A novel hybrid and publicly available model for spur gear vibrations

User Manual

Authors:

Lior Bachar

Omri Matania

`genGearSimulatedData`

Welcome to the dynamic model of spur gear vibrations!

This function is the core function of the dynamic model, and it is what users call upon to generate a simulated vibration signal. The function encompasses a series of eight sequential steps, each serving as a crucial component in the overall process. The following stages represent a concise outline of the function, as depicted qualitatively in **Figure 1**:

1. ***Model parameters configuration***: This initial phase involves the setup of the simulation parameters and operational conditions, combining default values with user inputs.
2. ***Healthy tooth profile generation***: The subsequent step focuses on the generation of healthy tooth profiles and the associated properties of the contact line for each wheel.
3. ***Manufacturing errors***: Here, we introduce manufacturing errors to the teeth involute profile, enhancing the model's realism.
4. ***Tooth faults and defect information***: In this stage, tooth faults are created upon request, followed by the construction of the defect information structure.
5. ***Gear mesh stiffness calculation***: In this phase, the non-linear gear mesh stiffness is computed, a pivotal factor in understanding gear vibration dynamics.
6. ***Euler-Lagrange components***: Moving forward, we delve into the computation of multi-degree-of-freedom Euler-Lagrange components, which play a fundamental role in formulating the non-linear set of equations of motion.
7. ***Numerical solution***: With the groundwork laid, we proceed to numerically solve the Euler-Lagrange equations combining advanced and classic methods.
8. **Building the simulated vibration signal**: The process reaches its conclusion as it assembles the simulated vibration signal.

These eight stages collectively embody the functionality of our dynamic gear vibration model, and each stage will be explained thoroughly to help you acquire a deep understanding of this model.

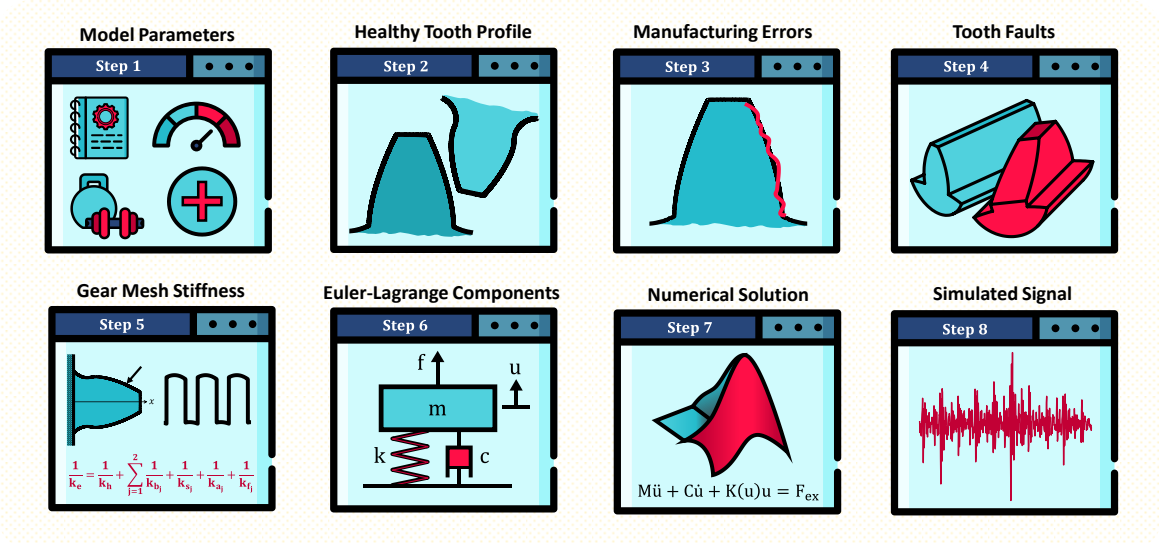

Figure 1 - Building blocks of the dynamic model, arranged by sequential steps.

getParam

This function constructs structures containing all necessary information for generating simulations of gear vibrations. It serves as the only interface between the user and the model. Certain parameters, such as the gear's module, number of teeth, tooth width of both wheels, input speed, and output load, must be provided by the user. Additional input parameters are optional and already set with reasonable default values. By default, the gears are assumed to be in a healthy state. In case of a damaged wheel, the user needs to provide basic information regarding the fault's geometry, following the specific requirements for each fault type.

Users have the flexibility to customize default values using variable-length input argument lists (`varargin`), referencing a dictionary available within this function. In this case, the default values of the desired fields will be overwritten with the specified values. The function iterates over `varargin` and translates the string `key` using the dictionary `dict`. Then it sets the value of the relevant variable.

The function generates eight structures with incomplete fields that will be completed later within the model, including: (1)-(2) wheel information for both wheels (including parameters such as pressure angle, mass, material, involute profile resolutions, etc.), (3) test rig parameters (for example, eccentricity, damping ratio, support bearing stiffness, etc.), (4) simulation parameters (sampling rate, resolution in cycle, numerical solution parameters, etc.), (5) operating conditions (speed and load), (6) initial conditions, and (7)-(8) defect information for both wheels.

Example:

```
genGearSimulatedData( ...
module, z, toothWidth, surfQuality, speedIn, loadOut, 'Alpha', 14.5*pi/180) ;
```

In the following instance, the user inputs a specified pressure angle (α) of 14.5°, thereby replacing the default value of 20° for the pressure angle. As no damage-related properties were provided, the model will generate a simulation of a healthy gear by default.

GetMaterialProp

This function stores material properties, including modulus of elasticity, Poission ratio, and density, for common metals used for gear manufacturing. It gets as an input the material name and returns the stored properties for this metal.

calcMassProp

This function calculates the mass, rotational moment of inertia, and polar moment of inertia of a gear wheel based on its dimensions and material properties, using predefined formulas unless user-defined values are provided.

Firstly, the mass of the gear wheel is determined either by user input or calculated based on an approximation. The commonly used approximation considers the gear wheel as a circular cylinder with the pitch radius R_p and a height equal to the tooth width W, as illustrated in **Figure 2**:

$$m \cong \pi R_p^2 W$$

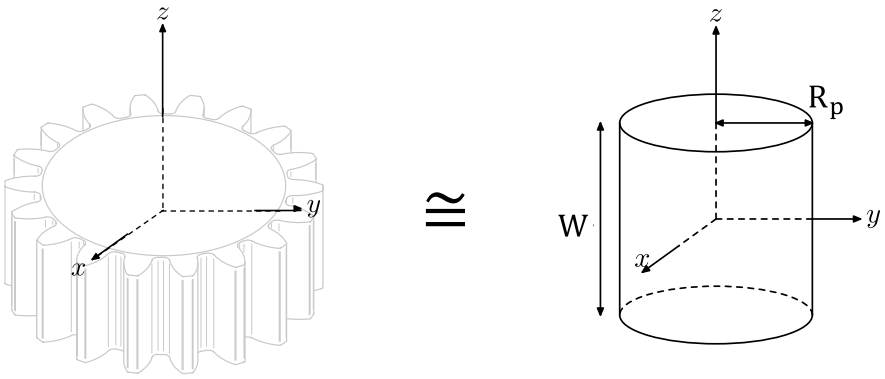

Figure 2 - Approximation of a spur gear as a cylinder with pitch radius illustration

Next, the rotational and polar moments of inertia are either provided by the user or calculated using the mass of the gear wheel. The mass tensor of inertia, which describes how mass is distributed around the rotational axis, is generally expressed as follows:

$$[I] = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

Due to the circular symmetry depicted in **Figure 2**, the tensor of inertia can be simplified based on the coordinates as follows:

$$[I] = \begin{bmatrix} I_{rot} & 0 & 0 \\ 0 & I_{rot} & 0 \\ 0 & 0 & J_{polar} \end{bmatrix}$$

By approximating the gear wheel to a cylinder with the pitch circle, we can obtain the following formulas for the moments:

$$I_{rot} = \int (x^2 + z^2) dm = \frac{1}{12} m\left(3R_p^2 + W^2\right)$$

$$J_{polar} = \int r^2 dm = \frac{1}{2} m R_p^2$$

`calcShaftStiff`

The torsion stiffness of a circular shaft can be calculated using beam theory. The torsion torque is given by the equation:

$$T = \frac{GJ}{L}\theta$$

where T is the torsion torque, G is the shear modulus of the material, J is the polar moment of inertia of the shaft, L is the length of the shaft, and θ is the angular displacement, as shown in **Figure 3**.

Based on Hooke's law, the torsion torque T is directly proportional to the angular displacement θ, with a proportionality constant k. Therefore, we have:

$$T = k\theta$$

By comparing the two equations, we can express the torsional stiffness of the shaft:

$$k = \frac{GJ}{L}$$

The polar moment of inertia J for a cylindrical shaft with a radius r can be calculated as:

$$J = \frac{\pi}{2}r^2$$

Substituting the expression for J into the equation for k, we can simplify and obtain the torsion stiffness formula:

$$k = \frac{\pi r^4 G}{2L}$$

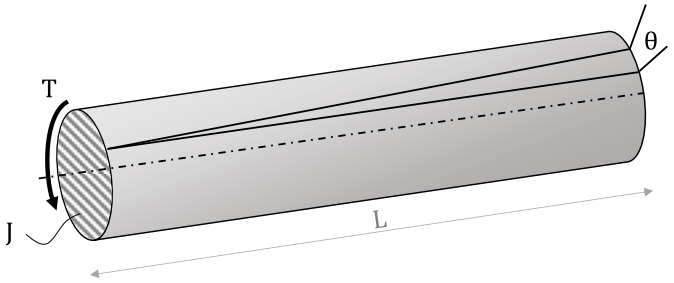

Figure 3 - Illustration of torsional shaft parameters

genHealthyToothProfile

The shape of a spur gear tooth is determined by several key elements, such as the involute profile, addendum, pitch, base, and dedendum circles, along with the fillet curve connecting the flank and dedendum. The involute profile plays a crucial role in achieving a seamless and accurate gear meshing. The purpose of this function is to calculate the Cartesian coordinates of a properly formed tooth in spur gears. The only parameters needed for calculating the tooth profile are the module (m), number of teeth (z), and the nominal pressure angle (α_0). The picture in **Figure 4** provides a visual representation of the common nomenclature used for gears. Taken from Budynas and Nisbett's book *'Shigley's Mechanical Engineering Design'* (2011, p. 676), this illustration showcases the various components and terminology associated with gears. It serves as a helpful reference for understanding terms such as pitch circle, addendum, dedendum, pressure angle, and other essential elements of gear design.

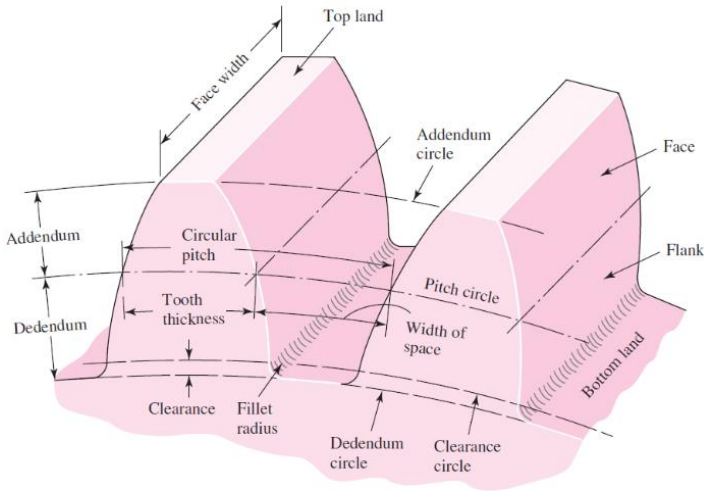

Figure 4 - Nomenclature of spur gear teeth [1]

Characteristic radii:

The geometric properties of a spur gear tooth are determined by characteristic circles, namely the pitch, addendum, dedendum, and base circles. These circles play a significant role in gear design as they serve important purposes and are calculated as follows:

- The pitch circle is crucial in gear design, representing the circle from which the module and diametral pitch are determined. It facilitates the transfer of motion by engaging with the pitch circle of another gear. The radius of the pitch circle can be calculated as follows:

$$\mathrm{R_{pitch}} = 0.5 \times \mathrm{m} \times \mathrm{z}$$

- The dedendum (or root) circle ensures smooth engagement between gear teeth by providing the necessary clearance. It defines the minimum space required to avoid interference during operation. The radius of the dedendum circle can be calculated as follows:

$$\mathrm{R_d} = 0.5 \times \mathrm{m} \times \mathrm{z} - 1.25 \times \mathrm{m}$$

