DFT) is defined given by

$$X_k \;=\; \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \tag{7.104}$$

where x_n are the values of a signal at equally spaced times $n = 0, \ldots, N-1$. The output X_k is a complex number which encodes the amplitude and phase of a sinusoidal wave with frequency k/N cycles per time unit[2]. The effect of computing X_k is to find the coefficients of a signal approximation by a linear combination of such waves. Since each wave has a whole number of cycles per N time units, the approximation will be periodic with period N. This approximation is given by the inverse Fourier transform

$$x_n \;=\; \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{i2\pi kn}/N \tag{7.105}$$

with

N	number of time samples
n	current sample $(0, \ldots, N-1)$
x_n	value of the signal at time t_n

[2]This comes from Euler's formula: $\exp(\frac{i2\pi kn}{N}) = \cos(\frac{2\pi kn}{N}) + i\sin(\frac{2\pi kn}{N})$.

k	current frequency from 0 Hz up to $N-1$ Hz
X_k	amount of frequency k in the signal (amplitude and phase)
$1/N$	normalization factor
n/N	percent of "going through" time, $2\pi k$ speed in radians/sec

The DFT can be efficiently computed by the Fast Fourier Transform (FFT) algorithm.

Units and spacing

Let Δt denote the spacing between points in time and N be the number of points/time samples. The spacing Δf of the points in frequency is the $1/(N\Delta t)$. The quantity $1/\Delta t$ is called the sampling frequency f_s, which should be at least twice of the highest frequency that is present in the underlying continuous signal.

Nyquist frequency, aliasing, mirroring

When all frequencies present in the underlying continuous signal are below the Nyquist frequency $f_q = \frac{1}{2\Delta t}$, then the discretely sampled time series contains all of the information in the original continuous signal. This is known as the sampling theorem and is a remarkable fact.

If a frequency exceeds the Nyquist frequency, the power from that frequency is still transferred to the FFT, but it gets mapped to a bin in the result as if it had been wrapped around. This phenomena is termed aliasing. One can think of the resulting spectrum as being a circular buffer, representing signal at each frequency modulo the maximum frequency of $\frac{1}{\Delta t}$. A frequency of $\frac{1}{2\Delta t}$ would map to the middle bin, also frequencies of $\frac{3}{2\Delta t}$, $\frac{5}{2\Delta t}$ and so on.

Amplitude and phase

Each number in the result of FFT is a complex number, an encoding of both the amplitude and phase shift of each frequency component. For example, if a 200 Hz component is present, the magnitude of the result at 200 Hz (as given by the absolute value of complex numbers) gives the power density at that frequency. For recovering the original signal, the phase of the component is also relevant. Even though the power density at a certain frequency f is the same for a $\sin(2\pi ft)$, a $\cos(2\pi ft)$, or a $\sin(2\pi ft + \phi_0)$, the phase is different in each of these cases.

The absolute height of an FFT is sometimes confusing. The FFT reflects the total energy of the signal, including the positive and negative frequencies. Only the positive frequencies are from interest and therefore for getting the amplitudes of the input, the FFT must be multiplied by 2.

Efficiency

The FFT algorithm is a fast implementation of the Discrete Fourier Transform (DFT), which is most efficient when the number of elements N in t and f is an even power of 2. The worst efficiency will occur if N is a prime number, the efficiency of the FFT decreases to the efficiency of the DFT itself. A DFT requires $O(N^2)$ steps, a FFT requires $O(N \log(N))$ steps.

7.9.6 Modal damping

Finding proper specification for the structural damping is one of the most challenging input tasks, because verification is possible only by performing a response analysis in the time domine. One way to specify the damping is by means of modal damping, where the frequency-dependent damping ratio as the percentage of the critical damping is specified for each mode. This will then result in a diagonal damping matrix associated with the modes.

The modal damping ratio can be calculated via

$$\zeta_i = \frac{c_i}{c_{\mathrm{cr},i}} \tag{7.106}$$

with the critical damping $c_{\mathrm{cr},i} = 2m_i\omega_i$ and $\omega_i^2 = \frac{k_i}{m_i}$. The important characteristic in modal damping reflects that the damping values are calculated at natural frequencies and not at the excitation frequencies.

Chapter 8 Control System

Mechanisms are often connected to or acted upon by items such as sensors, controllers, and actuators; therefore, a need exists for multidisciplinary mechanism and/or finite-element control simulation. Consequently, a control system has been developed for simulation in a composite system.

8.1 Problem statement

Elements have been designed to model these sensors, controllers, actuators, and so on, which are referred to as control elements or control blocks. Control elements are actually functions, also referred to as control equations, that describe the modeled element. The control elements can be connected together with input and output blocks to perform more complex operations. Once connected the elements are referred to as a control module.

Separate numerical methods are used for the structural and control calculations, so that only minor changes are imposed on the FEM part.

The structural equation constitutes a 2^{nd} order system for the linear case and can be written as:

$$\mathbf{M}\ddot{\mathbf{r}} + \mathbf{C}\dot{\mathbf{r}} + \mathbf{K}\mathbf{r} = \mathbf{Q}(t) \tag{8.1}$$

where $\mathbf{r}$ is the vector of displacement, $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{K}$ are matrices for mass, damping, and stiffness, respectively, and $\mathbf{Q}(t)$ is a vector of time-dependent forces acting on the structure.

The control equations can usually be written in the following form:

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{u}, \mathbf{x}, \mathbf{z}) \tag{8.2}$$

$$\mathbf{O} = \mathbf{g}(t, \mathbf{u}, \mathbf{x}, \mathbf{z}) \tag{8.3}$$

where $\mathbf{u}$, $\mathbf{x}$ and $\mathbf{z}$ are vectors of inputs, state variables, and algebraic variables, respectively. Most control elements are not explicitly time-dependent. Exceptions are elements with inherent clock functions, such as those in a sample and hold element or a multiplexing unit.

The structural and control calculations are coupled when some elements from the displacement vector enter into input vector $\mathbf{u}$ and, in response, some of the forces in $\mathbf{Q}(t)$ are taken from the control variables $\mathbf{x}$ or $\mathbf{z}$. Any remaining inputs of $\mathbf{u}$ can be time functions, such as a controller reference.

The system for solving the structural calculations is based on Newmark's established β-method with (as a general rule) $\gamma = \frac{1}{2}$ and $\beta = \frac{1}{4}$ (see [11]). These parameters correspond to the trapezoidal rule.

The dimensions of the discrete structural calculations are usually much higher than those of the control calculations, and therefore represent the heaviest portion of the computations.

In terms of accuracy, the structural calculations can be expected to limit the time step more than the control calculations. However, cases may exist in which the coupling between the structural and control calculations limits the time step more than either of the two calculations.

The control system module integrates the variables in the differential-algebraic system one step forward from a given state. The solution for the control system part is iterated until there is convergence on each invocation; therefore, these iterations form an inner loop of the dynamic integration iterations.

8.2 Control variables

The system is divided into three types of variables: inputs, state, and algebraic variables. The user defines inputs, while the system automatically determines state and algebraic variables. The inputs are either external time functions such as a controller reference, or sensor inputs that are not changed in the control system.

Control elements are identified by type and have a number of inputs, internal states, and outputs. The internal states are not connected to other modules. In addition, there are a number of parameters the user needs to provide. Figure 8.1 shows a module of type nn, with i inputs, j internal states, k outputs, and m parameters.

The user needs to bear two things in mind concerning this system. First, some parameters are used to carry values from one time step to another. These parameters appear in parentheses in the library and are not provided by the user. Second, the distinction between inputs and outputs cannot be made for some elements before the configuration is set up. This is known as the causality problem and may occur in algebraic relations (see [9]). However, this problem is resolved by the system during initialization.

To configure a control scheme, the user draws a block scheme of modules from the library, as shown in Figure 8.2.

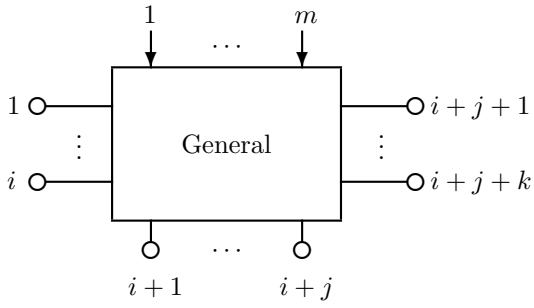


Figure 8.1: The general control module

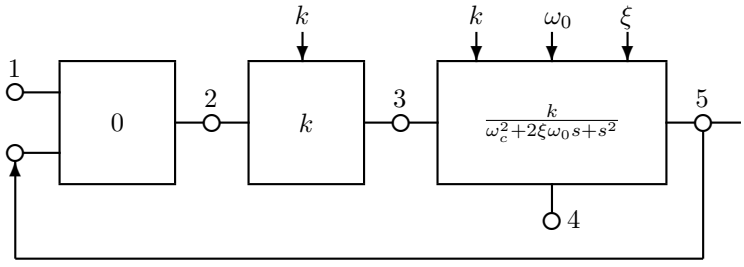


Figure 8.2: Configuration example

8.3 Control system tasks

The control system module manages the three main activities: Initialization, steady state computation, and integration.

8.3.1 Initialization

During initialization the task is to label the state and algebraic variables. For the algebraic variables of equation (8.3), initialization determines which variable to allocate to the function value. Each module must be prepared for this task, which is managed during the initialization process. Algebraic loops or causality problems may require several iterations.

8.3.2 Steady state

With the different types of variables collected in one vector,

$$\mathbf{y} = \begin{bmatrix} \mathbf{u} \\ \mathbf{x} \\ \mathbf{z} \end{bmatrix} \tag{8.4}$$

the set of equations may be written:

$$\mathbf{F}(t, \mathbf{y}, \dot{\mathbf{y}}) = \begin{bmatrix} -\mathbf{u} + \mathbf{s}(t) \\ -\dot{\mathbf{x}} + \mathbf{f}(t, \mathbf{y}) \\ \mathbf{g}(t, \mathbf{y}) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{8.5}$$

Steady state is found by setting the derivative to zero. Applying Newton iteration to equation (8.5) produces the following system which is solved at iteration number k:

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{f}_x & -\mathbf{f}_z \\ \mathbf{0} & -\mathbf{g}_x & -\mathbf{g}_z \end{bmatrix} \begin{bmatrix} \Delta\mathbf{u}^k \\ \Delta\mathbf{x}^k \\ \Delta\mathbf{z}^k \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{f}(t, \mathbf{u}, \mathbf{x}^k, \mathbf{z}^k) \\ \mathbf{g}(t, \mathbf{u}, \mathbf{x}^k, \mathbf{z}^k) \end{bmatrix} \tag{8.6}$$

A standard equation solver is used for the solution of this vector equation.

8.3.3 Time integration

Integration is performed using two numerical methods to estimate local error.

A starting point of equation (8.5) is established for the development of the numerical equations. All the input components can be treated as time functions, even though some of them are determined by the structure calculations. The inputs are not changed in the control calculations. Because they are labeled, the variables in vector **y** do not have to be be ordered according to type; this is for convenience of development notation only.

Since Newmark's β method is of second order, it naturally follows that a method of second order is chosen for the control part also. Therefore, Lobatto IIIC, an implicit, second-order Runge-Kutta method, is selected. Backward Euler, which is first-order and also implicit, is used for local error estimation. Local extrapolation is used, meaning that the integration proceeds with the result from the higher-order method.

The general m-level Runge-Kutta (RK) method for the solution of

$\mathbf{c}$	$\mathbf{A}$
	$\mathbf{b}^T$

Table 8.1: Butcher tableau

1	1
	1

0	1/2	-1/2
1	1/2	1/2
	1/2	1/2

Table 8.2: Backward Euler and Lobatto IIIC

equation (8.5) can be written:

$$F(t + c_i h, Y_i, \dot{Y}_i) = 0 \tag{8.7}$$

$$Y_i = y_n + h \sum_{j=1}^{m} a_{ij} \dot{Y}_i \tag{8.8}$$

$$y_{n+1} = y_n + h \sum_{i=1}^{m} b_i \dot{Y}_i \tag{8.9}$$

where h is the time step. To visualize a particular method, the matrix $\mathbf{A}$ and the vectors $\mathbf{b}$ and $\mathbf{c}$ formed by the a, b and c coefficients are usually put in a tableau as shown in Table 8.1. For backward Euler and Lobatto IIIC, we have the following Butcher tableau given by Table 8.2.