- The base circle is a vital part of gear tooth geometry, acting as a theoretical reference for tooth profile design. It serves as the tangent point for the involute curve, which constructs the tooth shape. The radius of the base circle can be calculated as follows:

$$R_b = 0.5 \times m \times z \times \cos(\alpha_0)$$

- The addendum circle defines the upper part of the gear tooth profile and determines the level of engagement between gears. The addendum is the distance between the addendum circle and the pitch circle. The radius of the addendum circle can be calculated as follows:

$$R_a = 0.5 \times m \times z + m$$

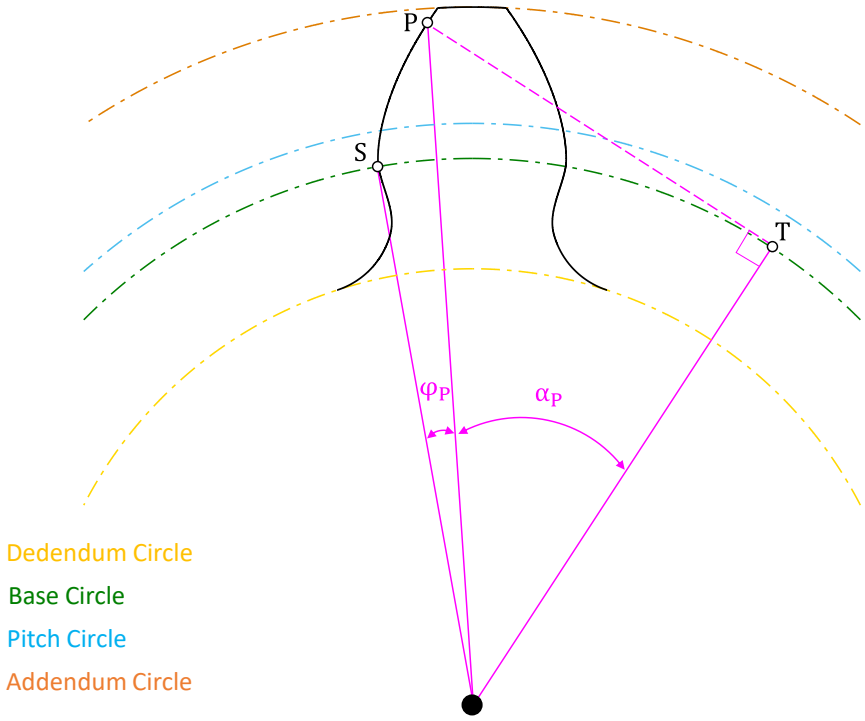

Figure 5 - Illustration of the involute profile properties in spur gears.

The involute function:

The involute function is a mathematical curve that shapes spur gear teeth. It is derived by unwinding a taut string from the gear's base circle, resulting in a smooth tooth profile. This unique curve ensures precise and constant contact between meshing gears, as depicted in **Figure 5**. A notable property of the involute function is that, in the absence of sliding along the base circle, the arclength traced on the base circle is equal to the length of the tangent line from the corresponding point (P) on the involute curve to the base circle.

$$\widehat{ST} = \overline{TP}$$

The arclength along the base circle can be calculated as follows:

$$\widehat{ST} = R_b \cdot (\varphi_P + \alpha_P)$$

The length of the tangent line from point P to the base circle can be calculated as follows:

$$\overline{TP} = R_b \cdot \tan(\alpha_P)$$

By substituting the given relation, we obtain the involute function that calculates the angle φ_P as follows:

$$\varphi_P = \text{inv}(\alpha_p) = \tan(\alpha_P) - \alpha_P$$

The involute profile:

The involute profile of a spur gear tooth begins as a tangent from the base circle and gradually extends outward, curving away from the base circle. This smooth curve continues until it reaches the addendum circle, defining the tip of the tooth and completing the involute profile shape, as illustrated in **Figure 6**. The involute function is employed to calculate the angle θ_P for any point P along the involute profile.

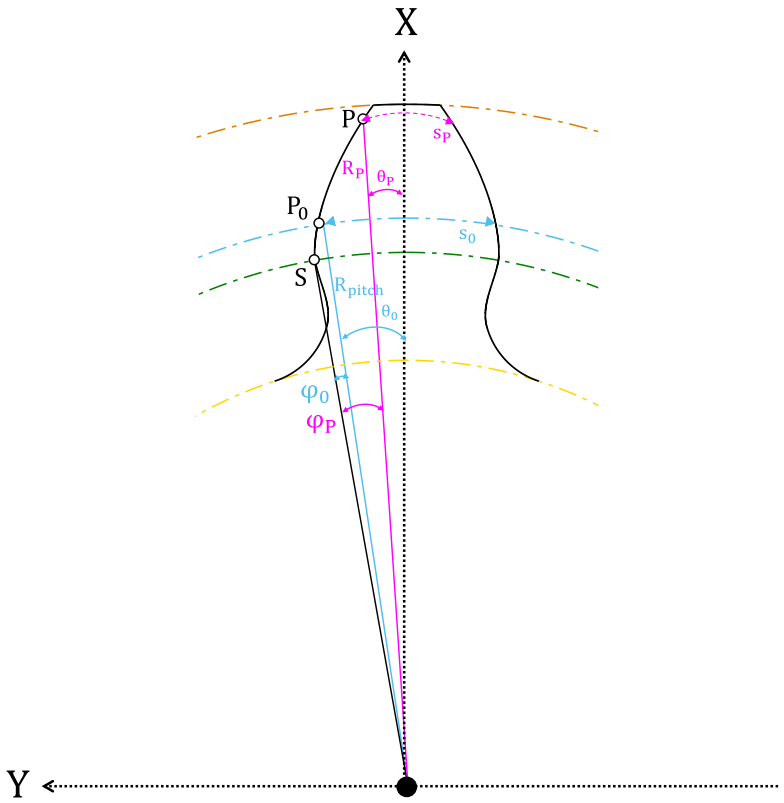

Figure 6 - Illustration of the involute profile calculation

The pitch circle acts as a reference circle that intersects the involute profile at point P_0. At this reference point P_0 along the pitch circle, the tooth thickness s_0 is precisely half of the circular pitch, indicating that:

$$\text{circular pitch} = 2s_0 = \pi m \quad \rightarrow \quad s_0 = \pi m/2$$

Therefore, the angle θ_0 corresponding to the arc on the pitch circle can be calculated as:

$$2\theta_0 = \frac{s_0}{R_{pitch}} = \frac{\pi m/2}{mz/2} \rightarrow \quad \theta_0 = \frac{\pi}{2z}$$

The angle between the point S on the base circle and the X-axis can be calculated either using the angles corresponding to the reference point P_0 or an arbitrary point on the involute profile P, as follows:

$$\theta_P + \varphi_P = \theta_0 + \varphi_0$$

The angle φ is calculated using the involute function, resulting in the following expression for θ_P:

$$\theta_P = \theta_0 + \text{inv}(\alpha_0) - \text{inv}(\alpha_P)$$

Where α_0 represents the nominal pressure angle, and α_P is calculated using the triangle depicted in **Figure 5** as follows:

$$R_b = R_P \cos(\alpha_P) \quad \rightarrow \quad \alpha_P = \cos^{-1}\left(\frac{R_b}{R_P}\right)$$

Given that the involute profile lies between the base and addendum circles, the limits for α_P would be:

$$\alpha_P \in \left[0, \cos^{-1}\left(\frac{R_b}{R_a}\right)\right]$$

If the tooth is rotated at an initial angle θ_i, the angle θ_P on the involute profile of each face can be expressed as follows:

$$\theta_P = \theta_i \pm [\theta_0 + \text{inv}(\alpha_0) - \text{inv}(\alpha_P)]$$

The radius from the origin to a point P on the involute profile is calculated as follows:

$$R_P = \frac{R_b}{\cos(\alpha_P)}$$

Finally, the Cartesian coordinates of each point on the involute profile can be calculated as follows:

$$X_P = R_P \cdot \cos(\theta_P)$$
$$Y_P = R_P \cdot \sin(\theta_P)$$

The tooth tip profile:

The tooth tip profile is formed by an arc traced on the addendum circle, connecting the edges of the involute profiles of each tooth face, as illustrated in **Figure 7**. The Cartesian coordinates of the involute profile are:

$$X_{tip} = R_a \cos(\theta_a)$$
$$Y_{tip} = R_a \sin(\theta_a)$$

Where

$$\theta_a \in [\theta_{a_{top}}, \theta_{a_{bottom}}]$$
$$\theta_{a_{top,bottom}} = \theta_i \pm \left[\theta_0 + \text{inv}(\alpha_0) - \text{inv}\left(\cos^{-1}\left(\frac{R_b}{R_a}\right)\right)\right]$$

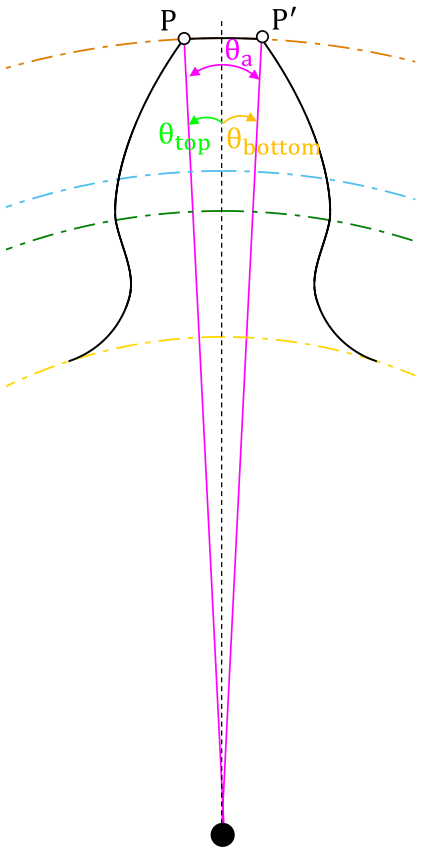

Figure 7 - Illustration of the tooth tip curve parameters

The fillet curve:

The fillet is a curved transition region between the flank and the dedendum of a gear tooth profile, designed to minimize stress concentrations and enhance the tooth's strength and durability. It is crucial to note that when the dedendum circle exceeds the size of the base circle, a fillet will not be present. This situation often occurs in wheels with a high number of teeth, irrespective of their module.

The angle of tooth pattern on a spur gear is evenly distributed along the pitch circle. The pattern begins at the starting point of the fillet curve on the dedendum (root) circle and extends until the completion of the fillet curve on the opposite face of the tooth, as illustrated in **Figure 8**.

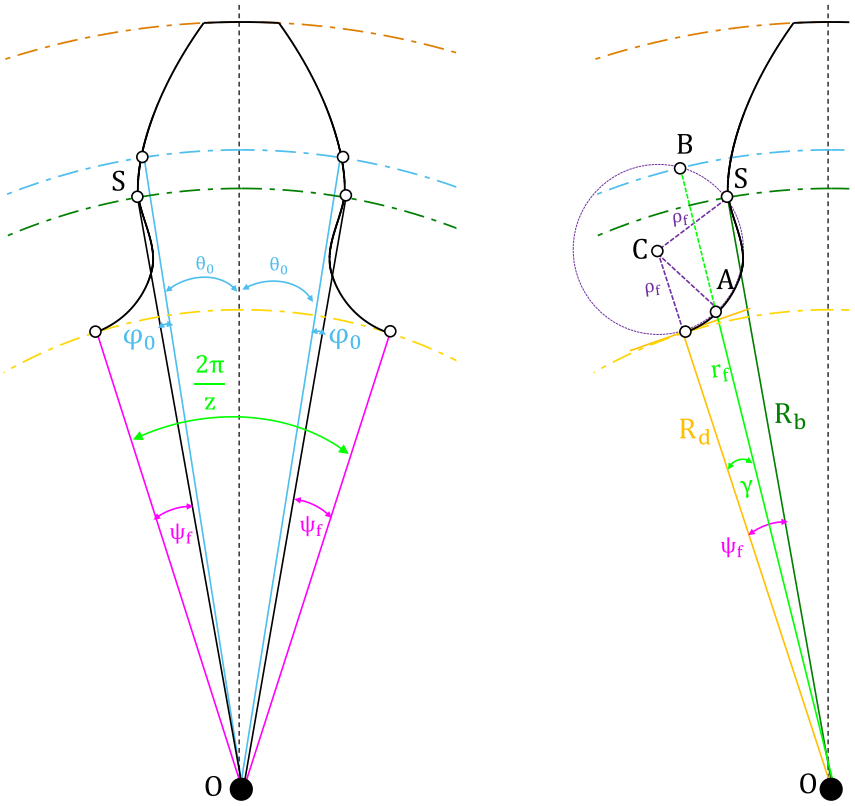

Figure 8 - Illustration of the fillet curve parameters

Considering the presence of z teeth, the total angle of each tooth can be determined as $2\pi/z$. This value, as depicted in **Figure 8**, can also be expressed as:

$$\frac{2\pi}{z} = 2\theta_0 + 2\varphi_0 + 2\psi_f$$

$$\rightarrow \quad \psi_f = \frac{\pi}{z} - \theta_0 - \varphi_0$$

Recalling the calculations mentioned earlier:

$$\theta_0 = \frac{\pi}{2z} \qquad \varphi_0 = \mathrm{inv}(\alpha_0)$$

The angle ψ_f of the fillet curve can be calculated as follows:

$$\psi_{\mathrm{fillet}} = \frac{\pi}{2z} - \mathrm{inv}(\alpha_0) = \theta_0 - \mathrm{inv}(\alpha_0)$$

The fillet curve is characterized by an arc of a circle with a specific radius, denoted as ρ_f. This circular arc smoothly connects the flank and the dedendum of the gear tooth. According to the law of cosines in triangle ΔSCO:

$$\rho_f^2 = (R_d + \rho_f)^2 + R_b^2 - 2R_b(R_d + \rho_f) \cdot \cos(\psi_f)$$

Therefore,

$$\rho_f = \frac{R_d^2 + R_b^2 - 2R_d R_b \cos(\psi_f)}{2R_b \cos(\psi_f) - 2R_d}$$

Each radius r_f from the origin intersects the fillet curve circle at two points, namely A and B. However, only point A is necessary to construct the fillet curve. This can be understood by applying the law of cosines in triangle ΔOCA:

$$AC^2 = AO^2 + CO^2 - 2 \cdot AO \cdot CO \cdot \cos(\gamma)$$
$$\rho_f^2 = r_f^2 + (R_d + \rho_f)^2 - 2 \cdot r_f \cdot (R_d + \rho_f) \cdot \cos(\gamma)$$
$$r_f^2 - [2(R_d + \rho_f)\cos(\gamma)] \cdot r_f + R_d(R_d + 2\rho_f) = 0$$
$$\rightarrow \; r_{f_{A,B}} = (R_d + \rho_f)\cos(\gamma) \pm \sqrt{[(R_d + \rho_f)\cos(\gamma)]^2 - R_d(R_d + 2\rho_f)}$$

Recall that in this case, only point A is necessary, which corresponds to the lower value among the two solutions obtained:

$$r_f = (R_d + \rho_f)\cos(\gamma) - \sqrt{[(R_d + \rho_f)\cos(\gamma)]^2 - R_d(R_d + 2\rho_f)}$$

$$\gamma \in [0, \psi_f]$$

Finally, the Cartesian coordinates of each point on the fillet curve can be calculated as follows:

$$X_f = r_f \cdot \cos(\theta_f)$$
$$Y_f = r_f \cdot \sin(\theta_f)$$

Where

$$\theta_f = \theta_i \pm [\theta_0 + \mathrm{inv}(\alpha_0) + \psi_f - \gamma]$$

genTipRelief

This function serves as a sub-function within `genHealthyToothProfile` and is responsible for creating tip relief in the x-y plane. Tip relief is an indispensable step in high tier gear manufacturing, as it reduces the likelihood of tooth interference and facilitates smoother gear meshing by mitigating stress concentrations at the tooth tip. This feature is widely recommended for gears and is suggested as an optional consideration in the model.

In addition to the `include` flag set in `getParam` to incorporate tip relief, the tip relief curve requires three user-defined parameters: the distances in the x-axis (xRelief) and y-axis (yRelief) between the tooth tip and the tip relief edge, and the desired shape of the tip relief curve (either `linear` or `parabolic`), as illustrated in **Figure 9**. By default, the model generates a parabolic tip relief curve with typical xRelief and yRelief values of 1mm.

The linear tip relief curve is the simplest, formed by extending a line between the relieved edge and the tooth tip based on the tip relief dimensions. However, a linear curve exhibits discontinuity in the slope of the tooth profile which has an impact proper operation. In contrast, the parabolic tip relief curve is the most common choice, ensuring smooth continuity at the relieved tip edge, as illustrated in the comparison in **Figure 9**.

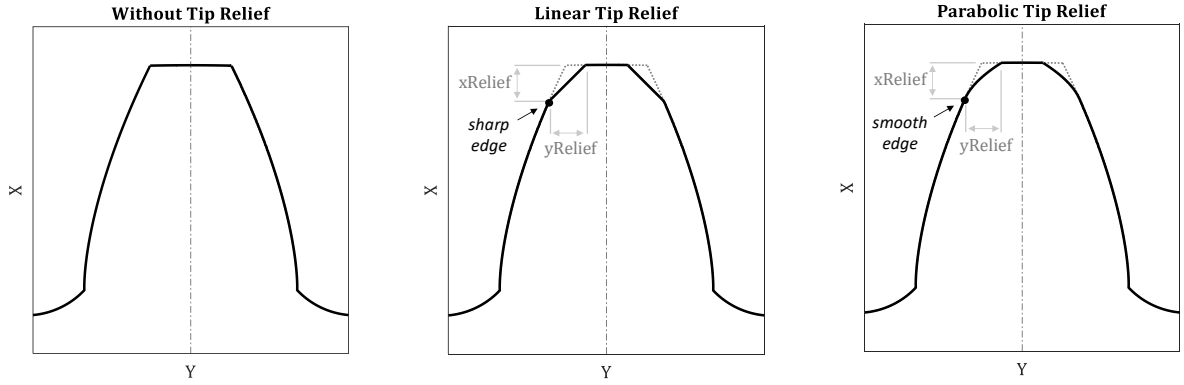

Figure 9 - Comparison of common tip relief curves

`calcContLineProp`

This function performs two key calculations. Firstly, it calculates the contact ratio using a common formula. Secondly, the function employs a geometrical analysis to determine the initial contact point. This point signifies the location where tooth engagement occurs between the gears. By combining these calculations, the function provides valuable insights into the contact properties of the gear system.

Initial contact point

The calculation of the initial contact point assumes that tooth engagement occurs when the addendum of the driven wheel intersects the line of action, while tooth separation occurs when the addendum of the driving wheel intersects the line of action. The radii to the initial contact points of the driving and driven wheels can be determined through a simple geometric analysis, as depicted in **Figure 10**. Along the line of action, five points of interest are marked as follows:

- A, E: The tangent points between the line of action and the base circle of the driving and driven wheels, respectively.
- B, D: The intersection points between the line of action and the addendum circle of the driven and driving wheels, respectively.
- C: The tangent point between the pitch circles.

The initial contact point on the driving wheel corresponds to the radius O_1B, whereas the final contact point on the driven wheel corresponds to the radius O_2D.
By considering the right triangle ΔO_1AB, we can obtain the following relationship:

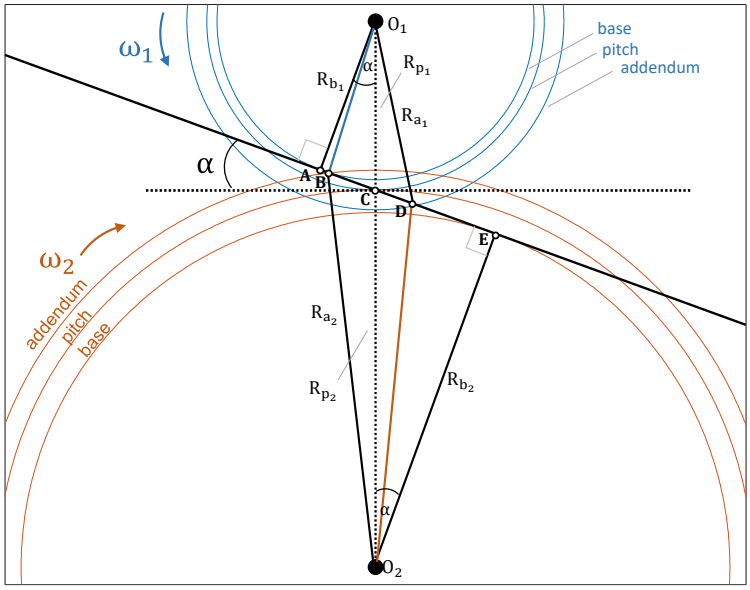

Figure 10 - Illustration for the parameters of the initial contact radius calculation

$$O_1B = \sqrt{AB^2 + R_{b_1}^2}$$

The length of the segment AB can be expressed as follows:

$$AB = AC + CE - BE$$

By considering the right triangles ΔO_1AC and ΔO_2CE, we can establish the following relationships:

$$AC = R_{p_1} \sin(\alpha)$$
$$CE = R_{p_2} \sin(\alpha)$$

By considering the right triangle ΔO_2BE, we can obtain the following relationship:

$$BE = \sqrt{R_{a_2}^2 - R_{b_2}^2}$$

By substituting the given relations, we can calculate the radius to the theoretical initial contact point on the driving wheel as follows:

$$O_1B = \sqrt{\left[\sqrt{R_{a_2}^2 - R_{b_2}^2} - \left(R_{p_1} + R_{p_2}\right)\sin(\alpha)\right]^2 + R_{b_1}^2}$$

Similarly, the radius to the theoretical final contact point on the driven wheel can be calculated as follows:

$$O_2D = \sqrt{\left[\sqrt{R_{a_1}^2 - R_{b_1}^2} - \left(R_{p_1} + R_{p_2}\right)\sin(\alpha)\right]^2 + R_{b_2}^2}$$

Contact ratio

The contact ratio ε is a crucial factor in assessing the quality of gear meshing in spur gears. It represents the number of mesh cycles that occur during tooth engagements. A higher contact ratio indicates smoother contact and quieter transmission. The contact ratio in spur gears is calculated solely based on the number of teeth (z) and pressure angle (α). The calculation is performed using the following formula:

$$\varepsilon = \frac{\sqrt{(z_1 + 2)^2 - (z_1 \cos(\alpha))^2} + \sqrt{(z_2 + 2)^2 - (z_2 \cos(\alpha))^2} - (z_1 + z_2)\sin(\alpha)}{2\pi \cos(\alpha)}$$

From a geometrical perspective, it is worth noting that the numerator of the contact ratio represents the length of the action BD in **Figure 10**, which signifies the distance from tooth engagement to tooth separation. Meanwhile, the denominator represents the base pitch, which is the circumference of the base circle divided by the number of teeth, both normalized by the module. This geometrical interpretation provides insight into the relationship between the contact ratio formula and the specific geometric properties of the gear system.

genSurfQualityErrs

This function is designed to generate teeth profile errors resulting from manufacturing imperfections and surface quality, which are inevitable. The calculation of these errors adheres to the DIN-3962 standard for determining the surface roughness of cylindrical gears []. The profile errors are individually computed for each tooth, starting from the initial contact point to the tip of the tooth. Subsequently, these errors from all teeth are combined into a single vector, representing the profile errors for a complete cycle of the wheel, as illustrated in **Figure 11**. It is important to note that the profile errors vector is piecewise continuous, as there is no inherent expectation of continuity in the transitional regions between teeth.

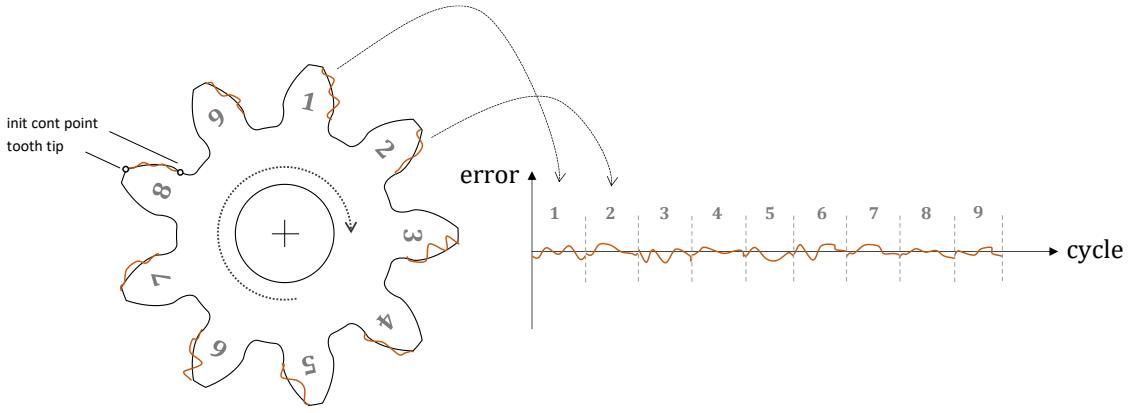

Figure 11 - A graphic illustration for the profile errors vector in cycle

Surface roughness parameters according to DIN-3962 standard

The DIN-3962 standard serves as a fundamental guideline for assessing the quality of gear teeth in various industrial applications, as depicted in **Figure 12**. This illustration provides a qualitative representation showcasing the ideal profile, actual profile, and mean actual profile of the gear teeth. It also includes informative visualizations such as the addendum circle (marking the tooth tip), the initial contact point circle, the theoretical usable involute profile, and the base circle.