For an RK method satisfying

$$b_i = a_{mi}, \text{ for } i = 1, \ldots, m \tag{8.10}$$

a recording of the response in the variables is obtained as in the two methods above.

$$y_{n+1} = Y_m \tag{8.11}$$

From equations (8.10) and (8.11) an explicit expression may be derived for Y:

$$\dot{Y}_i = \frac{1}{h} \sum_{j=1}^{m} d_{ij} \left(Y_j - y_n \right), \text{ where } d_{ij} = D = A^{-1} \tag{8.12}$$

The equations (8.8)–(8.12) can then be replaced by:

$$F\left(t + c_i h Y_i \frac{1}{h} \sum_{j=1}^{m} d_{ij} \left(Y_j - y_n \right) \right) = 0, \text{ for } i = 1, \ldots, m \tag{8.13}$$

$$y_{n+1} = Y_m \tag{8.14}$$

For Backward Euler this gives:

$$F\left(t_{n+1}, \tilde{y}_{n+1}, \frac{1}{h}\left(\tilde{y}_{n+1} - y_n\right)\right) = 0 \tag{8.15}$$

and for Lobatto IIIC:

$$F\left(t_n, Y_1, \frac{1}{h}\left(Y_1 + Y_2 - 2y_n\right)\right) = 0 \tag{8.16}$$

$$F\left(t_{n+1}, Y_2, \frac{1}{h}\left(-Y_1 + y_2\right)\right) = 0 \tag{8.17}$$

The local error estimate is then:

$$l_{n+1} = \|y_{n+1} - \tilde{y}_{n+1}\| \tag{8.18}$$

To find the solution of the implicit numerical equations (8.15), (8.16) and (8.17), Newton iteration is used as in steady state. Starting values for the iterations are:

$$\tilde{y}^0_{n+1} = y_n + \frac{h_{n+1}}{h_n}\left(y_n - y_{n-1}\right) \tag{8.19}$$

$$Y^0_{1,n+1} = y_n \tag{8.20}$$

$$Y^0_{2,n+1} = \tilde{y}_{n+1} \tag{8.21}$$

The Jacobian, which is needed in the Newton matrices, is generated numerically. This is done by excitation of each variable in turn and recording of the response in the other variables.

8.4 Control element library

The control elements represent basic algebraic or differential functions; time-dependent sample and hold functions; basic linear transfer functions; the most frequently used controllers; and algebraic elements with discontinuities in one of their derivatives, such as the logical switch, limitation, and dead zone.

In the case of the sample and hold element, the user meets no restriction with respect to relations between the sample period and the numerical time step. However, to maintain the order of the method, the control system has to adjust the time step to collect the appropriate sample points.

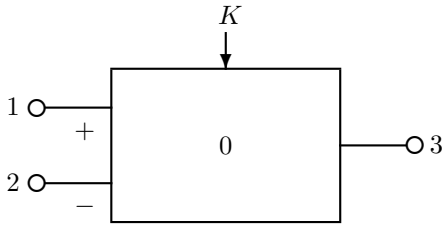
Each sample represents a discontinuous change in the variable. The discontinuity is said to be of order q when it occurs in the q'th derivative of

one of the variables and the lower-order derivatives are continuous. Gear and Østerby [7] have shown that the accuracy of the result will drop below the order p when $q \leq p$, unless we the integration points hit the discrete discontinuity points. With $p = 2$, actions must be taken for discontinuities in the first derivative of the right-hand side of equation (8.3).

The discontinuity points of regular sampling is foreseen easily, however, this is not the case for other element types, such as the logical switch or dead zone. In these cases the discontinuity point must be found by interpolation. For this purpose, an interpolant is used that is optimal for Lobatto IIIC. For development of interpolants for RK-methods refer to [6] for an example.

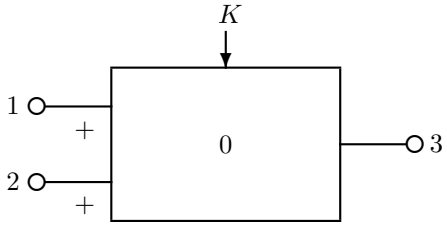
8.4.1 Basic Elements

Comparator **type 1**



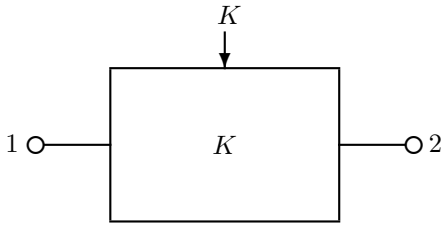
$$y_3 = K(y_1 - y_2)$$

Adder **type 2**



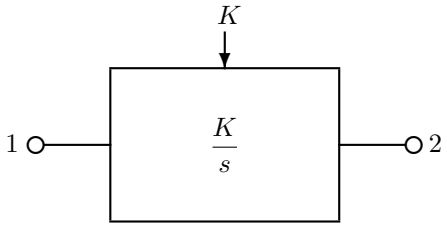
$$y_3 = K(y_1 + y_2)$$

Amplifier **type 3**



$$y_2 = K y_1$$

Integrator **type 4**

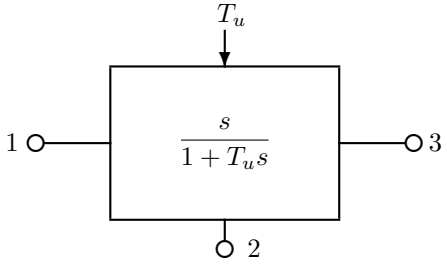


$$\dot{y}_2 = K y_1$$

The resulting output is then

$$y_2 = K \int_0^t y_1 d\tau$$

Limited derivator **type 5**

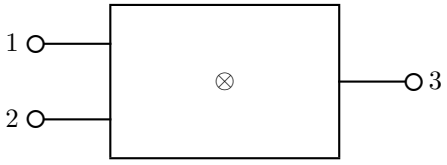


$$\dot{y}_2 = y_3$$

$$y_3 = \frac{1}{T_u}(y_1 - y_2)$$

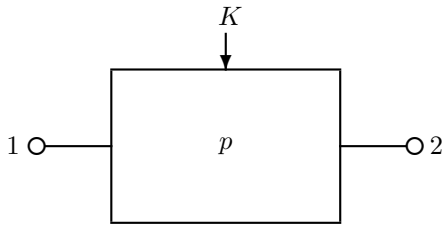
y_2 is an internal variable.

Multiplier **type 6**



$$y_3 = y_1 y_2$$

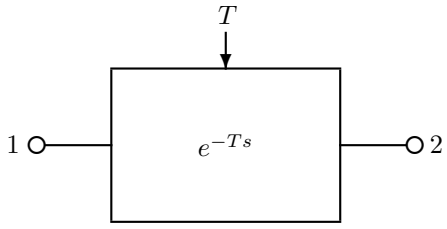
Power **type 7**



$$y_2 = y_1^p$$

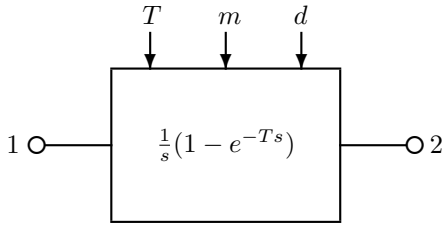
8.4.2 Time dependent elements

Time delay **type 11**



$$y_2(t) = y_1(t - T)$$

Sample and hold **type 12**



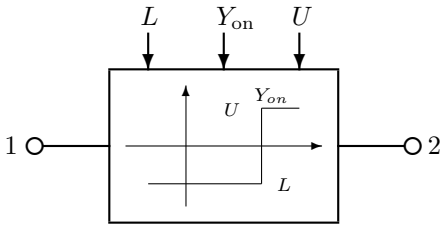
$$y_2(t) = y_1(t^*)$$

$$t^* = (\frac{t - d\,T}{m\,T})m\,T$$

T - basic sample period
m - multiplicity of basic sample period
d - phase delay in basic sample periods

8.4.3 Piecewise Continuous Elements

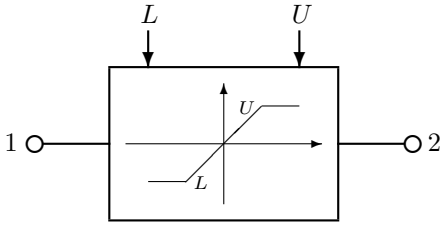
Logic switch **type 21**



$$y_2 = \begin{cases} L & \text{for} \quad y_1 < Y_{\text{on}} \\ U & \text{for} \quad y_1 \geq Y_{\text{on}} \end{cases}$$

L - lower value
U - upper value
Y_{on} - switch coordinate

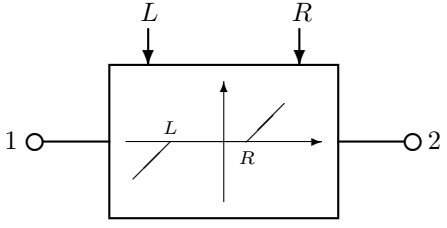
Limitation **type 22**



$$y_2 = \max\{L, \min\{U, y_1\}\}$$

L - lower limit
U - upper limit

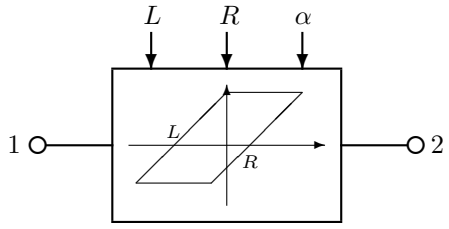
Dead Zone **type 23**



$$y_2 = \begin{cases} y_1 - L & \text{for} \quad y_1 < L \\ 0 & \text{for} \quad L \leq y_1 \leq R \\ y_1 - R & \text{for} \quad y_1 > Y_0 \end{cases}$$

L - left limit
U - right limit

Hysteresis **type 24**

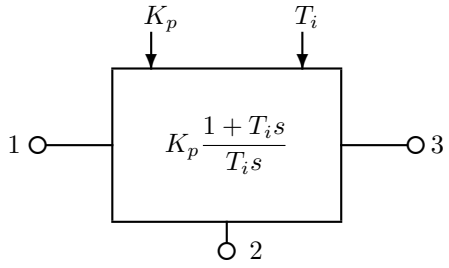


$$y_2 = \begin{cases} \alpha(y_1 - R) & \text{on right slope} \\ \alpha(y_1 - L) & \text{on left slope} \\ y_1 & \text{otherwise} \end{cases}$$

L - left limit
R - right limit

8.4.4 Compensator Elements

PI **type 31**



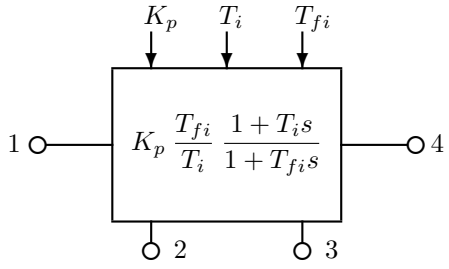
$$\dot{y}_2 = y_1$$
$$y_3 = K_p\left(y_1 + \frac{1}{T_i}y_2\right)$$

where y_2 is an internal variable.

The resulting output is then

$$y_3 = K_p\left(y_1 + \frac{1}{T_i}\int_0^t y_1 d\tau\right)$$

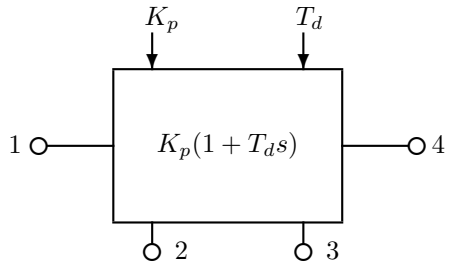
P + lim. I **type 32**



$$\dot{y}_2 = y_1 \qquad T_{fi} > T_i$$
$$\dot{y}_3 = y_4$$
$$y_4 = K_p\left(y_1 + \frac{1}{T_i}y_2\right) - \frac{1}{T_{fi}}y_3$$

y_2 and y_3 are internal variables.

PD **type 33**

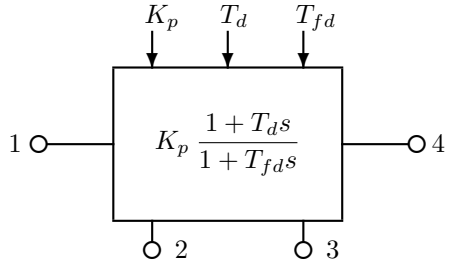


$$\dot{y}_2 = y_1$$
$$\dot{y}_3 = y_4$$
$$y_3 = K_p\,(T_d y_1 + y_2)$$

where y_2 and y_3 are internal variables.

The resulting output is then
$y_4 = K_p\,(y_1 + T_d \dot{y}_1)$

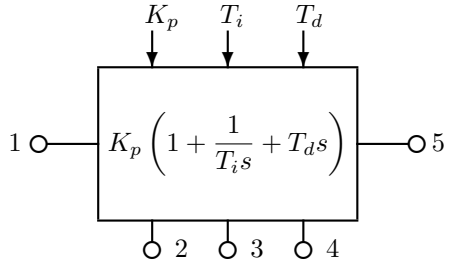
P + lim. D **type 34**



$$\dot{y}_2 = y_1 \qquad T_{fd} < T_d$$
$$\dot{y}_3 = y_4$$
$$y_4 = \frac{1}{T_{fd}}\,(K_p(T_d y_1 + y_2) - y_3)$$

y_2 and y_3 are internal variables.