Within the realm of spur gears, this standard introduces two crucial surface roughness parameters, both of which are highlighted in **Figure 12**. The first parameter, $f_{H\alpha}$, is known as the profile angle error or profile slope deviation. It is measured as the circumferential distance, typically in micrometers, between the edges of the mean profile. The second parameter, $f_{f\alpha}$, characterizes the profile form deviation. It is calculated as the circumferential distance between the margins of the mean profile.

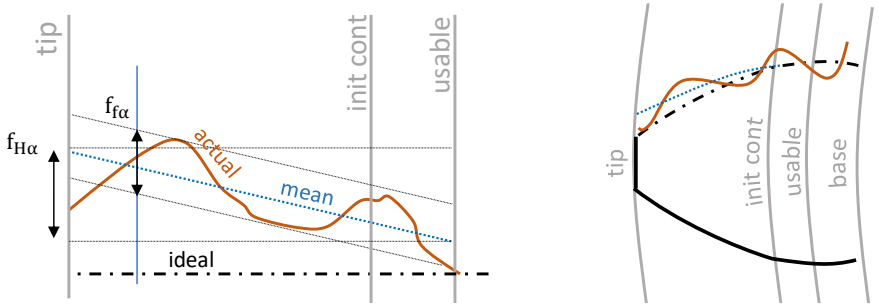

Figure 12 - Relevant parameters of the surface roughness according to DIN-3962 standard

The formula for the profile errors

The function for profile errors should express the deviation of the actual profile from the ideal involute profile along it. It can be challenging to track the position along the profile due to the complexity and curvature of the involute function. However, it is important to note that the involute function has a unique property: the tangent line from the involute to the base circle is equal to the arc length from the tangent point to the tooth profile along the base circle ($\widehat{ST} = \overline{TP}$ in **Figure 12**). Additionally, it's worth recalling that the pressure line in spur gears is tangent to the base circle of both the input and output wheels, as depicted in **Figure 10**. The length of the pressure line represents the distance between the intersection points of the initial contact points of each wheel. It is assumed that the initial contact point on one wheel touches the tooth tip on the addendum circle of the other wheel. In this case, the pressure line can be used to scale the location on the involute profile. The relative location on the pressure line, denoted as $\hat{s}$, ranging from 0 to 1, can be calculated as follows:

$$\hat{s}(cyc) = \frac{s(cyc) - s_{init\_cont}}{s_{tip} - s_{init\_cont}}$$

Here, s_{init_cont} and s_{tip} represent the intersection points with the initial contact point and the tooth tip along the line of action, respectively, and $s(cyc)$ represents any point in between.

The profile errors of each tooth ($i = 1, .., z$) can be expressed using a formula that consists of a deterministic term and a random term, as shown below, based on the work of XXX [].

$$erros_i(cyc) = f_{H\alpha_i}\hat{s} + \frac{1}{2} f_{f\alpha_i} \sin\left(2\pi n_{cyc_i}\hat{s} + \phi_i\right) + noise_i(cyc)$$

Here, the number of cycles n_{cyc_i} is determined using common rules of thumb, and the phase shifting ϕ_i is randomly chosen within the range of [0, π]. Each of the tooth-dependent parameters in the formula ($f_{H\alpha}$, $f_{f\alpha_i}$, n_{cyc_i}, ϕ_i) is determined separately for each tooth by setting a nominal value with random noise and remains constant throughout the cycle. The noise level $noise_i$ is randomly selected for each tooth and for each cycle point.

The final steps involve concatenating the errors of all teeth together and resampling the signal to match the desired resolution. Additionally, it is important to remember that the output wheel enters the pressure line at the tooth tip and ends at the initial contact point, resulting in its profile being flipped.

`genDefectInfoStruct`

This function takes two structures as input for each wheel, totaling four structures: one containing nominal wheel information (wheelInfo and otherWheelInfo) and the other holding information about any existing damages (wheelDefectInfo and otherWheelDefectInfo). The function returns wheelDefectInfo and otherWheelDefectInfo updated defect with vital information.

- It is important to note that, to date, the dynamic model can only handle faults on the output (driven) wheel; attempting to seed a fault on the input wheel will result in an error.
- otherWheelInfo is stored in wheelDefectInfo for future operations.
- If a wheel is damaged, the function generates the damaged profile and other essential parameters. These are stored in both wheelDefectInfo and otherWheelDefectInfo for relevant operations.
- In the case of a `MissingTooth` fault, no damaged profile is generated. However, t otherWheelDefectInfo is still updated.
- If a wheel is healthy, otherWheelDefectInfo (whether damaged or not) remains unchanged, and the function returns. ⚠ This is **crucial** because we must avoid overwriting this structure in such scenario.

genToothBreakageProfile

This function calculates the profile of a broken tooth by utilizing parameters from both a healthy tooth and a defective tooth. It requires input of the wheel profile and another structure that contains the fault dimensions, as shown in **Figure 13**, along with other relevant parameters. The function calculates the start and end indexes of the defect based on the tooth profile and the rotation cycle of the input wheel. Furthermore, the function constructs a linear line based on the fault shape that aids in determining the variations of the tooth width along the X-axis, and the position of the defect center along the Z-axis.

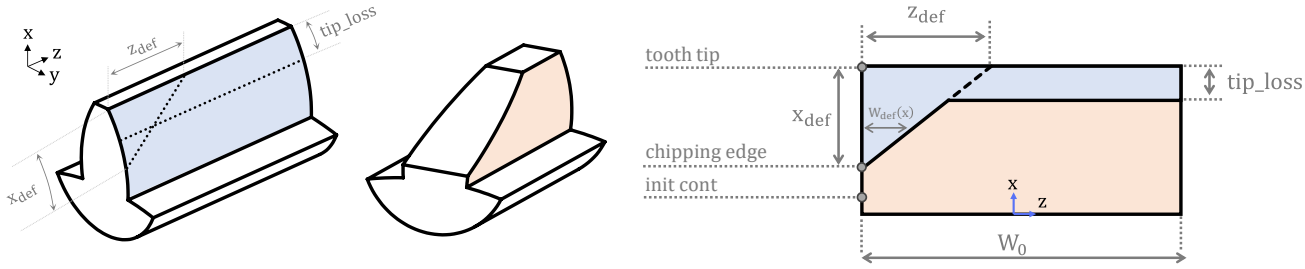

Figure 13 - Illustrative sketches and definitions of the simulated tooth breakage faults

The tooth breakage is characterized by two distinct parameters: chipping, which entails a partial tooth loss starting from the tip, and tip loss, which indicates an overall reduction in tooth height. The chipping angle is defined by the parameters $[x_{def}, z_{def}]$, while the tip_loss parameter directly governs the extent of the tip reduction. The structure of the defect information is augmented with the following fields:

Cycle points related to the fault's entrance and exit

Entrance and exit points from the fault are determined based on the input shaft's rotational cycle. The cycle starts with the interaction between the initial contact point of the input wheel and the tooth tip of the output wheel. Given that breakage is seeded to the first interacting output wheel tooth, the entrance cycle is inherently zero. Calculating the exit cycle is more intricate and it involves the contact ratio ε, which represents the gearmesh cycles elapsed from tooth engagement to separation. The relative duration along the x-axis from tooth tip to chipping edge, multiplied by ε, indicates gearmesh cycles during fault interaction, as illustrated in **Figure 13**. With z_{in} as the gearmesh cycles per input shaft cycle, the fault's exit cycle point is derived by dividing this product by z_{in}.

$$cyc_{in}(\text{entrance}) = 0$$

$$cyc_{in}(\text{exit}) = \frac{\text{tooth\_tip} - \text{chip\_edge}}{\text{tooth\_tip} - \text{init\_cont}} \times \varepsilon \times \frac{1}{z_{in}}$$

Variation of tooth width along x-axis

The defect width increases linearly from zero at the chipping edge to z_{def} at the tooth tip, as illustrated in **Figure 13**. Therefore, the tooth width changes along the x-axis within the fault boundaries. As a result, symmetry is disrupted, and the center of area along the z-axis becomes dependent on x. The following equations summarize the variation of the tooth width along the x-axis:

$$W_{def}(x) = \frac{z_{def}}{x_{def}} \times \big(x - X(\text{chipping edge})\big)$$

$$W(x) = W_0 - W_{def}(x)$$

$$Z_c(x) = 0.5 \times W_{def}(x)$$

Where W_0 is the healthy tooth width, W_{def} is the defect width, Z_c is the coordinate of the center of area along the z-axis, and x is the independent variable within the fault boundaries.

It's important to highlight that further calculations for the variable gearmesh stiffness depend, among others, on the contact length along the z-axis. The variation of the tooth width along x should be taken into account for the other wheel (in this instance, the input wheel) as well. Consequently, the vector $W_{def}(x)$ is flipped, trimmed within contact boundaries, and then interpolated to align with the dimensions of the other wheel's profile. As the other wheel is assumed to be healthy, geometric symmetry is upheld, maintaining the center of area coordinate at zero along the z-axis.

genToothDestructionProfile

The provided function computes the profile of a destructed involute tooth in the x-y plane. The destructed profile is represented as a linear line in 2D (or a plane in 3D), uniformly removing the involute profile across the tooth face, as depicted in **Figure 14**. It takes as input the wheel profile and another structure containing fault dimensions, updating the defect information structure in the output with the damaged profile and fault indices.

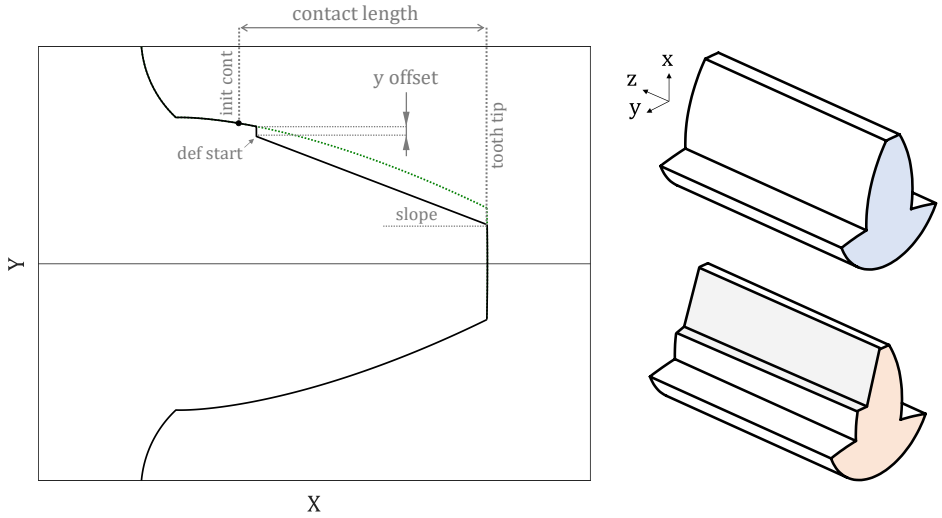

Figure 14 - Illustrative sketches and definitions of the simulated destructed tooth

The destruction of the involute tooth is defined by three key parameters illustrated in **Figure 14**: the starting point of the defect represented by the Cartesian coordinates $[x_{start}, y_{start}]$, the y-offset from the involute curve at the defect's starting point, and the slope (i.e., angle) of the linear line responsible for eliminating the involute profile (the function inherently negates its value, disregarding the user-inputted sign). These parameters are essential for formulating the defect linear line ($\mathrm{def\_line}$) equation, expressed as follows:

$$\mathrm{def\_line}(x) = \tan(\mathrm{slope}) \times (x - x_{start}) + (y_{start} - y_{offset})$$

The modified profile is created by substituting the original involute profile with the defect line within the relevant boundaries. It is crucial to emphasize the following clarifications in this regard:

Defect starting point

The user provides a coefficient in the open interval (-1,+1), instead of the defect starting point directly. This coefficient signifies the distance from the initial contact point relative to the entire contact length, as depicted in **Figure 14**. A coefficient of zero aligns the defect starting point with the initial contact point. Although negative coefficients are allowed, any defect indices preceding the initial contact index after line generation are discarded as regions lacking contact are not susceptible to destruction in this specific fault scenario.

Defect ending point

The endpoint of the destructed profile is determined by the last intersection of the defect line with the involute profile, and it may not necessarily coincide with the last x-point on the involute profile.

Fault displacement

The fault displacement is defined as the deviation between the ideal involute and the destructed profile *along the line of action*. We assume that this displacement, when multiplied by the gear mesh stiffness, generates a meshing force resulting from the fault, as long as other sources, such as surface quality errors and additional faults causing involute distortion. Since we assume that this fault distorts the involute profile along the y-axis while the x-axis remains constant, the calculation of the fault displacement involves projecting the y-distance between the involute and destructed profile in the pressure line direction for each x-point, as illustrated in **Figure 2**. The angle φ represents the angle of the tangent line to the involute profile, and slope denotes the inclination angle (negative angle) of the destructed profile, set by the user. Through simple geometric manipulation and the application of the sine theorem to the yellow triangle in **Figure 2**, the following formula can be derived:

$$\mathrm{FaultDisplacement} = (\mathrm{Y} - \mathrm{Y_{def}}) \times \frac{\cos(\mathrm{slope})}{\cos(\varphi + \mathrm{slope})}$$

Final steps involve trimming and flipping the fault displacement within the contact limits. Additionally, profile modifications, such as tip relief, are taken into consideration if they exist.

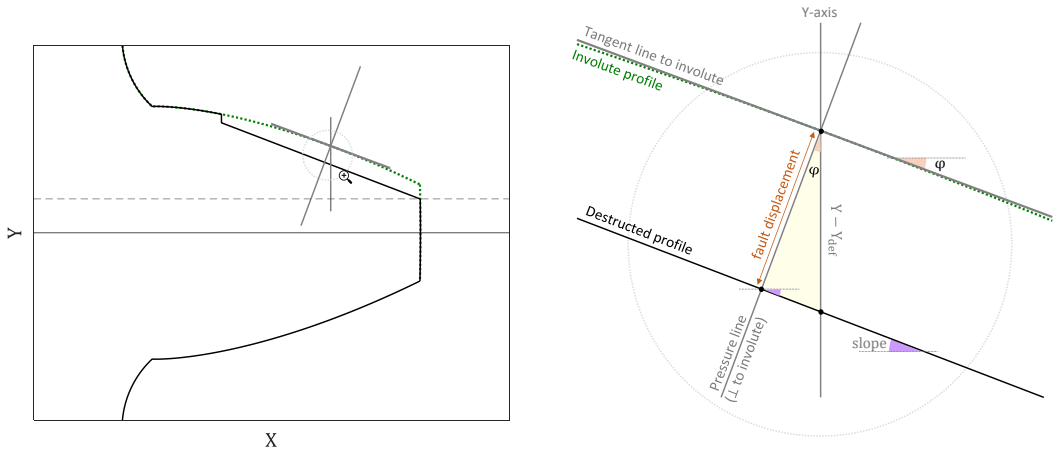

Figure 15 – Procedure for calculating fault displacement in tooth destruction faults

genToothFaceFaultProfile

This function generates local tooth face faults, resembling pitting faults common in gear transmission. Tooth face faults include '`PartialFaceFault`' (removal of material from part of the tooth face) and '`ThroughFaceFault`' (removal of material from the entire tooth face). The model simulates a circular 'electrode' in the x-y plane with predefined radius and center coordinates, removing material from the operational tooth face, as shown in **Figure 16**.

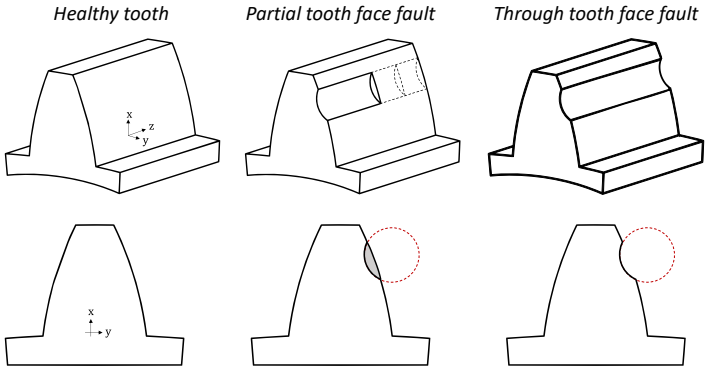

Figure 16 - Qualitative sketches of healthy tooth, tooth with partial tooth face fault, and a tooth with through tooth face fault

While `PartialFaceFault` and `ThroughFaceFault` share some intuitive similarities, they are treated differently in the model and thus necessitate distinct parameters for subsequent calculations. This section will introduce the shared parameters to both fault types, followed by separate sections explaining the specific parameters for each fault type. This function invokes several functions exclusively designed for tooth face faults, defined within `generate_tooth_face_fault_profile`. These functions will be explained in detail in this section. **Figure 17** illustrates the general flow of this function and the subfunctions called at each stage.

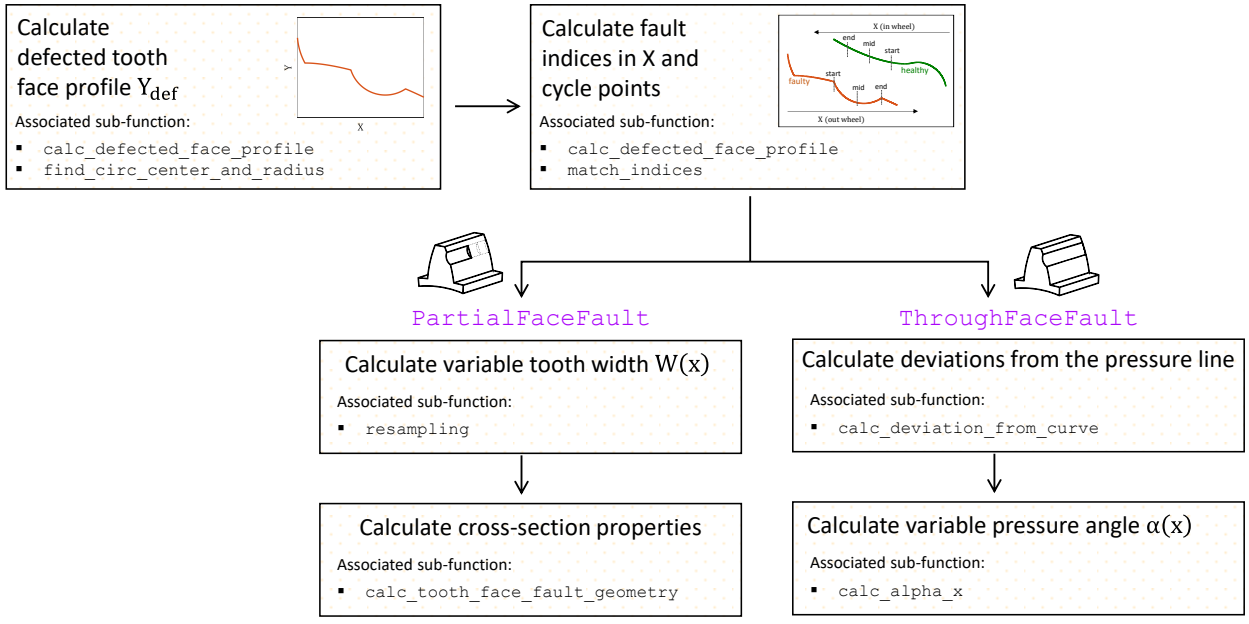

Figure 17 – A block diagram showing the general flow of generating tooth face fault profile

Fault-related cycle points calculations

Inside `genToothFaceFaultProfile`, most calculations occur in subfunctions. One critical calculation involves converting fault indices in X to cycle points of the input wheel. It is shown that with ε representing the contact ratio, and z_{in} is the number of teeth on the input wheel, a total number of ε/z_{in} cycles of the input shaft occur from the moment a tooth pair engages at $[X_{in}(\text{init\_cont}), X_{out}(\text{tip})]$ to separation at $[X_{in}(\text{tip}), X_{out}(\text{init\_cont})]$. Consequently, fault-related cycle points can be calculated by multiplying this cycle count by the relative proportion of fault indices along X_{out}. The cycle point when the interaction with the fault begins corresponds to the relative duration from $X_{out}(\text{tip})$ to $X_{out}(\text{def\_end})$, while the cycle point when the interaction with the fault ends corresponds to the relative duration from $X_{out}(\text{tip})$ to $X_{out}(\text{def\_start})$, as follows:

$$\text{cyc}_{in}(\text{start}) = \frac{\varepsilon}{z_{in}} \times \frac{X_{out}(\text{def\_end}) - X_{out}(\text{tip})}{X_{out}(\text{init\_cont}) - X_{out}(\text{tip})} \qquad \text{cyc}_{in}(\text{end}) = \frac{\varepsilon}{z_{in}} \times \frac{X_{out}(\text{def\_start}) - X_{out}(\text{tip})}{X_{out}(\text{init\_cont}) - X_{out}(\text{tip})}$$

`calcDefectedFaceProfile`

The damaged tooth face profile, denoted as $Y_{def}(x)$, is generated by "subtracting" the overlapping area between the healthy tooth face profile $Y(x)$ and a circle representing the fault geometry, as shown in **Figure 18**. This process involves the following steps:

1. Calculating the radius and center coordinates of the circle. Users have two options: they can either input the radius and circle center directly or use three coefficients (ranging from 0 to 1). These coefficients correspond to the fault edges along the active involute profile and the depth coefficient, representing a percentage of the half-chord connecting the fault edges. The latter option calls the function `findCircCenAndR` described below.
2. Calculating the coordinates of the circle's circumference within the angle range of $[\pi, 2\pi]$. Only the lower semicircle is considered for tooth face faults, as scenarios where more than half of the circle overlaps with the tooth profile are unrealistic. The segment of the tooth profile that matches the x-coordinates of the semicircle is interpolated to align with its dimensions. Finally, the fault edges based on the points with the smallest distance between the semicircle and the involute profile are identified.
3. Calculating the damaged tooth face profile $Y_{def}(x)$ by interpolating the relevant segment in the fault circle to match the dimensions of the corresponding segment in the healthy tooth face profile and replacing it with the defective region.

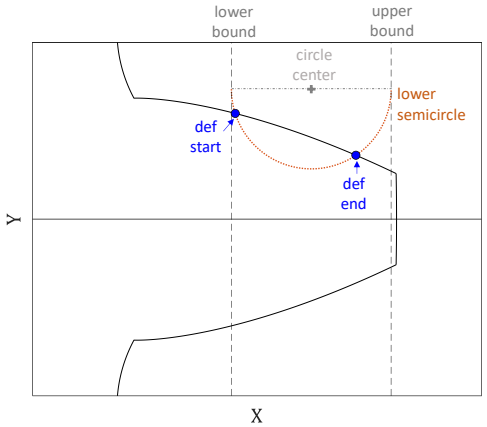

Figure 18 - Illustration of the tooth face fault parameters

`findCircCenAndR`

This function calculates the radius and center coordinates of a circle using two points on its circumference and a depth coefficient, as depicted in **Figure 19**. The depth coefficient, ranging from 0 to 1, defines the depth as a percentage of the half-chord. Here is how this function works:

1. Calculate the chord length between two points: $\text{chord} = |\text{point}_1 - \text{point}_2|$.

2. Define the depth as complementary to the chord bisector: $\text{depth} = \text{depth\_coeff} \times \text{chord}/2$.

3. Calculate the radius using the Pythagorean equation (see triangle in **Figure 19**):

$$\text{radius} = \text{chord}/4 \times (\text{depth\_coeff} + 1/\text{depth\_coeff})$$

4. Determine the coordinates of the circle center by finding the perpendicular bisector to the chord. The magnitude of the bisector is obtained from the Pythagorean equation, and the direction is perpendicular to the unit vector of the chord:

$$|\text{bisector}| = \sqrt{\text{radius}^2 - (\text{chord}/2)^2}$$

$$\widehat{\text{chord}} = (\text{point}_2 - \text{point}_1)/\text{chord} = (+\text{n}_1 \quad +\text{n}_2)^{\text{T}}$$

$$\widehat{\text{bisector}} = (-\text{n}_2 \quad +\text{n}_1)^{\text{T}} = \text{flip}\left(\widehat{\text{chord}}\right) \odot (-1 \quad +1)^{\text{T}}$$

5. Obtain two potential circle center vectors using vector addition or subtraction between the midchord and the bisector: $(\text{x}_\text{c} \quad \text{y}_\text{c})^{\text{T}} = \text{midchord} \pm |\text{bisector}| \times \widehat{\text{bisector}}$. In this context, we select the solution with the upper center due to the specific nature of the simulated tooth face fault, as shown in **Figure 19**.