PID **type 35**

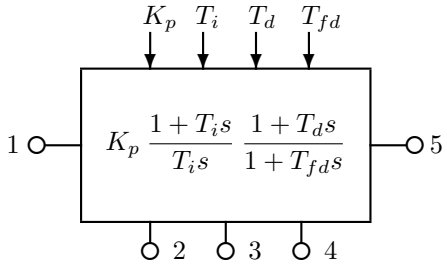


$$\dot{y}_2 = y_1$$
$$\dot{y}_3 = y_2$$
$$\dot{y}_4 = y_5$$
$$y_4 = K_p\left(T_d y_1 + y_2 + \frac{1}{T_i} y_3\right)$$

where y_2, y_3 and y_4 are internal variables.

The resulting output is then
$$y_5 = K_p\left(y_1 + \frac{1}{T_i}\int_0^t y_1 d\tau + T_d \dot{y}_1\right)$$

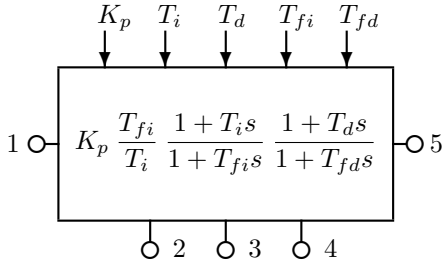
PI + lim. D **type 36**



$$\dot{y}_2 = y_1$$
$$\dot{y}_3 = K_p \left(y_1 + \frac{1}{T_i} y_2 \right)$$
$$\dot{y}_4 = y_5$$
$$y_5 = \frac{1}{T_{fd}} \left(T_d \dot{y}_3 + y_3 - y_4 \right)$$

y_2, y_3 and y_4 are internal variables.

P + lim. I + lim. D **type 37**



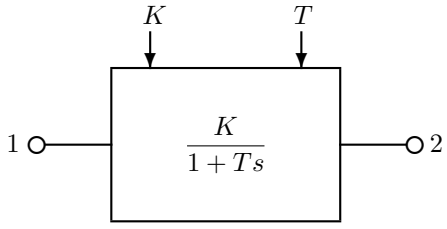
$$\dot{y}_2 = y_1$$
$$\dot{y}_3 = K_p \left(y_1 + \frac{1}{T_i} y_2 \right) - \frac{1}{T_{fi}} y_3$$
$$\dot{y}_4 = y_5$$
$$y_5 = \frac{1}{T_{fd}} \left(T_d \dot{y}_3 + y_3 - y_4 \right)$$

y_2, y_3 and y_4 are internal variables.

8

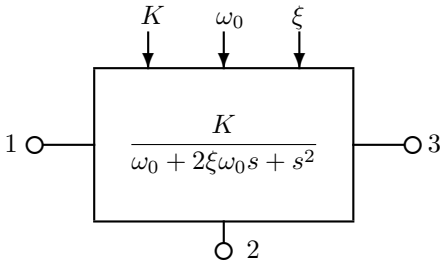
8.4.5 General Transfer Functions

Real pole **type 41**



$$y_2 = \frac{1}{T}(K y_1 - y_2)$$

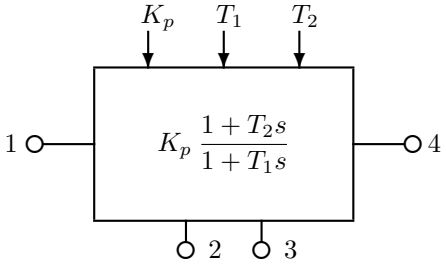
Complex Conjugate Pole **type 42**



$$\dot{y}_2 = K y_1 - 2\xi\omega_0 - \omega_0^2 y_3$$
$$\dot{y}_3 = y_2$$

y_2 is an internal variable.

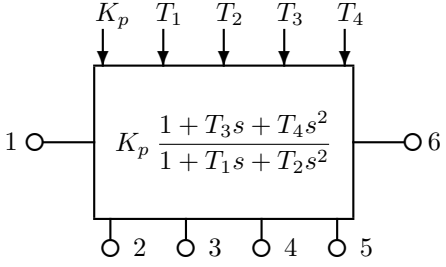
1st Order Element **type 43**



$$\dot{y}_2 = y_1$$
$$\dot{y}_3 = y_4$$
$$\dot{y}_4 = \frac{1}{T_1}\left(K_p\left(T_2 y_1 + y_2\right) - y_3\right)$$

y_2 and y_3 are internal variables.

2nd Order Element **type 44**



$$\dot{y}_2 = y_1$$
$$\dot{y}_3 = y_2$$
$$\dot{y}_4 = y_6$$
$$\dot{y}_5 = y_4$$
$$\dot{y}_6 = \frac{1}{T_2}(K_p(T_4 y_1 + T_3 y_2 + y_3) - T_1 y_4 - y_5)$$

y_2, y_3, y_4 and y_5 are internal variables.

Chapter 9 Simulation Results

9.1 Fatigue analysis

Through fatigue analysis, one can asses the estimated life of a structural component subjected to cyclic or repetitive loads, based on the computed stress or strain history and some additional material properties.

The input to the fatigue analysis is the stress/strain reading in a virtual strain gauge for each time step of the dynamics simulation, or alternatively one can compute the *Signed absolute max* principal strain from the strain rosette tensor. Letting $\{\varepsilon_1, \varepsilon_2\}$ denote the maximum and minimum principal strains[1], respectively, the signed absolute max value is defined through

$$\varepsilon_{\text{samax}} \;=\; \{\varepsilon_i : |\varepsilon_i| = \max\{|\varepsilon_1|, |\varepsilon_2|\}\} \tag{9.1}$$

Similarly, one can derive the signed absolute max principal stress, σ_{samax} from the stress tensor. Using equation (9.1) will normally give a more conservative life assessment compared to using a gauge leg reading, in cases where direction of the maximum strain varies during the simulated event. Using a gauge strain directly does not account for such variations.

9.1.1 Peak valley extraction

The first step in the process of obtaining the estimated life at a given point, is to simplify the stress/strain history curve measured by the virtual strain gauge, and removing oscillations smaller than a given threshold (gate value). This is a process often referred to as *peak valley extraction*.

Consider the typical stress history reading, i.e., $\sigma_{\text{samax}}(t)$, in Figure 9.1a), which typically consists of hundreds (if not thousands) of data points. In a life assessment, it is only the turning points of the curve that matter, i.e., where the gradient of the curve changes sign. Thus, after peak valley extraction using a gate value of 1 MPa, the processed curve ends up as shown in Figure 9.1b). The number of data points has been reduced from 201 to only 22 in this case.

[1] These quantities are the same as the largest and smallest eigenvalues of the symmetric strain tensor.

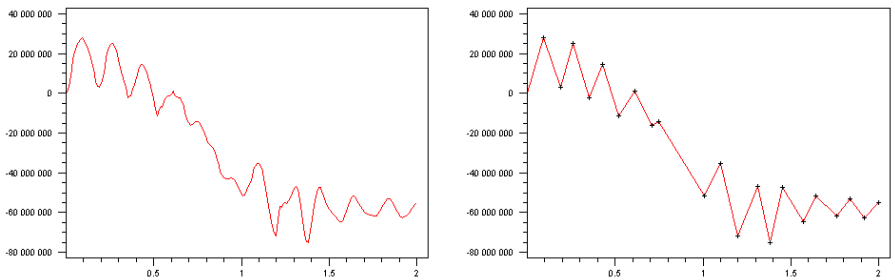
9.1.2 Rainflow analysis

The next step of the fatigue analysis is to perform a *rainflow counting* on the processed stress/strain signal, in order to further reduce a spectrum of varying stresses/strains into a set of simple stress/strain reversals. The process is named rainflow counting because it can be explained by viewing the stress history curve like the one in Figure 9.1b) as a series of Padoga roofs when it is rotated 90 degrees with the time axis vertically, and letting drops of water flow from each peak and valley. The stress cycles are then defined by how each flow of water is terminated according to a set of rules.

In the algorithm adopted in Fedem, we traverse the peak and valley curve in steps looking at three line segments defined by four neighboring points of the curve (labeled 1, 2, 3, 4) in each step: If the middle line 2-3 is shorter than both end lines 1-2 and 3-4, and its gradient has opposite sign as the gradient of both lines 1-2 and 3-4, then the points 2 and 3 define a full stress cycle and is removed from the curve. We then proceed to the next step in which the previous point 4 becomes the new point 2, while the subsequent two points in the curve become the new points 3 and 4. If the points 2 and 3 were not removed, all four point counters are incremented while proceeding to the next step. We then continue until the entire curve has been traversed once.

The traversal is then restarted from the beginning of the curve, and repeated until no more full cycles could be found during one traversal. This process is illustrated in Figure 9.2 for the stress curve of Figure 9.1, where we have encircled the full cycles detected during the first traversal. When these points are removed, the resulting curve becomes as shown in Figure 9.3a).

Even when no more full cycles can be detected through the procedure



a) b)

Figure 9.1: Peak valley extraction of a stress history curve. a) The original stress curve. b) The processed curve consisting of the turning points (+) only.

outlined above, there will normally still be some left-over points in the curve. This is the situation already after the first traversal for the curve in Figure 9.2. The remaining curve is then modified by duplicating the point with the largest magnitude, and then shifting the added point and all preceding points to the end of the curve, as illustrated by the arrow in Figure 9.3a). The modified curve then becomes as shown in Figure 9.3b). Here, the two points indicated by the arrows are not 'turning points'. They are therefore removed from the curve without counting a cycle, resulting in the blue portion of the curve instead. The traversal can now be repeated on the modified curve, until only three points remain which then will count as the final cycle. This process is depicted by the Figures 9.3b-d), where one cycle is detected in each traversal.

The rainflow counting outlined above can be performed in a similar manner for a stress history and a strain history curve. In Fedem it is only applied on stress history results.

9.1.3 Damage and life calculation

The rainflow analysis produces a list of stress ranges with magnitudes $\bar{\sigma}_i$ representing the entire stress history in a given point for the duration of the

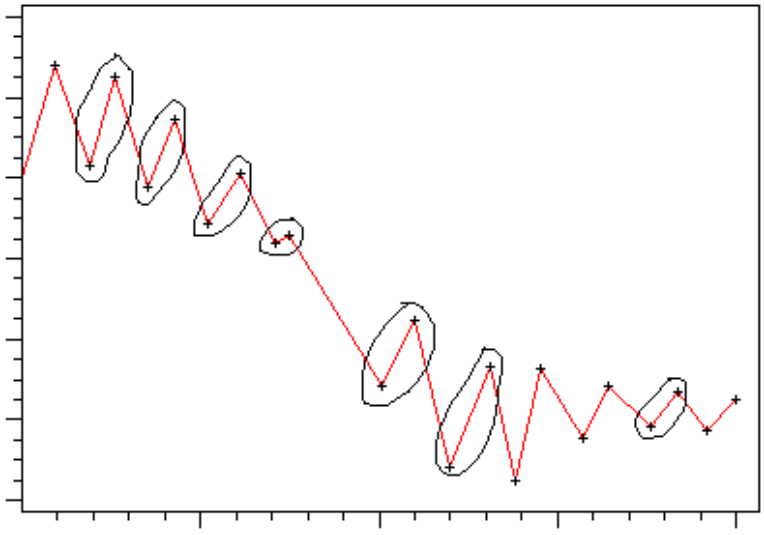


Figure 9.2: Rainflow analysis: Points defining full stress cycles detected during the first traversal of the curve in Figure 9.1b).

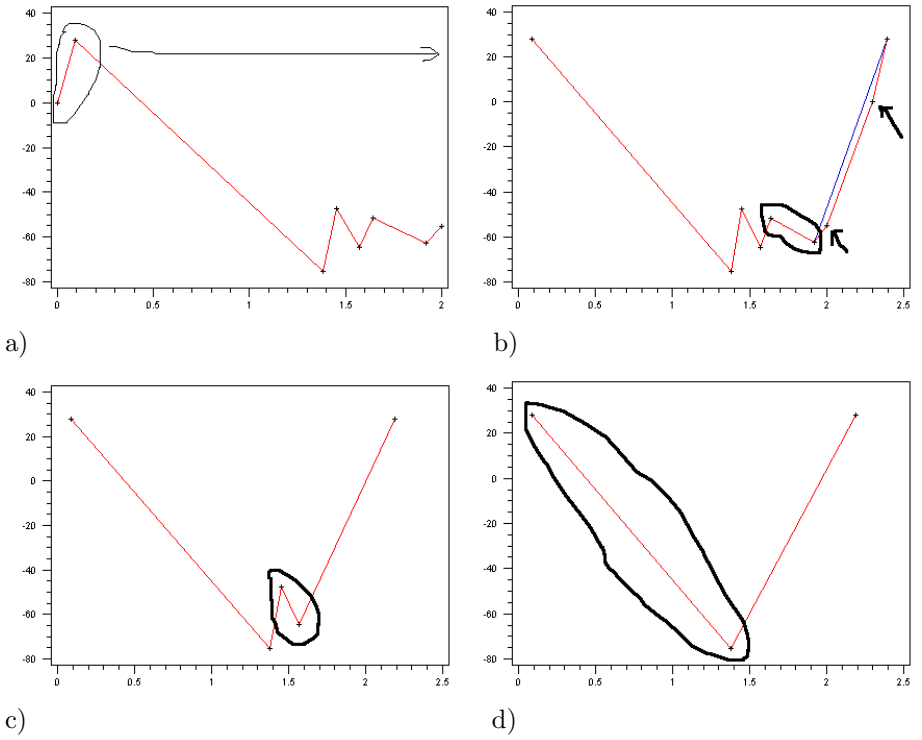


Figure 9.3: Subsequent steps of the rainflow analysis: a) Inserting an additional max point making it the new start and end point. b) Removing two 'non-turning' points and the next full stress cycle. c) Removing another full cycle in the next traversal. d) Counting the final cycle.

numerical simulation. The accumulated damage in that point can now be computed using the S-N curve for the material in question.

An S-N curve relates a stress range magnitude (S) to the number of repetitions (N) of a cycle of that magnitude, a material point can sustain before failure. S-N curves are typically derived from tests on samples of the material in question, and may be found in various design codes for certain materials and loading conditions.

For a given set of stress ranges, $\bar{\sigma}_i, i = 1 \ldots k$, we read the corresponding number of cycles before failure, N_i, from the specified S-N, curve. The

accumulated damage is then computed as

$$C = \sum_{i=1}^{k} \frac{1}{N_i} \tag{9.2}$$

and failure occurs when $C >= 1.0$. The estimated life in terms of number of repetitions of the simulated loading event is then the reciprocal of this value, $1/C$. If the simulated time span is denoted T_s, the estimated life at the material point is therefore

$$\text{Life} = \frac{T_s}{C} \tag{9.3}$$

9.2 Energy calculations

Energy calculations are described by the following terms:

- U_ε Strain energy, computed for all links (superelements), springs, and the system (the system is the sum of all mechanisms)
- U_k Kinetic energy, computed for all links, discrete masses, rotational inertias, and the system
- U_p Potential energy, computed for all links, discrete masses, and the system
- U_i Input energy, computed for all external forces, springs (contribution from non-constant, stress-free length), and the system
- U_d Energy loss, computed for all links (structural damping), axial dampers, joint dampers, joint friction, and the system
- U_e External energy, computed for all external "elements", such as tires, that are using 3rd party software modules. The energy can be one of (or a mix of) strain, damping, and other energy terms.
- U_{sum} Energy check-sum, computed for the system

All these terms are can be plotted as functions of time or other variables.

9.2.1 Strain energy

The total strain energy for the system is computed as a sum of all the element strain energies and spring strain energies.

Link strain energy

The strain energy for one link with linearly elastic material is given by

$$U_\varepsilon = \frac{1}{2}\mathbf{v}_d^T \mathbf{K} \mathbf{v}_d \tag{9.4}$$

where $\mathbf{v}_d$ is the deformational displacement of the element, and $\mathbf{K}$ is the element stiffness matrix. The strain energy is thus computed on a total form at each converged time-step; in other words, no storage of previous strain energy is necessary.

Spring strain energy

Even the nonlinear springs have hyper-elastic behavior, and the strain energy is computed totally at each converged time-step.

$$\begin{aligned} \text{Linear springs :} \qquad & U_\varepsilon = \frac{1}{2} k_s u^2 \\ \text{Nonlinear springs:} \qquad & U_\varepsilon = \int_0^u f_s\left(\hat{u}\right) d\hat{u} \end{aligned} \tag{9.5}$$

As the expression for the hyper-elastic, nonlinear spring energy also includes the linear spring element, it is used for all spring elements, linear and nonlinear, and for axial and joint springs. Incremental calculations of the spring strain energy are not necessary as long as only hyper-elastic nonlinear springs are included in the model.

9.2.2 Kinetic energy

System kinetic energy consists of the contributions from all the links, all the lumped masses, and all the discrete rotational inertias.

Link kinetic energy

The link kinetic energy is calculated from the superelement mass matrix, $\mathbf{M}$ and the superelement velocity vector, $\dot{\mathbf{v}}$, as:

$$U_k = \frac{1}{2}\dot{\mathbf{v}}^T \mathbf{M} \dot{\mathbf{v}}. \tag{9.6}$$

In other words, it is also computed on a total form for each time-step.

Kinetic energy from discrete masses and inertias

Discrete masses and rotational inertias contribute to the kinetic energy as:

$$U_k = \frac{1}{2} m \dot{\mathbf{u}}^T \dot{\mathbf{u}} + \frac{1}{2} \boldsymbol{\omega}^T \mathbf{I}_\omega \boldsymbol{\omega}, \tag{9.7}$$

where m is the lumped mass and $\mathbf{I}_\omega$ is the rotational inertia matrix for the lumped mass.

9.2.3 Potential energy

As is the case with the kinetic energy, the system's potential energy is the sum off all the contributions from the links and the discrete masses. However, the discrete rotational inertias do not contribute to the potential energy.

Link potential energy

Computation of the potential energy should reflect the changes in potential energy rather than the potential energy in relation to the coordinate system used for modeling. Choosing a coordinate system which gives large potential energies may hide other energy contributions. By subtracting the potential energy of the masses in their initial position C_0 from the potential energy at the present position C_n, the initial

potential energy of all masses, superelement masses, and lumped masses is zero.

$$U_p = m\mathbf{g}^T \left(\mathbf{x}_{C_n} - \mathbf{x}_{C_0} \right) \tag{9.8}$$

Calculations of the potential energy of the links use the displacements of the link centroid. This calculation neglects the relative displacement of the centroid due to internal deformations of the superelements; however this is justified by assuming small deformations. The calculation of the potential energy is performed entirely without storing energies from previous steps.

9.2.4 Input energy

Input energy to the system consists of contributions from external forces and springs that have non-constant, stress-free length.

Input energy from external forces

Since the external forces are non-conservative (i.e., can have a time variation and co-rotational behavior), the input energy from the external forces must be computed in an incremental manner from one time-step to the next:

$$U_i = \int_0^t \mathbf{F}^T\left(\hat{t}\right) \dot{\mathbf{u}}\, d\hat{t} \approx \sum_{k=1}^{n} \overline{\mathbf{F}}_k^T \Delta \mathbf{u}_k \quad \text{where} \quad \overline{\mathbf{F}}_k = \frac{1}{2}\left(\mathbf{F}_{k-1} + \mathbf{F}_k\right) \tag{9.9}$$

Subscript k is the time-step index. The summation expression above represents a trapezoidal integration scheme.

Input energy from springs

The stress-free lengths of the springs (both axial springs and joint springs) are subject to possible change. When the spring is stressed, this represents an input energy contribution. This input energy must be computed incrementally for all the springs:

$$U_i = \int_0^t f_s(\hat{t}) l_0\, d\hat{t} \approx \sum_{k=1}^{n} \overline{f}_{s_k} \Delta l_0 \quad \text{where} \quad \overline{f}_{s_k} = \frac{1}{2}(f_{s_{k-1}} + f_{s_k}) \tag{9.10}$$

9.2.5 Energy loss

Total energy loss consists of the contributions from structural damping in the links and discrete dampers (both axial dampers and joint dampers), and energy loss from joint friction.

Energy loss from link structural damping

The structural damping of the links is composed of mass and stiffness proportional damping ($\mathbf{C} = \alpha_1\mathbf{M} + \alpha_2\mathbf{K}$), which is used in the following damping energy expression for a link

$$U_d = \int_0^t \dot{\mathbf{v}}^T \mathbf{C} \dot{\mathbf{v}} d\hat{t} \approx \sum_{k=1}^{n} h\alpha_1 \bar{\dot{\mathbf{v}}}_k^T \mathbf{M} \bar{\dot{\mathbf{v}}}_k + \sum_{k=1}^{n} h\alpha_2 \dot{\mathbf{v}}_{dk}^T \mathbf{K} \dot{\mathbf{v}}_{dk} \tag{9.11}$$

where $\bar{\dot{\mathbf{v}}}_k^T = \frac{1}{2}\left(\dot{\mathbf{v}}_{k-1} + \dot{\mathbf{v}}_k\right)$, and $\dot{\mathbf{v}}_{dk} = \frac{1}{h}\left(\mathbf{v}_{dk} - \mathbf{v}_{dk-1}\right)$, and h represent the time-step size. Note that the deformational velocities, $\dot{\mathbf{v}}_d$, are used when computing the stiffness proportional damping energy. This is important to avoid false damping energies from rigid-body velocities when using large time-steps.

Energy loss from discrete dampers

Energy loss from discrete dampers (axial dampers and joint dampers) is computed as

$$U_d = \int_0^t f_d \dot{u}\, dt \approx \sum_{k=1}^{n} \overline{f}_{d\,k} \Delta u \tag{9.12}$$

where $\overline{f}_{d\,k} = \frac{1}{2}\left(f_{d\,k-1} + f_{d\,k}\right)$

Energy loss from friction

Friction energy loss is computationally analogous to the damping loss

$$U_d = \int_0^t f_f\, dt \approx \sum_{k=1}^{n} \overline{f}_{f\,k} \Delta u_k \tag{9.13}$$

where $\overline{f}_{f\,k} = \frac{1}{2}\left(f_{f\,k-1} + f_{f\,k}\right)$

9.2.6 External energy

External energy from 3rd party components such as tires is calculated is calculated through the incremental work performed at the component interface from one time-step to the next:

$$U_e = \int_0^t \mathbf{F}^T\left(\hat{t}\right) \dot{\mathbf{u}}\, d\hat{t} \approx \sum_{k=1}^{n} \overline{\mathbf{F}}_k^T \Delta \mathbf{u}_k \quad \text{where} \quad \overline{\mathbf{F}}_k = \frac{1}{2}\left(\mathbf{F}_{k-1} + \mathbf{F}_k\right) \tag{9.14}$$

Subscript k is the time-step index. The summation expression above represents a trapezoidal integration scheme. The forces $\mathbf{F}$ from the component can have contribution from both stiffness and damping forces, and the energy contribution is thus a mixed energy.

9.2.7 Energy check-sum

An energy check-sum is computed for the system to verify that the system energy is preserved:

$$U_{\text{sum}} = U_{p_{\text{system}}} + U_{k\text{system}} + U_{\varepsilon\text{system}} + U_{d\text{system}} - U_{i\text{system}} - U_{e\text{system}} \tag{9.15}$$

The check-sum should remain constant during time integration.

Appendix A Finite Element Library

The finite element library in Fedem contains the element types listed in Table A.1. The BUSH, SPRING and RSPRING elements are mass less and contribute to the stiffness matrix only, whereas the CMASS element contributes to the mass matrix only. The RBAR, RGD and WAVGM elements have neither mass nor stiffness on their own. They are constraint elements, i.e., they introduce linear couplings between the degrees of freedom of the other elements in various manners. All the other elements listed in Table A.1 are standard linearized finite elements with both stiffness and mass.

A brief description of these basic elements is given in the following sections. In addition, the formulation of the Generic part element is given in Section A.16. The latter element is used to represent a link in the Fedem Dynamics Solver, when a proper finite element representation is not available or has not yet been established (see the Fedem 8.0 User Guide, *Section 4.1 "Links"*).

Element name	Element description
FFT3	3-node triangular shell element
FFQ4	4-node quadrilateral shell element
TET4	4-node isoparametric tetrahedron solid element
TET10	10-node isoparametric tetrahedron solid element
WEDG6	6-node isoparametric prismatic solid element
WEDG15	15-node isoparametric prismatic solid element
HEX8	Linear isoparametric hexahedron solid element
HEX20	Quadratic isoparametric hexahedron solid element
BEAM2	2-node linear beam element, also used for spot welds
BUSH	2-node bushing element (generalized spring)
SPRING	2-node translatory spring
RSPRING	2-node rotational spring
CMASS	Single-node concentrated mass element
RBAR	2-node rigid bar
RGD	Multi-node rigid body
WAVGM	Multi-node weighted averaged motion element

Table A.1: Fedem element library.

A.1 FFT3

The 3-node triangular shell element FFT3 is composed of a triangular element for plate bending by Allman [1], and a triangular membrane element with rotational Degree of Freedom (DOF) by Bergan and Felippa [4]. The membrane part of FFT3 has 9 DOFs; 6 corner translations, and 3 corner normal rotations. The element is coordinate invariant and passes the patch test for any geometry.