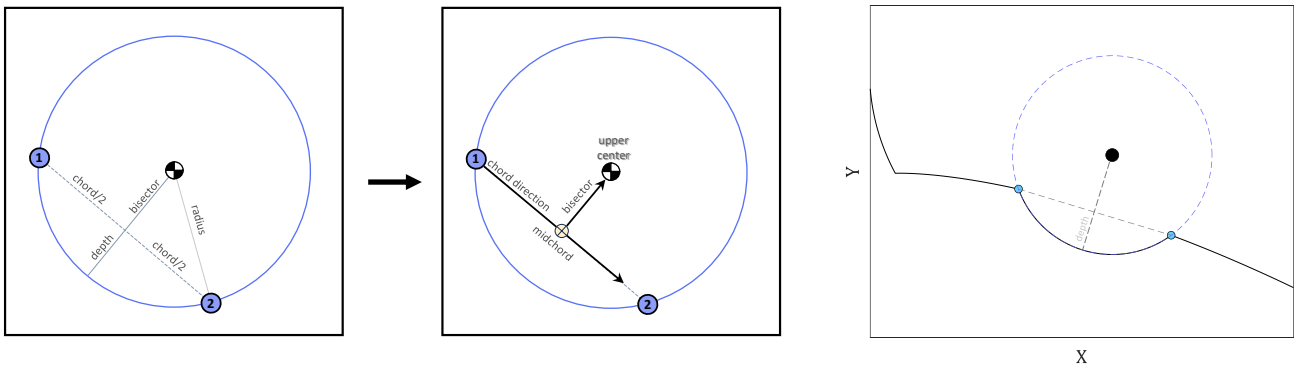

Figure 19 - Stages of calculating circle center and radius with a representative application in the model.

matchIndices

This function is employed to align the dimensions of the tooth profiles between the input and output wheels. Because the wheels have different number of points within the contact region, a conversion becomes necessary for specific operations in the model. It is assumed that a linear relationship exists between the indices of the source (s) and the target (t). For instance, the initial index of the gear (e.g., 2,700) corresponds to the index of the pinion tip (e.g., 10,000), and the index of the gear tip (e.g., 10,000) aligns with the initial index of the pinion (e.g., 3,100). A qualitative illustration is provided in **Figure 20**.

The function comprises two straightforward steps: first, a linear interpolation, and then the rounding of the interpolated value to ensure it becomes an integer index, as outlined below:

$$t_{ind} = \text{round}\left[t_{start} + \frac{t_{end} - t_{start}}{s_{end} - s_{start}} \cdot (s_{ind} - s_{start})\right]$$

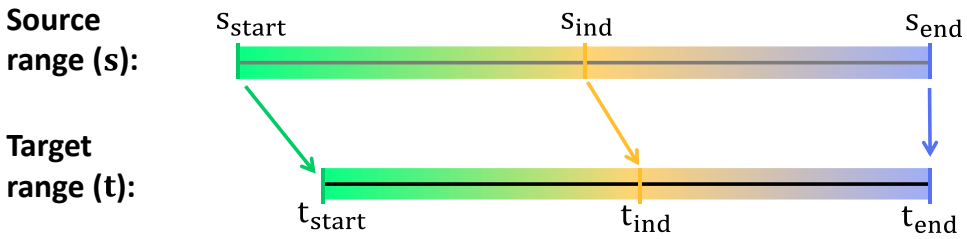

Figure 20 - An illustration of index matching by linear interpolation.

calcToothFaceFaultGeometry

This function calculates key geometric properties of the cross-section in the z-y along the x-axis in case of a partial tooth face fault, including the area $A(x)$, the center of area coordinates $Y_c(x)$ and $Z_c(x)$, the moments of inertia $I_y(x)$ and $I_z(x)$, as well as the product of inertia $I_{yz}(x)$.

In the healthy state, the cross-sectional area has a uniform rectangular shape. However, when a tooth face fault occurs, the area alternates between a rectangle in healthy x-indices and a polygon in defective x-indices. In the y-z plane, the defective area takes the form of a rectangle subtracted from the operational tooth face, as depicted in **Figure 21**. This defective area is characterized by three parameters illustrated in **Figure 21**, which include the coordinates of the defective operational face $Y_{def}(x)$, the defect width along the z-axis W_{def}, and a reference z-coordinate of the fault edge z_0, which is set to zero by default. Geometrical properties are determined using the superposition principle of two rectangles corresponding to the healthy (h) and defective (def) areas, represented in blue and red in **Figure 21**.

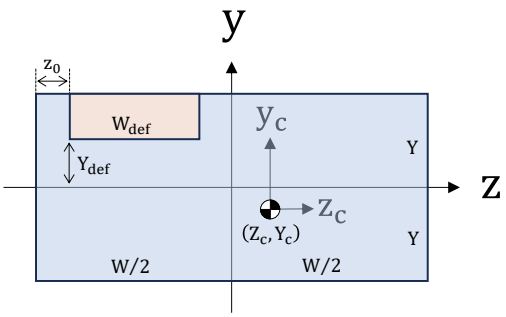

Figure 21 - An illustration of the cross-section in the y-z plane in the tooth face fault status

$$A = A_h - A_{def} \qquad Y_c = \frac{Y_{c_h} A_h - Y_{c_{def}} A_{def}}{A} \qquad Z_c = \frac{Y_{c_h} A_h - Y_{c_{def}} A_{def}}{A}$$

Moments of inertia and the product of inertia are computed using the superposition principle and Steiner's theorem as following where i, j, ℓ can be either y or z:

$$\Delta_i = \ell_{c_i} - \ell_c \qquad I_{ij} = I_{ij_c} + \Delta_i \Delta_j \cdot A \qquad I_{ij} = I_{ij_h} - I_{ij_{def}}$$

The necessary parameters for computing geometric properties are outlined in **Table 1**, derived from **Figure 21.**

Table 1 - Geometric properties of tooth face fault

	A_i	Y_{c_i}	Z_{c_i}	$I_{y_{c_i}}$	$I_{z_{c_i}}$	I_{zc_i}
Healthy (h)	$2WY$	0	0	$W^3Y/6$	$2WY^3/3$	0
Defect (def)	$W_{def}(Y - Y_{def})$	$0.5(Y + Y_{def})$	$z_0 - 0.5(W - W_{def})$	$W_{def}^3(Y - Y_{def})/12$	$W_{def}(Y - Y_{def})^3/12$	0

`calcDeviationFromCurve`

This function takes Cartesian coordinates $[X, Y]$ for a curve along with indices representing two points on the curve. It stretches a chord between these two points and calculates the maximum deviation between the curve and the chord. **Figure 22** provides a qualitative illustration of this function's implementation and its application in the model for a case of a through tooth face fault. The indices correspond to the fault edges, while the curve represents the involute profile. Given the complexity of involute profiles, a simple and straightforward method for determining the maximum deviation, denoted as h, involves rotating the chord horizontally with respect to the y-axis and then calculating the maximum distance between this horizontal chord and the curve within the range defined by the specified indices.

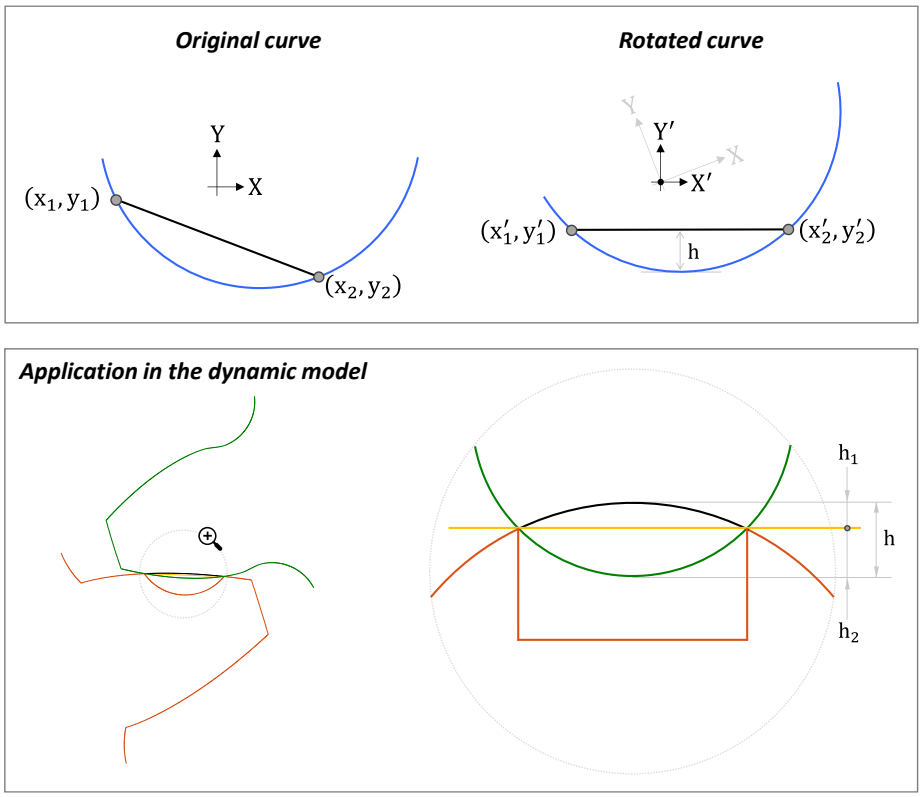

Figure 22 - Top: Illustration of finding maximal deviation from curve using rotational matrix. Bottom – Illustration for an application of the function in modeling through tooth face faults.

calcAlphaX

This function calculates the variation of the pressure angle α along the x-axis for both wheels in the case of a through tooth face fault, given the indices of the fault edges along the x-axis of both the damaged and healthy wheels. Surprisingly, the healthy wheel plays a key role in this function. When considering a point P_x on the involute profile of the healthy wheel (i.e., the wheel without the defect) at a specific x-coordinate, the distance R_x represents the radius from the wheel center to point P. In this modeling of the through face fault, we assume that the healthy tooth deviates from the nominal pressure line and instead rotates around the edges of the fault on the damaged wheel, as illustrated in **Figure 3**. These deviations from the nominal pressure line result in variations in the pressure angle $\alpha(x)$, which depend on the position along the x-axis for this fault.

Figure 23 depicts the scenario where the driving wheel is healthy while the driven wheel is defective. In this case, the interaction with the fault on the healthy wheel occurs from the ending to the starting points of the defect, as shown in **Figure 23**. Specifically, from the ending point to the middle point of the defect, the healthy tooth rotates around the ending edge of the fault. Meanwhile, from the middle point to the starting point, the healthy tooth rotates around the starting edge of the fault, as illustrated in **Figure 23**. To calculate the pressure angle for this fault, we determine the angle corresponding to the arclength $\widehat{GT}$ on the base circle, as defined in **Figure 23**. This angle is the angle between the radius extending from the point P_x and the tangent line to the base circle drawn from the actual contact point P_{edge}, where edge can be either start or end. The formula for the variable pressure angle is derived below, utilizing principles presented in the function **genHealthyToothProfile**.

$$R_b \cdot \alpha(x) = \widehat{GT} = \widehat{GQ} + \widehat{ST} - \widehat{SQ} \quad \rightarrow \alpha(x) = \frac{\widehat{GQ}}{R_b} + \frac{\widehat{ST} - \widehat{SQ}}{R_b} = \alpha_0 + \frac{\overline{P_{edge}T} - \overline{P_xT}}{R_b}$$

$$\rightarrow \alpha(x) = \alpha_0 + \frac{\sqrt{R_{edge}^2 - R_b^2} - \sqrt{R_x^2 - R_b^2}}{R_b} \quad (edge = start, end)$$