The element performs significantly better than the constant strain triangle. Because of the presence of the normal rotation DOFs, FFT3 is well suited to modeling general shell structures.

The FFT3 element is referred to in a global Cartesian coordinate system as shown in Figure A.1. The element nodes are numbered 1-2-3, and a local coordinate system (x, y, z) is defined in such a way that the $x - y$ plane coincides with the middle surface of the element. Origin is taken at node 1, and positive x-axis coincides with edge 1-2, while positive y-axis is taken in direction from the x-axis toward node 3. Positive z-axis is defined in such a way that x, y and z form a right-handed coordinate system. The nodal DOFs are u, v, w, r_x, r_y and r_z.

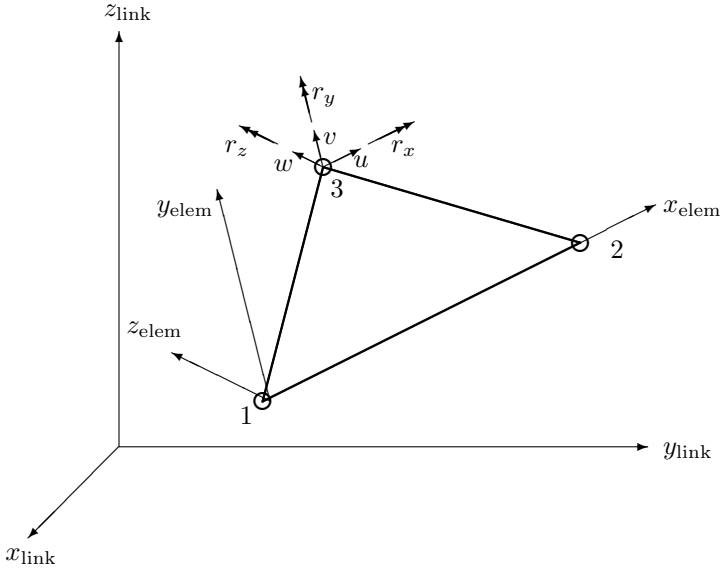


Figure A.1: FFT3, Flat triangular shell element

A.2 FFQ4

The 4-node quadrilateral shell element FFQ4 is composed of a Quadrilateral plate Bending Element with Shear deformation (QBESH) and a Quadrilateral Membrane element with Rotational degrees of Freedom (QMRF). The QMRF element has 12 DOFs; 8 corner translations (u, v), and 4 corner normal rotations (r_z). The element is coordinate invariant and passes the patch test for any geometry. The element performance is significantly better than that of the linear isoparametric quadrilateral. Because of the presence of the normal rotation DOFs, QMRF is well suited to modeling general shell structures. The QBESH element is a quadrilateral plate-bending element that passes the individual element test. The element has 12 DOFs; 3 DOFs (w, r_x, r_y) at each of the 4 nodes. Transverse shear deformation is included in the formulation.

The FFQ4 element is capable of handling warped element geometries by utilizing projection techniques that restore force equilibrium and correct rigid body motion (see [5] and [12]).

The FFQ4 element is referred to in a Cartesian link coordinate system as shown in Figure A.2. The element nodes are numbered counter-clockwise 1-2-3-4. The nodal DOFs are u, v, w, r_x, r_y, and r_z.

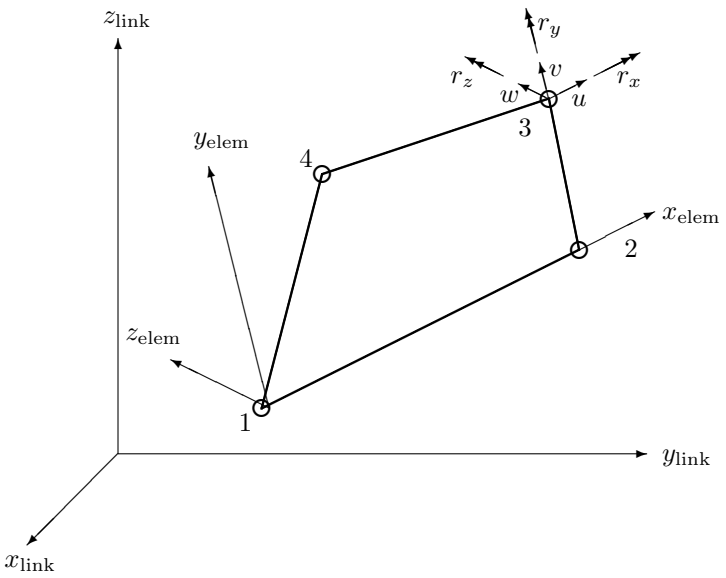


Figure A.2: FFQ4, Flat quadrilateral shell element

A.3 TET4

TET4 is a solid constant strain tetrahedron element with 4 nodes and 12 DOFs. The nodal points at the corners of the tetrahedron are numbered 1 through 4 as shown in Figure A.3.

A.4 TET10

TET10 is an isoparametric tetrahedron element with 30 DOFs; 3 DOFs (u, v, w) at each of the 10 nodes (4 corner nodes and 6 mid-edge nodes). The edges may be straight or curved. Figure A.4 shows a typical element and its local node numbering. The local node numbering 1 through 10 must be carried out in a right-hand direction with nodes 1, 7 and 10 on the same edge.

A.5 WEDG6

WEDG6 is an isoparametric triangular prism element with 18 DOFs; 3 DOFs (u, v, w) at each of the 6 corner nodes. The edges are straight, and Figure A.5 shows a typical element with local node numbering. The local node numbering 1 through 6 must be carried out in a right-hand direction, with nodes 1 and 4 on the same edge. The WEDG6 element has constant strains for triangular cross-sections, and a limited linear variation of the strains in the direction transverse to the triangular side.

A.6 WEDG15

WEDG15 is an isoparametric triangular prism element with 45 DOFs; 3 DOFs (u, v, w) at each of the 15 nodes (6 corner and 9 mid-edge nodes). The edges may be straight or curved. Figure A.6 shows a typical element with local node numbering. The local node numbering 1 through 15 must be carried out in a right-hand direction with nodes 1, 7 and 10 on the same edge. WEDG15 has stresses with minimum linear variation; in addition, it has an incomplete quadratic variation of stresses in the direction transverse to the triangular side. Some stress components also have a quadratic variation in the directions parallel to the triangular side, for example, the transverse stress component.

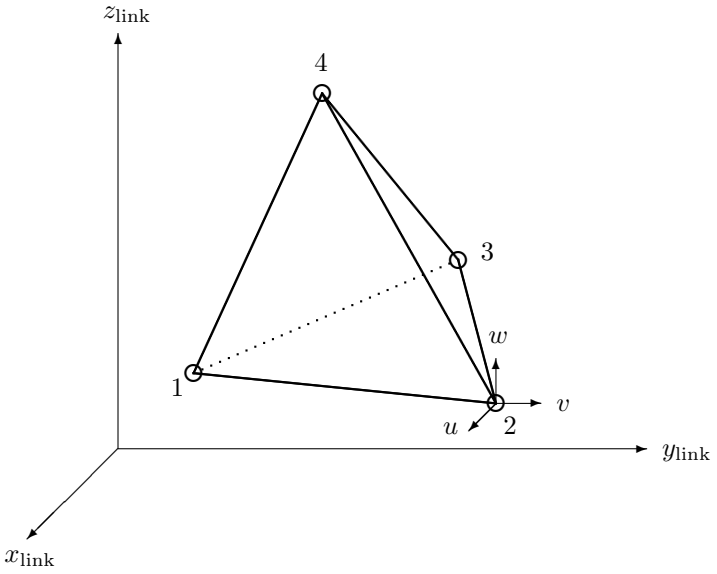


Figure A.3: TET4, Constant strain tetrahedron element

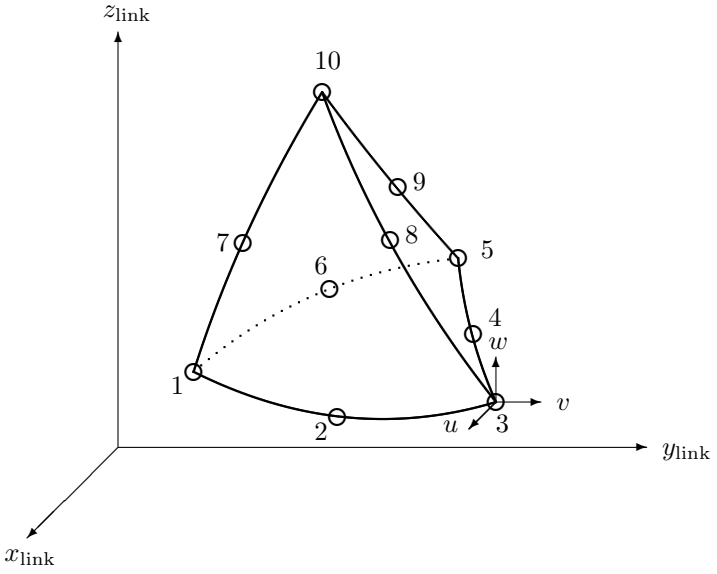


Figure A.4: TET10, Isoparametric tetrahedron element

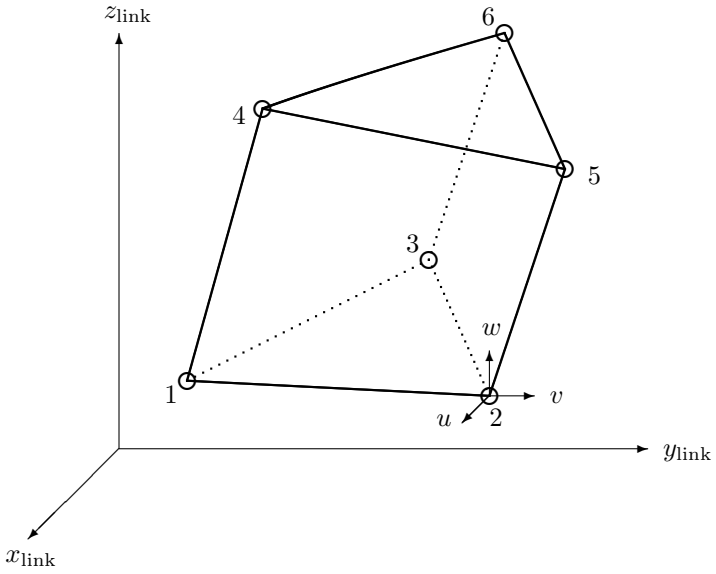


Figure A.5: WEDG6, Isoparametric triangular prismatic element

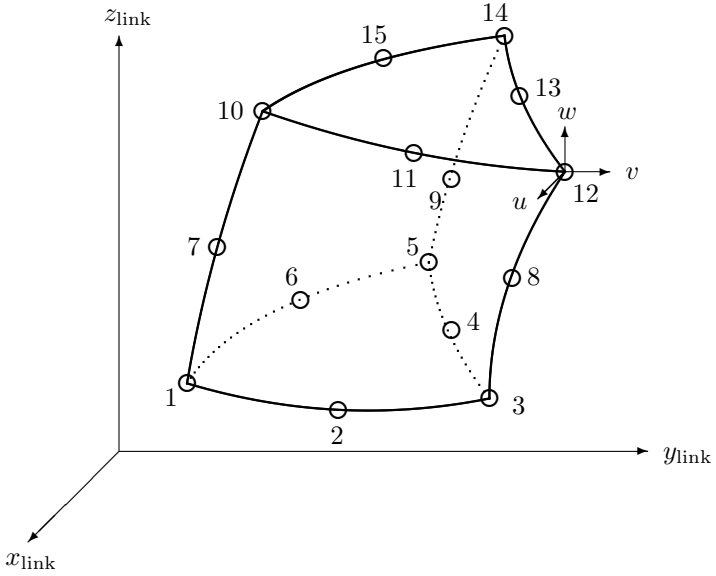


Figure A.6: WEDG15, Isoparametric prismatic element

A.7 HEX8

HEX8 is a solid hexahedron element with 8 nodal points and 24 DOFs. The nodal points located at the corners of the hexahedron are numbered 1 through 8 as shown in Figure A.7. HEX8 is an isoparametric element with linear displacement shape functions.

A.8 HEX20

HEX20 is an isoparametric hexahedron element with 60 DOFs; 3 DOFs (u, v, w) at each of the 20 nodes (8 corner and 12 mid-edge nodes). The edges may be straight or curved, and Figure A.8 shows a typical element with local node numbering. The local node numbering 1 through 20 must be carried out in a right-hand direction with nodes 1, 9 and 13 on the same edge. HEX20 yields an incomplete quadratic variation of the displacements. The minimum variation of stresses along a border is linear for this element.

A.9 BEAM2

The BEAM2 element is based on Euler-Bernoulli's beam theory with quadratic shape functions and continuous first derivatives. The deformations account for are bending, shear, axial and St. Venant torsion. The element is straight with uniform cross section and material properties. The cross section does not need to be symmetrical. The 2 nodal points of the element, one at each end, may be offset in relation to the principle axis.

Figure A.9 shows an arbitrary beam element referred to in a link coordinate system (X, Y, Z) and a local system (x, y, z). The local x-axis coincides with the beam axis through the center of gravity of the cross sections, and is positive in the direction from point I to J. The local y- and z-axes coincide with the principal axes of the cross section. An auxiliary point K defines together with the beam axis the local xz-plane. The local z-axis is positive in the direction from the element toward point K. The end points I and J are connected, via fictitious rigid eccentricities, to the nodal points II and JJ, respectively, at which nodal parameters are defined.

The BEAM2 element may optionally be equipped with pin flags in end I and/or J. They are used to remove connections between the associated grid point and selected DOFs of the beam defined in the local element coordinate system. Thus, they work like inserting a local hinge in the beam for the

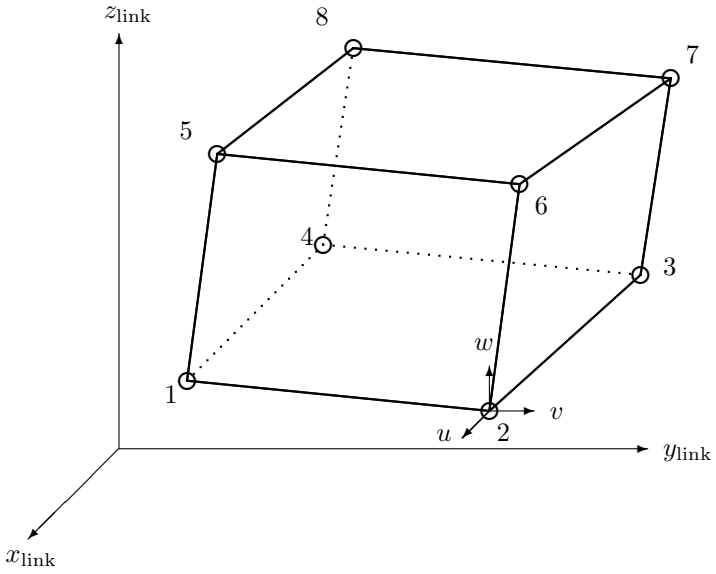


Figure A.7: HEX8, Isoparametric hexahedron element

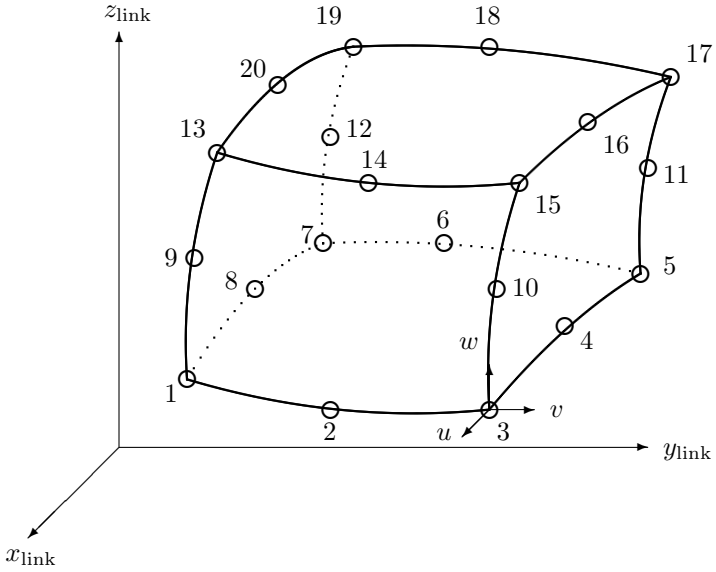


Figure A.8: HEX20, Isoparametric hexahedron element

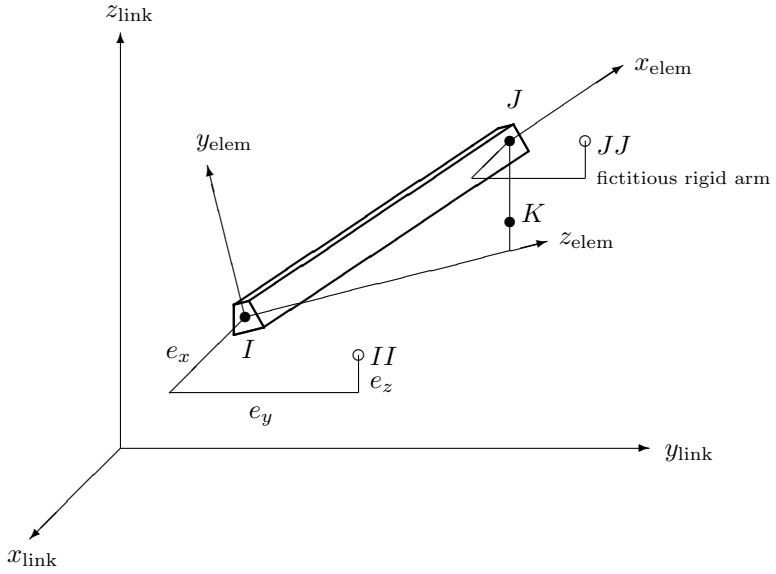


Figure A.9: BEAM2, Beam element

selected DOFs. The beam must have stiffness associated with the DOFs that are released in this manner, e.g. if local DOF 4 is released in one end, the beam must have a nonzero torsional stiffness.

A.9.1 Spot weld element

The BEAM2 element described above is also used to represent spot welds[1] in Fedem. A circular massive cross section is assumed for a spot weld beam. An explicit local z-axis definition is therefore not needed for such elements.

Instead of the rigid arms, a spot weld BEAM2 element is equipped with a WAVGM element in each end (see Section A.15 below), in order to distribute the forces transferred by the beam over a group of nodes in the welded surfaces. The end points of the beam, which are connected to the reference node of the WAVGM elements, are then defined by the projection of a specified point onto each of the 2 welded surfaces. The WAVGM elements may span an arbitrary number of nodes, but typically they are connected to all surface nodes of the finite element that is intersected by the spot weld beam.

[1] Known as CWELD elements in Nastran.

Since the welded surfaces typically are quite close, the actual length of the spot weld beam will be relatively small (or maybe zero). However, it is possible to equip the spot weld beam with a separate effective length property to be used instead of its actual length when calculating the element stiffness.

A.10 BUSH

BUSH is a 2-node generalized spring element of zero length. The two element nodes may, or may not be coincident. The spring element is positioned independently of the two element nodes, and is connected to these nodes via rigid arms, as shown in Figure A.10. Nominal stiffness values for the 3 translational and 3 rotational DOFs are given in a local element coordinate system. This local system is either specified explicitly as an element property, or defined implicitly through the two nodes and a given auxiliary point. The implicit definition is similar to that of the local coordinate system for a beam element (see Section A.9). Thus, this definition is not applicable if the two element nodes are coincident. The BUSH element does not contribute to the mass matrix.

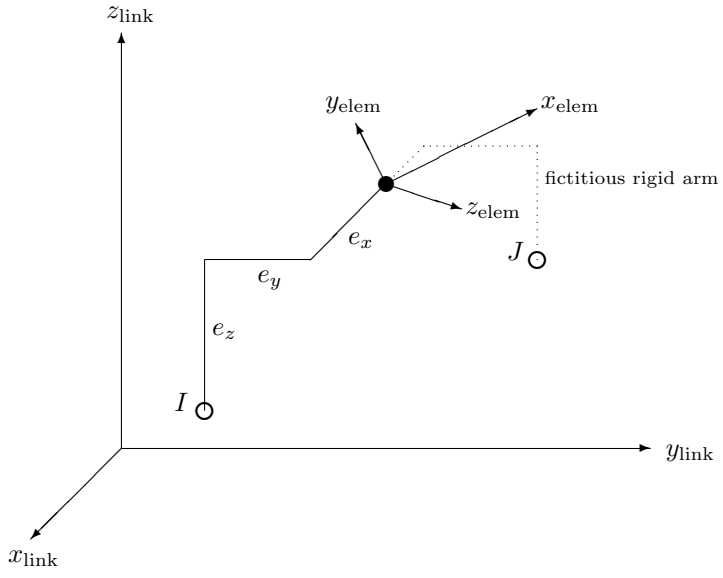


Figure A.10: BUSH, Generalized spring element

The BUSH element may also be specified without any stiffness properties, but only the element topology. Such elements are automatically created during modeling in Fedem, for instance when a mechanism joint is attached to a link at a slave node of an RGD, RBAR or WAVGM element (see the Fedem 8.0 User Guide, *Section 3.6, "Attaching and detaching elements"*). A property-less BUSH element is the created as the connection between this slave node and an added added external node (triad) at the same location.

The nominal stiffness values for the property-less BUSH element are computed automatically based on the overall stiffness properties of the link[2]. For the translational and rotational stiffnesses, k_t and k_r, respectively, the following three alternative procedures are available:

$$k_t = \frac{0.1}{\varepsilon_{\text{sing}}} \min\{\text{diag}(\mathbf{K}_{\text{tra}})\} \quad , \quad k_r = \frac{0.1}{\varepsilon_{\text{sing}}} \min\{\text{diag}(\mathbf{K}_{\text{rot}})\} \tag{A.1}$$

$$k_t = C_s \frac{\text{tr}(\mathbf{K}_{\text{tra}})}{n_{\text{tra}}} \quad , \quad k_r = C_s \frac{\text{tr}(\mathbf{K}_{\text{rot}})}{n_{\text{rot}}} \tag{A.2}$$

$$k_t = C_s \max\{\text{diag}(\mathbf{K}_{\text{tra}})\} \quad , \quad k_r = C_s \max\{\text{diag}(\mathbf{K}_{\text{rot}})\} \tag{A.3}$$

Here, $\mathbf{K}_{\text{tra}}$ and $\mathbf{K}_{\text{rot}}$ are the translational and rotational parts, respectively, of the fully assembled link stiffness matrix, and n_{tra} and n_{rot} denote the total number of translational and rotational DOFs, respectively. Moreover, $\text{diag}(\cdot)$ denotes the diagonal elements of a given matrix, whereas $\text{tr}(\cdot)$ is the trace operator (sum of diagonal elements). Finally, $\varepsilon_{\text{sing}}$ denotes the *singularity criterion* used by the Fedem Link Reducer when factoring the link stiffness matrix (specified by the user through the Link property panel, see the Fedem 8.0 User Guide, *Section 4.1.5,"Link properties"*), and C_s is a user-defined scaling factor that may be specified through the command-line option `-autoStiffScale` when running the Link Reducer (default is 10^2).

The wanted procedure is selected through the command-line option `-autoStiffMethod`. The default is to use equation (A.3).

A.11 SPRING and RSPRING

SPRING is a 2-node linear spring element which adds a 6×6 symmetric stiffness matrix to the translatory DOFs of the 2 nodes. The element stiffness matrix is referred to in the global (or link) coordinate system. The element may be of zero length, i.e. the 2 nodes can have identical coordinates.

[2]Unless the link is completely rigid, e.g., it consists of a single RGD element. In that case $k_t = k_r = 2.0 \cdot 10^{11}$ is used instead.

RSPRING is similar to SPRING, but for the rotational DOFs. The SPRING element may be connected to both 3-DOF and 6-DOF nodes[3], whereas the RSPRING element may be connected to 6-DOF nodes only. The SPRING and RSPRING elements do not contribute to the mass matrix.

A.12 CMASS

CMASS is a 1-node concentrated mass element which adds a 6×6 symmetric mass matrix to a 3- or 6-DOF node. The element mass matrix is referred to in the global (or link) coordinate system. Note that any non-zero inertia terms are ignored for CMASS elements that are connected to 3-DOF nodes. The CMASS element does not contribute to the stiffness matrix.

The CMASS element may also be specified without any mass properties, but only the element topology. During modeling in Fedem, such elements are automatically created at the extra node that is added when creating property-less BUSH elements at slave nodes, see Section A.10. The CMASS element is needed at such added nodes to avoid that the assembled mass matrix becomes singular, with subsequent failure in the eigenvalue analysis. For such property-less CMASS elements, a diagonal element matrix is assumed with the following values for the translational and rotational DOFs, respectively:

$$m_t = C_m \max\{\text{diag}(\mathbf{M}_{\text{tra}})\} \tag{A.4}$$

$$m_r = C_m \max\{\text{diag}(\mathbf{M}_{\text{rot}})\} \tag{A.5}$$

Here, $\mathbf{M}_{\text{tra}}$ and $\mathbf{M}_{\text{rot}}$ are the translational and rotational parts, respectively, of the fully assembled mass matrix, and C_m is a user-defined scaling factor that is specified through the command-line argument `-autoMassScale` when running the Fedem Link Reducer (default value is 10^{-9}).