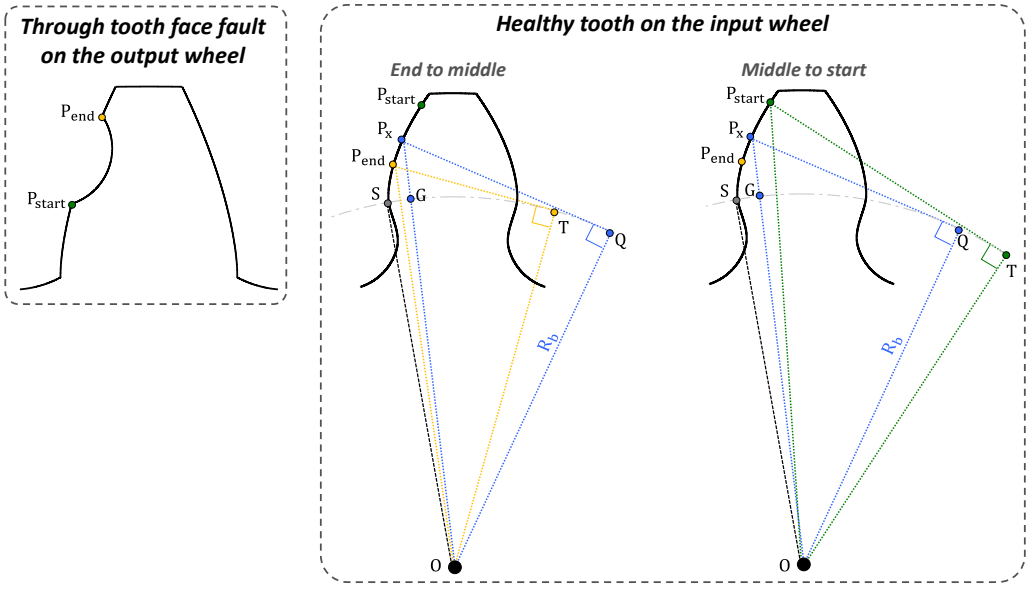

Figure 23 - Illustration of the calculation of the variable pressure angle in case of a through tooth face fault

calcAlphaCyc

The pressure angle indicates the direction of the meshing force, defined by the angle of the pressure line. In spur gears, the pressure line remains constant ideally. However, when tooth faults, like through face faults, cause deviations in the pressure line, the pressure angle changes. This function calculates the variable pressure angle for through tooth face faults based on the input shaft's cycle, following the same principles as the `calcAlphaX` function and its output. For more details, refer to `calcAlphaX`.

calc_fault_displacement

This function calculates the displacement resulting from local or distributed tooth faults, which cause deviations from the nominal pressure line. The maximum deviation accounts for both the input and output wheels' deviations. When the output wheel is assumed to be defective, the interaction with the fault happens once during a cycle of the output wheel. The interaction frequency with the fault is determined by the transmission ratio (tr), which is the ratio between the number of teeth on the wheels: $\mathrm{tr} = \mathrm{z_{out}}/\mathrm{z_{in}}$. Consequently, the displacement depends on the relative cycle's modulo with respect to the transmission ratio (tr).

The relative cycle is defined as follows:

$$\text{relative cyc} = \mathrm{rem}\left(\frac{\mathrm{cyc_{in}} - \mathrm{cyc_{in}(start)}}{\mathrm{cyc_{in}(end)} - \mathrm{cyc_{in}(start)}}, \mathrm{tr}\right)$$

The function describing the fault displacement must meet the following constraints:

- It exhibits symmetry and cyclic behavior, completing a single rolling cycle between the fault's starting point ($\text{relative cyc} = 0$) and its ending point ($\text{relative cyc} = 1$).
- The displacement remains continuous and smooth at the fault edges, characterized by both a zero value and a zero slope.
- The maximum displacement occurs at the midpoint of the rolling cycle between the edges ($\text{relative cyc} = 0.5$) and equals the negative value of h, which is the maximal deviation calculated in the function `calc_deviation_from_curve` (see **Figure 22**).

The function satisfying all these requirements is depicted in **Figure 24** and is written as follows:

$$\text{fault displacment} = -\frac{\mathrm{h}}{2} \cdot [1 - \cos(2\pi \cdot \text{relative cyc})]$$

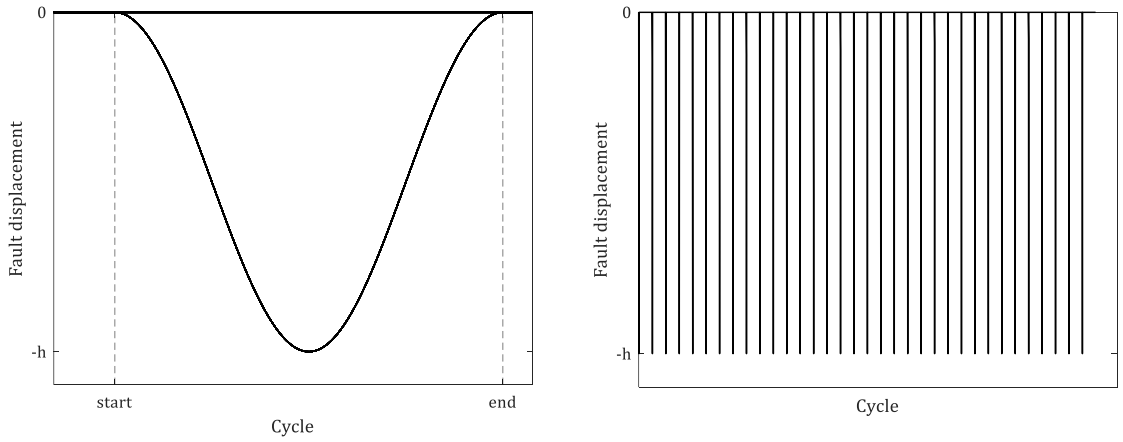

Figure 24 - Left: Fault displacement in case of a through tooth face fault in one fault's rolling cycle. Right: The pattern of fault displacement for multiple cycles of the input wheel.

`calcGearMeshStiff`

This function calculates the gear mesh stiffness (**gms**) as a function of the input shaft's rotational cycle. The gms exhibits a square-wave-like curve in shape, as each meshing tooth pair sequentially contributes its stiffness to the overall gms at distinct intervals. **Figure 25** illustrates the process of gms calculation and its cyclic nature in both the healthy state and in the event of a tooth breakage in the output wheel.

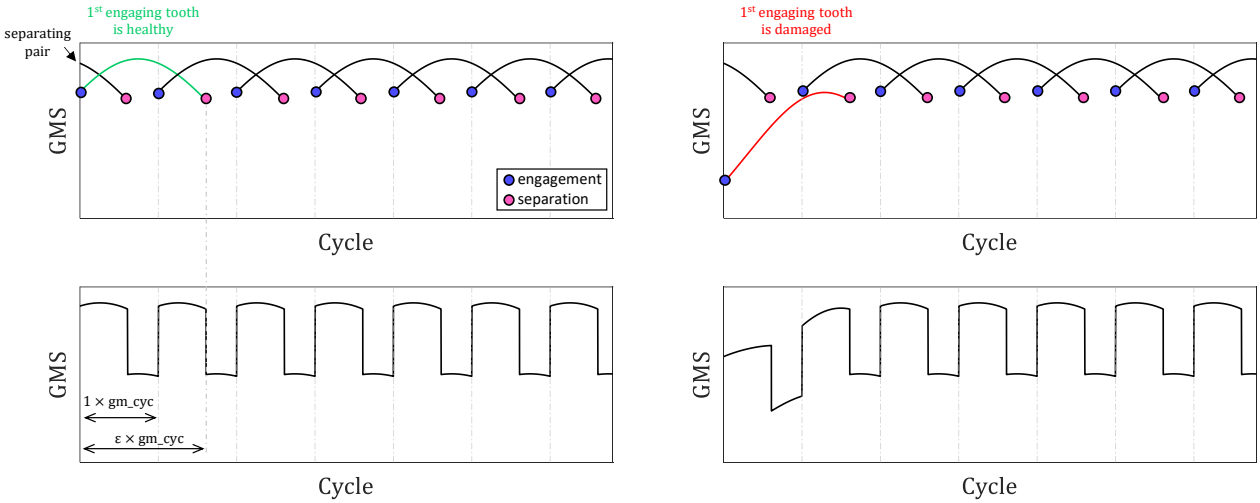

Figure 25 - An illustration of the gms calculation for a healthy gear (left) and damage gear with a tooth breakage fault (right).

The **`calcKeq`** function calculates the equivalent stiffness (K_{eq}) of a single tooth pair, spanning from tooth engagement to tooth separation. It is important to understand that the gear mesh frequency characterizes the frequency of tooth engagements or tooth separations along the line of action. Meanwhile, the contact ratio (ε) is defined as the number of gear mesh cycles that occur from the moment a tooth engages until it separates. Consequently, the duration of K_{eq} spans ε gear mesh cycles, as illustrated in **Figure 25**. With z_{in} teeth on the input wheel meshing in one shaft cycle, the gear mesh (gm) cycle is:

$$gm\_cyc = 1/z_{in}$$

The calculation of the gms involves several steps. Initially, we compute the equivalent stiffness of a healthy tooth pair. Then, we create a vector of cycle points spanning from 0 to $\varepsilon \times gm\_cyc$ that matches the dimensions of the equivalent stiffness. Subsequently, we construct another vector of cycle points with a finer resolution and interpolate the equivalent stiffness of a healthy tooth pair to match this finer resolution.

Next, we proceed to calculate the total gms in a healthy state. During this stage, we calculate the periodic pattern of the gms within a single gear mesh cycle, presented in **Figure 26**, by aggregating the remaining stiffness contributed by the separating tooth pair and the total stiffness of the engaging tooth pair, as illustrated in **Figure 25**. This pattern is then repeated and concatenated to cover a complete cycle of the output shaft, specifically for z_{out} gear mesh cycles. It's important to emphasize that all calculations assume

a contact ratio ranging from 1 to 2. This implies that the transition in the overall gms occurs between two meshing pairs to a single tooth pair meshing.

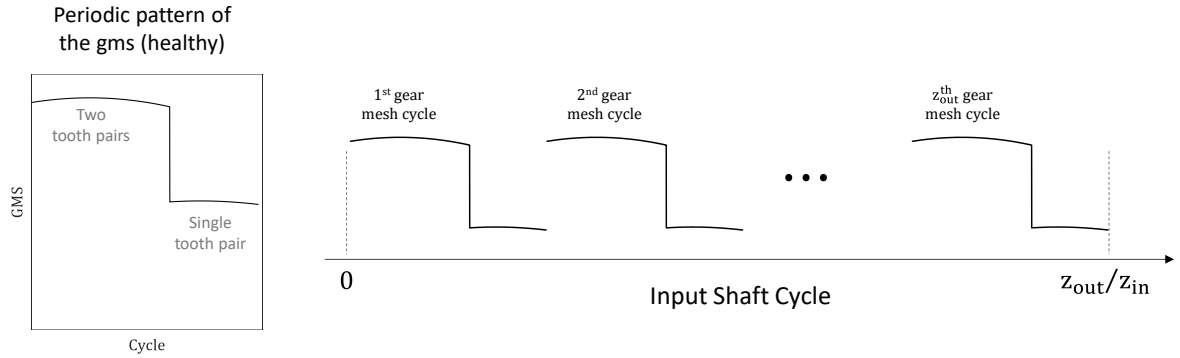

Figure 26 - An illustration of building the overall gms by repeating and concatenating the periodic pattern of the gms

In case of a faulty gear, the damaged tooth is seeded to the first meshing tooth in the calculated gms. After computing the equivalent stiffness of the defective tooth using the function **`calcKeq`**, the gms is updated by subtracting the equivalent stiffness of a healthy tooth from the beginning of the gms and replacing it with the equivalent stiffness of the damaged tooth, as illustrated in **Figure 25**.

Finally, for computational efficiency, the gms is downsample. It's important to note that the structure of the gms retains the gms in fine resolution for documentation purposes.

calcKeq

This function is responsible for calculating the equivalent stiffness K_{eq} of a single tooth pair along the line of action for every X point along the output wheel profile. The equivalent stiffness vector is defined by the starting point, which is the initial contact point on the involute profile of the input wheel ($R_{\text{init cont}_{in}}$), and the tooth tip of the output wheel ($R_{a_{out}}$). It extends to the tooth tip of the input wheel ($R_{a_{in}}$) and the initial contact point on the involute profile of the output wheel ($R_{\text{init cont}_{out}}$), as shown in **Figure 27**.