In some cases, it may happen that the value m_r defined by equation (A.5), is identically zero. For instance, if the finite element model consists of solid elements only, in addition to at least one WAVGM element with automatically added BUSH and CMASS elements at its slave node, there are no mass contributions to the rotational DOFs in the model. In such cases, a value for m_r is instead derived from the global inertia tensor, $\mathbf{I}$, that may be computed

[3] 3-DOF nodes are internal nodes that are used by solid finite elements only and thus lack rotational DOFs. 6-DOF nodes have 3 translatory- and 3 rotational DOFs, and are connected to at least one shell, beam or RBAR element, or is a master node in a RGD element

from the finite element model, i.e.

$$\mathbf{I} := \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} = \int_{\Omega} \rho \begin{bmatrix} y^2+z^2 & xy & xz \\ & x^2+z^2 & yz \\ \text{symm.} & & x^2+y^2 \end{bmatrix} dV \quad \text{(A.6)}$$

where ρ is the mass density, and

$$m_r = \frac{C_m}{3}\left(I_{xx} + I_{yy} + I_{zz}\right) \quad \text{(A.7)}$$

A.13 RBAR

RBAR is a rigid bar element with 2 nodal points and 6 DOFs assigned to each node. The nodal points located at each end of the bar are numbered 1 and 2 as shown in Figure A.11. The nodal DOFs are referred to in the link coordinate system. The element has no material properties. The 12 DOFs of the element are related to each other through the following set of equations:

$$u_2 = u_1 + e_z\theta_{y1} - e_y\theta_{z1} \quad \text{(A.8)}$$
$$v_2 = v_1 + e_x\theta_{z1} - e_z\theta_{x1} \quad \text{(A.9)}$$
$$w_2 = w_1 + e_y\theta_{x1} - e_x\theta_{y1} \quad \text{(A.10)}$$
$$\theta_{x2} = \theta_{x1} \quad \text{(A.11)}$$
$$\theta_{y2} = \theta_{y1} \quad \text{(A.12)}$$
$$\theta_{z2} = \theta_{z1} \quad \text{(A.13)}$$

where e_x, e_y, e_z are the relative distance between the two nodes in the respective coordinate directions, as depicted in Figure A.11

The physical properties of the RBAR element are two sets of component numbers at each node, identifying the dependent and independent DOFs at the node. The total number of independent DOFs in the element must be equal to 6 and they must jointly be capable of representing any general rigid body motion of the element. The element may have up to 6 dependent DOFs. If no dependent DOFs are specified, all DOFs that are not specified as independent will be made dependent.

Since the RBAR element may have dependent DOFs at both nodes, none of the nodes can be a triad (external node) in Fedem. If all independent DOFs are gathered at one node (and all the dependent DOFs are at the other node), the RBAR element is equivalent to a 2-node RGD element, see Section A.14. During an analysis, rigid bar elements are processed using a

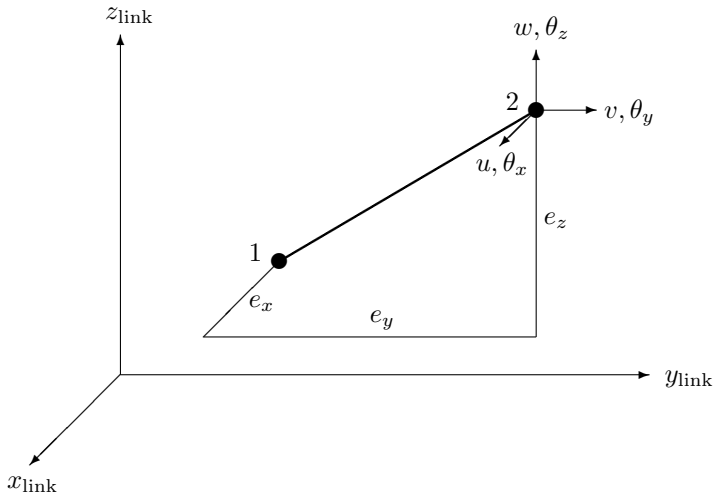


Figure A.11: RBAR, Rigid bar element

DOF elimination method. The constraint equations (A.8)–(A.13) are generated for each element and are used to eliminate the dependent DOFs from the global system of equations, before that system is assembled.

A.14 RGD

RGD is a rigid element with one master node having 6 independent DOFs, 3 translations and 3 rotations. All remaining nodes are slave nodes having either 3 or 6 dependent DOFs. Only the master node can be a triad (external node) in Fedem. The nodal points are numbered 1 through number of nodes as shown in Figure A.12. A RGD element has no material properties. The set of dependent DOFs at the slave nodes may optionally be specified as a physical property. The default behavior if no dependent DOFs are specified is that all DOFs at the slave nodes are made dependent on the master node DOFs.

The presence of a rigid element in a model implies that the motion of all the slave nodes on the element are to be constrained as though they were connected to the master node by mass less rigid beams (or semi-rigid if not all slave node DOFs are made dependent). During an analysis, rigid elements are processed using a DOF elimination method. A set of constraint equations, equivalent to Equations (A.8)–(A.13) is generated for each slave node, relating

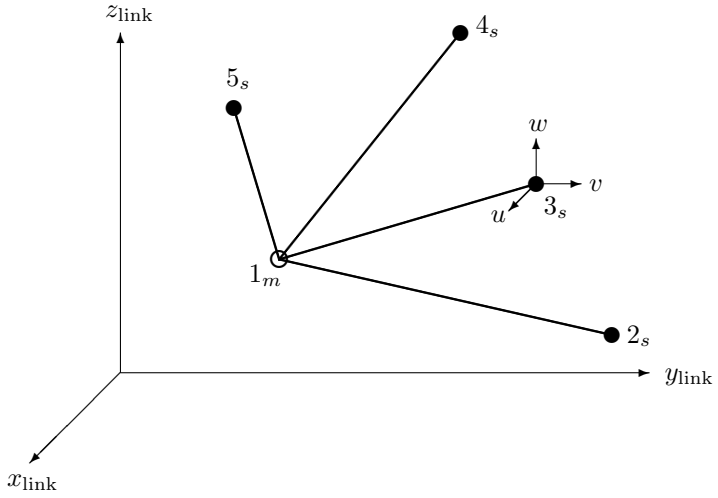


Figure A.12: RGD, Multi-node rigid element

the dependent DOFs to the independent DOFs of the master node. These equations are then used to eliminate all dependent DOFs from the global system of equations, prior to the system matrix assembly. Note that a master node in one RGD element may be a slave node in another RGD. Such chains of RGD elements are resolved explicitly in such a way that all slave DOFs ultimately are coupled only to independent DOFs that do not depend on other RGD constraints.

A.15 WAVGM

WAVGM is an interpolation constraint element[4], which defines the motion at a reference (slave) node as the weighted average of the motions at a set of other (master) nodes. The element topology is similar to that of the RGD-element, see Figure A.13, except that for WAVGM node 1 is a slave node containing the dependent DOFs, whereas all other nodes are masters with independent DOFs.

Unlike the RBAR and RGD elements described in the previous sections, the WAVGM element does not add stiffness to the link, unless the slave node already is connected to some of the master nodes via other finite elements.

[4]This element is known as RBE3 in Nastran.

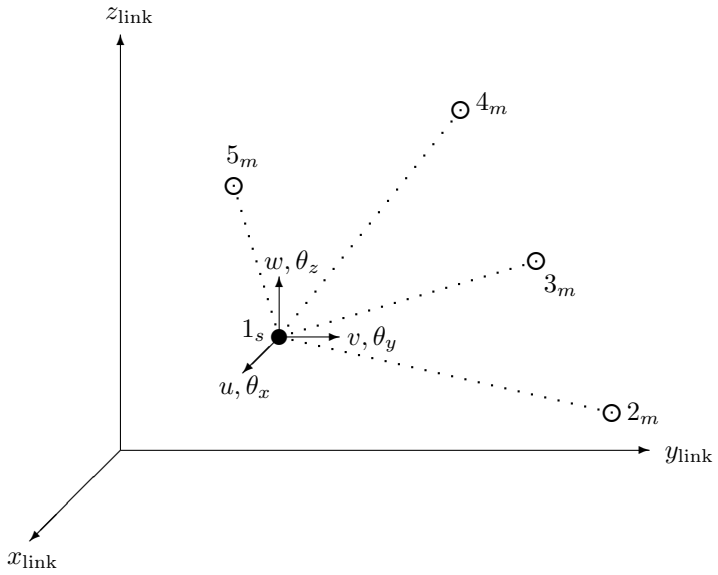


Figure A.13: WAVGM, Multi-node weighted averaged motion element

Thus, the WAVGM element works like a force distributor; forces that are applied at the reference (slave) node are distributed over the master nodes depending on the given weighting factors and the relative distance to the reference node.

The manner in which the forces are distributed is analogous to the classical bolt pattern analysis. Consider a given force $\boldsymbol{F}$ and a moment$\boldsymbol{M}$ applied at the reference node of the WAVGM element. They are first replaced by an equivalent force $\widetilde{\boldsymbol{F}} = \boldsymbol{F}$ and moment $\widetilde{\boldsymbol{M}} = \boldsymbol{M} + \boldsymbol{F} \times \boldsymbol{e}$ at the weighted center of gravity of the master nodes, where $\boldsymbol{e}$ is the offset vector between the reference node and the weighted center of gravity. The force $\widetilde{\boldsymbol{F}}$ is then distributed to the master nodes proportional to the given weighting factors, whereas the moment $\widetilde{\boldsymbol{M}}$ is distributed as forces proportional to their distance from the weighted center of gravity times their weighting factors. Alternatively, the moment $\widetilde{\boldsymbol{M}}$ may be distributed directly as moments at the master nodes (provided they are 6-DOF nodes) proportional to their weighting factors, in the same manner as the force $\widetilde{\boldsymbol{F}}$. This can be used if all master nodes of the element are co-linear, such that they cannot absorb a moment though a set of forces.

The weighting factors can in principle be different for the various force

component at a given master node. Thus, the force distribution is carried out on a component-by-component basis. The resulting expression for the i'th force component at master node number a is then

$$f_i^a \;=\; \frac{F_i\,\omega_i^a}{\sum_b \omega_i^b} \;+\; \frac{(M_j - F_k e_i + F_i e_k)\,\omega_i^a r_{ik}^a}{\sum_b \left({r_{ik}^b}^2 + {r_{ii}^b}^2 \right) \omega_i^b} \;-\; \frac{(M_k - F_i e_j + F_j e_i)\,\omega_i^a r_{ij}^a}{\sum_b \left({r_{ii}^b}^2 + {r_{ij}^b}^2 \right) \omega_i^b} \tag{A.14}$$

where (i, j, k) forms a cyclic permutation of the components x, y, z, and r_{ij}^a denotes the j'th component of the relative position vector $\boldsymbol{r}_i^a$ from the weighted center of gravity to the a'th master node, based on the weighting factors ω_i^a:

$$\boldsymbol{r}_i^a = \boldsymbol{x}^a - \frac{\sum_b \omega_i^b \boldsymbol{x}^b}{\sum_b \omega_i^b} \tag{A.15}$$

Alternatively, when the master nodes are co-linear such that the moment has to be distributed directly, the force components at master node a are given by only the first term of Equation (A.14), whereas the moment components are

$$m_i^a \;=\; \frac{(M_i - F_j e_k + F_k e_j)\,\omega_{3+i}^a}{\sum_b \omega_{3+i}^b} \tag{A.16}$$

The above expressions are now used to establish the governing constraint equations for the WAVGM element, which are used to eliminate the slave node DOFs in the global system of equations of the link FE model. Let $\mathbf{R}_m$ and $\mathbf{R}_s$ denote vectors that collect force components at all master DOFs and slave DOFs, respectively, in the element. Similarly, let $\mathbf{r}_m$ and $\mathbf{r}_s$ denote the associated displacement vectors. The master and slave components are then related through

$$\mathbf{r}_s \;=\; \mathbf{T}_c\,\mathbf{r}_m \tag{A.17}$$

$$\mathbf{R}_m \;=\; \mathbf{T}_c^T\,\mathbf{R}_s \tag{A.18}$$

The row of $\mathbf{T}_c^T$ (column of $\mathbf{T}_c$) corresponding to a given slave DOF is then obtained by in inserting a unit value for $F_x, F_y, F_z,\ M_x, M_y, M_z$, respectively, in turn while letting the other components be zero.

It is clear that for some WAVGM element geometries the denominators of Equation (A.14) might be small, or even zero. For instance, for a three-noded element where all nodes lie on a line that is parallel to the global x-axis, the denominator of the first term is zero when $i = 3$. The size of the denominator is therefore checked against a threshold value, and the resulting constraint coefficient is omitted if the denominator is smaller than this threshold. These

checks are performed as follows for the two terms:

$$\sum_b \left({r_{ik}^b}^2 + {r_{ii}^b}^2 \right) \omega_i^b \quad > \quad \left(\max_b \|r_{ik}^b\| \, \epsilon_{\text{tol}} \right)^2 + \left(\max_b \|r_{ii}^b\| \, \epsilon_{\text{tol}} \right)^2 \quad \text{(A.19)}$$

$$\sum_b \left({r_{ii}^b}^2 + {r_{ij}^b}^2 \right) \omega_i^b \quad > \quad \left(\max_b \|r_{ii}^b\| \, \epsilon_{\text{tol}} \right)^2 + \left(\max_b \|r_{ij}^b\| \, \epsilon_{\text{tol}} \right)^2 \quad \text{(A.20)}$$

where ϵ_{tol} is a relative tolerance parameter that may be set by the user through the command-line option `-tolWAVGM` of the Fedem Link Reducer (default value = 10^{-4}). Thus, constraint coefficients are added only for those terms satisfying the above conditions.

It should be emphasized that the constraints given by Equation (A.17) are enforced in strong form in Fedem (the same is true for the RGD and RBAR elements as well). This implies that a WAVGM slave node can not be a triad (external node) in Fedem. Moreover, WAVGM elements where the slave node already is connected to the master nodes trough other finite elements should be used with caution. Such element may result in an over-constrained system of equations, such that the resulting reduced link does not possess the necessary 6 rigid body modes. This may in turn make the dynamics simulation unstable.

A.16 Generic part element

A generic part consists of a rigid spider attached to a Center of Gravity (CG) node. Each spider leg spans the distance from the CG node to one of the other element nodes. At the end of each leg there is a linear spring (equivalent to the BUSH element described in Section A.10) with some stiffness k_t against translation in all directions, and stiffness k_r against rotation in all three directions. For two co-located nodes (i and j) the overall stiffness of one such spring element is then given by

$$\begin{bmatrix} \mathbf{f}_i \\ \mathbf{m}_i \\ \mathbf{f}_j \\ \mathbf{m}_j \end{bmatrix} = \begin{bmatrix} k_t\mathbf{I} & \mathbf{0} & -k_t\mathbf{I} & \mathbf{0} \\ \mathbf{0} & k_r\mathbf{I} & \mathbf{0} & -k_r\mathbf{I} \\ -k_t\mathbf{I} & \mathbf{0} & k_t\mathbf{I} & \mathbf{0} \\ \mathbf{0} & -k_r\mathbf{I} & \mathbf{0} & k_r\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{v}_i \\ \boldsymbol{\theta}_i \\ \mathbf{v}_j \\ \boldsymbol{\theta}_j \end{bmatrix} \quad \text{(A.21)}$$

With node j being rigidly attached to the CG node, one can establish the

virtual displacement relation

$$\begin{bmatrix} \mathbf{v}_j \\ \boldsymbol{\theta}_j \end{bmatrix} = \begin{bmatrix} \mathbf{1} & -\widehat{\mathbf{e}} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{v}_{CG} \\ \boldsymbol{\theta}_{CG} \end{bmatrix} \quad \text{where} \quad \mathbf{e} = \begin{bmatrix} x_j - x_{CG} \\ y_j - y_{CG} \\ z_j - z_{CG} \end{bmatrix} \tag{A.22}$$

Using the kinematic relationship above in a virtual work expression, one can establish the stiffness matrix for the spider leg between element node i and the CG node as

$$\begin{bmatrix} \mathbf{f}_i \\ \mathbf{m}_i \\ \mathbf{f}_{CG} \\ \mathbf{m}_{\mathrm{CG}} \end{bmatrix} = \begin{bmatrix} \mathbf{k}_t & \mathbf{0} & -\mathbf{k}_t & \mathbf{k}_t\hat{\mathbf{e}} \\ \mathbf{0} & \mathbf{k}_r & \mathbf{0} & -\mathbf{k}_r \\ -\mathbf{k}_t & \mathbf{0} & \mathbf{k}_t & -\mathbf{k}_t\hat{\mathbf{e}} \\ \hat{\mathbf{e}}^T\mathbf{k}_t & -\mathbf{k}_r & -\hat{\mathbf{e}}^T\mathbf{k}_t & (\mathbf{k}_r + \hat{\mathbf{e}}^T\mathbf{k}_T\hat{\mathbf{e}}) \end{bmatrix} \begin{bmatrix} \mathbf{v}_i \\ \boldsymbol{\theta}_i \\ \mathbf{v}_{CG} \\ \boldsymbol{\theta}_{\mathrm{CG}} \end{bmatrix} \tag{A.23}$$

where $\mathbf{k}_t = k_t\mathbf{I}$ and $\mathbf{k}_r = k_r\mathbf{I}$.

The full stiffness matrix for the generic part is then formed by assembling the nodal contributions from all such spider legs. Rewriting equation (A.23) with the more compact notation

$$\begin{bmatrix} \mathbf{f}_1^i \\ \mathbf{f}_2^i \end{bmatrix} = \begin{bmatrix} \mathbf{k}_{11}^i & \mathbf{k}_{12}^i \\ \mathbf{k}_{21}^i & \mathbf{k}_{22}^i \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^i \\ \mathbf{v}_2^i \end{bmatrix} \tag{A.24}$$

the assembly of equation (A.23) for all nodes from 1 to n (with n being the CG node) gives the following generic part stiffness matrix

$$\begin{bmatrix} \mathbf{f}_1^1 \\ \vdots \\ \mathbf{f}_1^{n-1} \\ \sum_{j=1}^{n-1} \mathbf{f}_2^j \end{bmatrix} = \begin{bmatrix} \mathbf{k}_{11}^1 & \dots & \mathbf{0} & \mathbf{k}_{12}^1 \\ \vdots & \ddots & & \vdots \\ \mathbf{0} & & \mathbf{k}_{11}^{n-1} & \mathbf{k}_{12}^{n-1} \\ \mathbf{k}_{21}^1 & \dots & \mathbf{k}_{21}^{n-1} & \sum_{j=1}^{n-1} \mathbf{k}_{22}^j \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^1 \\ \vdots \\ \mathbf{v}_1^{n-1} \\ \mathbf{v}_2^n \end{bmatrix} \tag{A.25}$$

The mass properties of the generic part element are assumed concentrated at the CG node, both with respect to translation and rotation. This gives the mass matrix defined in the relationship

$$\begin{bmatrix} \mathbf{f}_1^1 \\ \vdots \\ \mathbf{f}_1^{n-1} \\ \mathbf{f}_2^n \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \ddots & & \vdots \\ \mathbf{0} & & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{m}_{22}^n \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{v}}_1^1 \\ \vdots \\ \ddot{\mathbf{v}}_1^{n-1} \\ \ddot{\mathbf{v}}_2^n \end{bmatrix} \tag{A.26}$$

where

$$\mathbf{m}_{22}^{n} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & I_{xy} & I_{xz} \\ 0 & 0 & 0 & I_{xy} & I_{yy} & I_{yz} \\ 0 & 0 & 0 & I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \tag{A.27}$$

When using the generic part element with the element matrices defined by equations (A.25) and (A.26) in a dynamics simulation, one might get artificial oscillations in the CG node DOFs, depending on the actual magnitude of the characteristic mass and stiffness being used. This problem might be avoided by eliminating the CG node DOFs through static condensation of equations (A.25) and (A.26), before the time integration is started. This is similar to what is being done for the internal DOFs of the finite element parts in the model reduction process, see Sections 3.2.1 and 3.2.3.

Assuming that all $\mathbf{f}_2^j$ in equation (A.25) are always zero (since no distributed loads are associated with a generic part), the last line of equation (A.25) yields

$$\mathbf{v}_2^n = \mathbf{B} \begin{bmatrix} \mathbf{v}_1^1 \\ \vdots \\ \mathbf{v}_1^{n-1} \end{bmatrix} \quad \text{where} \quad \mathbf{B} = \left[\sum_{j=1}^{n-1} \mathbf{k}_{22}^j \right]^{-1} \begin{bmatrix} \mathbf{k}_{21}^1 & \dots & \mathbf{k}_{21}^{n-1} \end{bmatrix} \tag{A.28}$$

Combining equation (A.28) and its associated second-derivative with the first $n-1$ lines of equations (A.25) and (A.26), respectively, while pre-multiplying with $\mathbf{B}^T$ produces the following element matrices for the generic part

$$\mathbf{k} = \begin{bmatrix} \mathbf{k}_{21}^1 & \dots & \mathbf{k}_{21}^{n-1} \end{bmatrix}^T \mathbf{B} + \begin{bmatrix} \mathbf{k}_{11}^1 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{k}_{11}^{n-1} \end{bmatrix} \tag{A.29}$$

$$\mathbf{m} = \mathbf{B}^T i \mathbf{m}_{22}^n \mathbf{B} \tag{A.30}$$

The stiffness coefficients k_t and k_r that is used in equation (A.21) may be specified by the user for each part (see the Fedem 8.0 User Guide, *Section 4.1.5 "Link properties"*). However, it is also possible to let the coefficients be automatically computed, in such a way that the generic part behaves like an "almost" rigid element.

The automatic "rigid" stiffness is computed from the mass properties of

the part and a given target eigenfrequency:

$$k_t = (2\pi f_{\text{rig}})^2 \, m \tag{A.31}$$

$$k_r = (2\pi f_{\text{rig}})^2 \, \frac{1}{3} \sum_{i,j=x,y,z} I_{ij} \tag{A.32}$$

where m and I_{ij} are components of the mass matrix (A.27), and f_{rig} is the desired target eigenfrequency of the part (in [Hz]). This target value may be set by the user through the command-line option `-targetFrequencyRigid` for the Dynamics Solver. The default value is 10000 Hz.

Bibliography

[1] Allman, D. J., "A Simple Cubic Displacement Element for Plate Bending", *Int. Jour. for Num. Meth. in Engrg.*, Vol. 10, pp. 263–281, 1976. A.1

[2] Sivertsen, O. I., "Large Displacement Finite Element Formulations of Elastic Mechanism Dynamics", *Dr. Ing. Thesis*, Norwegian Institute of Technology, 1981.

[3] Bergan, P. G., Larsen, P. K. and Mollestad, E., "Svingning av konstruksjoner", Tapir forlag, 1981.

[4] Bergan, P. G. and Felippa, C. A., "A Triangular Membrane Element with Rotational Degrees of Freedom", *Comput. Methods Appl. Mech. Engrg.*, Vol. 50, pp. 25–69, 1985. A.1

[5] Rankin, C. C. and Nour-Omid, B., "The Use of Projectors to Improve Finite Element performance", *Computers and Structures*, Vol. 30, pp. 257–267, 1988. A.2

[6] Enright, W. H. et. al, "Interpolants for Runge–Kutta Formulas", *ACM Trans. on Math. Softw.*, Vol. 12, No. 3, pp. 193–218, 1986. 8.4

[7] Gear, C. W. and O. Østerby, "Solving Ordinary Differential Equations with Discontinuities", *ACM Trans. on Math. Softw.*, Vol. 10, No. 1, pp. 23–44, 1984. 8.4

[8] Hilber, H. M., Hughes T. J. R. and Taylor R. L., "Improved Numerical Dissipation for Time Integration Algorithms in Structural Dynamics", *Earthquake Engineering and Structural Dynamics*, Vol. 5, pp. 283–292, 1977. 7.4.1

[9] Iversen, T., "Parallel, Modular Integration for Dynamic Simulation of Industrial Processes", SINTEF report STF48 F86015; (in Norwegian), 1986. 8.2

[10] Iversen, T., "Multidisciplinary Simulation. Method, Software Structure and Documentation for the Control Part", SINTEF work note 88–63–K; (in Norwegian), 1988.

[11] Newmark, N. M., "A Method of Computation for Structural Dynamics", *J. Eng. Mech. Div. ACSE*, Vol. 85, EM3, 1959. 8.1

[12] Haugen, B., "Buckling and Stability Problems for Thin-Shell Structures using High-Performance Finite Elements", *Ph. D. Dissertation*, University of Colorado, 1994. A.2

[13] Geradin, M. and Rixen, D., "Mechanical Vibrations, Theory and Applications to Structural Dynamics", Wiley & Sons Ltd., 1997. 7.4.2

[14] Hulbert, G. M., Chung J. J., "A Time Integration Algorithm for Structural Dynamics with Improved Numerical Dissipation: The Generalized-α Method", *J. Appl. Mech.*, Vol. 60, pp. 371–375, 1993. (document), 7.4.3, 7.4.3, 7.3

[15] Brincker, R. and Ventura, C., "Introduction to Operational Modal Analysis", Wiley, 2015. 7.9