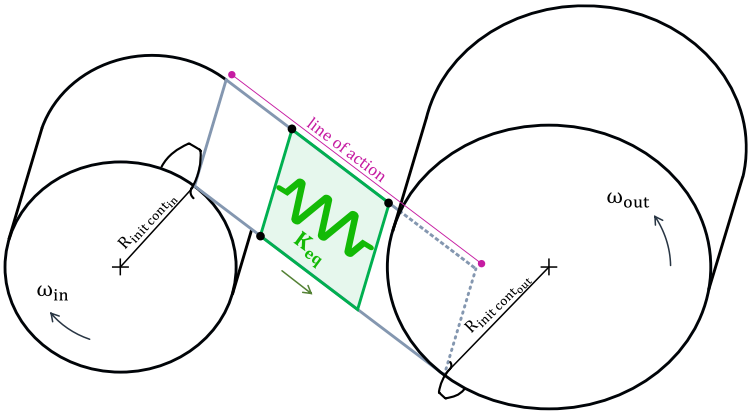

Figure 27 - Illustration for the equivalent stiffness of a tooth pair along the line of action

The equivalent stiffness, represented as K_{eq}, is estimated by considering it as the combined stiffness of nine springs connected in series. These nine springs include the Hertzian contact stiffness (K_h), and four stiffnesses for each wheel ($\text{wheel} = \text{in}, \text{out}$), corresponding to axial stress (K_a), shear stress (K_s), bending stress (K_b), and residual stress induced by the fillet foundation (K_f). Calculating the equivalent stiffness involves taking the inverse of each individual stiffness and summing them together, as illustrated in **Figure 28** and calculated as follows:

$$K_{\text{wheel}}^{-1} = K_a^{-1} + K_b^{-1} + K_s^{-1} + K_f^{-1}$$

$$K_{eq}^{-1} = K_h^{-1} + K_{in}^{-1} + K_{out}^{-1}$$

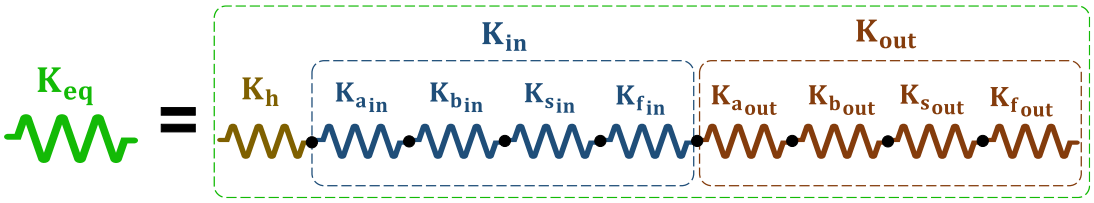

Figure 28 - The equivalent stiffness K_{eq} as a combination of nine springs connected in series.

The inverse stiffness of the input and output wheels, and so as the inverse Hertzian stiffness, are calculated separately before being summed. For each wheel, the stiffness is calculated along the tooth profile, starting from the initial contact point on the involute profile and extending to the tooth tip. This calculation choice requires two post-processing actions. Firstly, the inverse stiffness of the output wheel, K_{out}^{-1}, is flipped to ensure proper matching with the input wheel. Secondly, the inverse stiffness of the

input wheel, K_{in}^{-1}, is resampled to align with the dimensions of K_{out}^{-1} due to the varying resolution of the wheel profiles along the X-axis. The same flipping procedure is applied to the inverse Hertzian stiffness, K_{h}^{-1}, and finally, the inverse equivalent stiffness, K_{eq}^{-1}, can be calculated as illustrated in **Figure 29**.

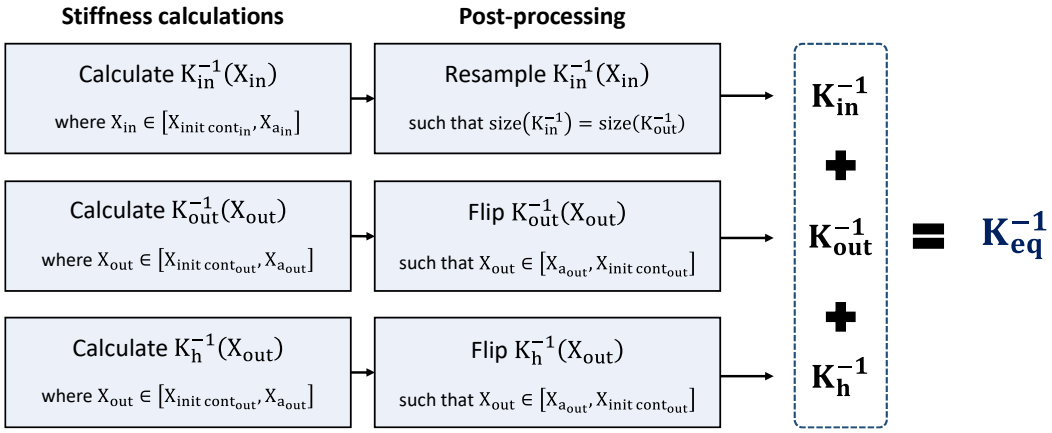

Figure 29 - Calculation procedure for inverse equivalent stiffness

Handling different health status cases

- In the case of a **healthy tooth pair**, regardless of the health status of the transmission, the function is initially called with only two input arguments containing the wheel information, instead of the expected four arguments including defect information. However, since the function requires the presence of the defectInfo structure, a conflict arises as it does not exist in this scenario. The function handles the absence of defect information and generates a degenerated defectInfo structure specifically for the healthy status. This degenerated structure includes the status label ('healthy') and the pressure angle α, ensuring proper functioning of the function in the absence of specific defect information.

- In the event of a **missing tooth fault**, where there is a loss of contact along the line of action, the equivalent stiffness is automatically set to zero. This occurs without any additional calculations, as the absence of contact renders further stiffness calculations unnecessary.

- In the case of a **partial tooth face fault** or **tooth breakage**, the width of the affected tooth becomes variable along the X-axis. To account for this variable tooth width, it is referred to as $W_{x_{in}}$ and $W_{x_{out}}$, denoting the tooth width at a specific X-coordinate for the input and output wheels, respectively.

- The **tooth breakage** fault is identified by chipping, which may or may not result in **tip loss**. In the case of tip loss, where contact is no longer present, the equivalent stiffness is manually replaced and set to zero for the affected indexes where the tooth tip is missing. This ensures that the absence of contact is properly accounted for in the calculation of the equivalent stiffness.

`calcInv(Axial/Bending/Shear)Stiff`

The procedure for computing the equivalent stiffness of an individual tooth pair, spanning from tooth engagement to separation along the x-axis, is detailed in **`calcKeq`**. In this section the inverse stiffness profiles corresponding to axial, shear, and bending stresses are derived. The inverse stiffness profiles are calculated separately for each wheel, employing principles of beam theory. The meshing tooth is treated as a cantilever subjected to a meshing force $\bar{F}$ comprising compressive axial (F_a) and shear (F_s) components. This force is distributed uniformly along the contact line within the x-z plane while its location varies with operational position (X_i), as illustrated in the free-body diagram in **Figure 30**:

$$\bar{F}(X_i) = -F_a(X_i) \cdot \hat{x} - F_s(X_i) \cdot \hat{y} = -F \sin(\alpha_i) \cdot \hat{x} - F \cos(\alpha_i) \cdot \hat{y}$$

By setting a unit meshing force ($F = 1$), the inverse stiffness values (k_n^{-1} where $n = a, s, b$) are derived by calculating potential strain energies (U_n) for axial (a), shear (s), and bending (b) stresses, as following:

$$k_n^{-1}(x) = \frac{2U_n(x)}{F^2}$$

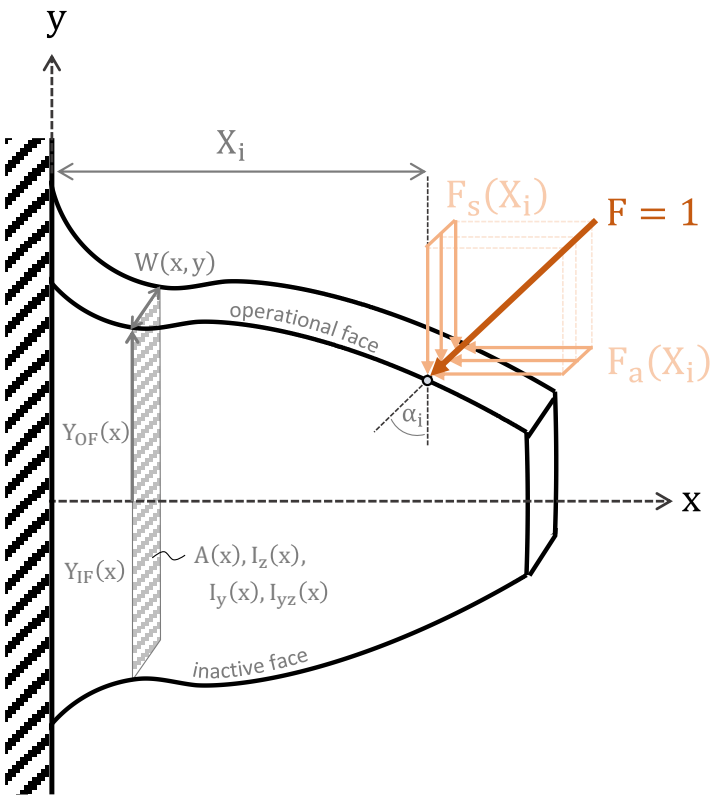

Figure 30 - A qualitative three-dimensional free-body diagram of a tooth in operation

Cumulative integration

The potential strain energies are calculated by integration along the x-axis between $x = 0$ and $x = X_i$, where X_i is the location of the meshing force along the tooth axis ($i = 1, \dots, N$). The terms inside the integral of the potential energy formulas consist of three groups of parameters: parameters related to the material (such as modulus of elasticity), parameters related to the applied load (force or moment), and parameters related to the geometry (such as cross-section properties). Most of the parameters are usually variable along the x-axis within the integral. However, the pressure angle α_i, and the coordinates of the radius to the meshing force ($X_i, Y_i, \bar{Z}_i$) are x-independent inside the integral.

To improve computation efficiency, two key principles are implemented:

1. Computing cumulative integral instead of iterating through $i = 1, \dots, N$. Cumulative integration in MATLAB is facilitated by the "`cumtrapz`" function.
2. X-independent terms (including $\bar{F}(X_i), X_i, Y_i, \bar{Z}_i$) are extracted from the cumulative integral, so instead of activating loops on the location along the tooth axis X_i and for each loop calculating the integral, the alternative of dot product, denoted $\odot$, is used, utilizing cumulative integration and vector multiplication.

`calcInvAxialStiff`

This function calculates the inverse stiffness derived from the potential strain energy of the tooth due to axial compressive stress. The axial strain energy is calculated as follows:

$$U_a(X_i) = \int_0^{X_i} \frac{F_a^2(X_i)}{2EA(x)}\,dx$$

An alternative form using cumulative integration and vector multiplication can be expressed as follows:

$$U_s(X) = F_a^2 \odot \int_0^{X} \frac{1}{2EA(x)}\,dx$$

where $F_a(X_i)$ is the axial force at X_i, E represents the modulus of elasticity, and $A(x)$ denotes the variable cross-sectional area along the x-axis. Specifically, the cross-sectional area in the y-z plane takes the form of a rectangle, as illustrated in **Figure 30**. Its base corresponds to the tooth width $W(x)$, while its height represents the distance between the tooth faces along the y-axis. Considering that the operational face (OF) can be either healthy or faulty, generally represented as $Y_{OF}(x)$, and the inactive face (IF) is assumed to be healthy, denoted as negative $Y(x)$, the expression of the cross-sectional area would be:

$$A(x) = W(x) \cdot \left(Y_{OF}(x) - Y_{IF}(x)\right)$$

$$A(x) = \begin{cases} W(x) \cdot 2Y(x) & Y_{OF} = Y \\ W(x) \cdot [Y_{OF}(x) + Y(x)] & Y_{OF} \neq Y \end{cases}$$

To summarize, the parameters that may vary with health status are the axial force $F_a(X_i)$, the tooth width $W(x)$, and the operational face $Y_{OF}(x)$. **Table 2** outlines the parameters subject to health status variation and those that remain constant according to the nominal healthy status.

Table 2 - Key parameters and their health status-related dependency in axial strain energy

Health status	Axial force $F_a(X_i)$	Tooth width $W(x)$	Operational face $Y_{OF}(x)$
`'Healthy'`	Constant	Constant	Healthy ($Y(x)$)
`'ToothBreakage'`	Constant	Variable	Healthy ($Y(x)$)
`'PartialFaceFault'`	Constant	Variable	Defected ($Y_{def}(x)$)
`'ThroughFaceFault'`	Variable	Constant	Defected ($Y_{def}(x)$)
`'ToothDestruction'`	Constant	Constant	Defected ($Y_{def}(x)$)

calcInvShearStiff

This function calculates the inverse stiffness derived from the potential strain energy of the tooth due to shear stress. The shear strain energy is calculated as follows:

$$U_s = \int_V \frac{\tau^2}{2G} dV$$

where τ is the shear stress, G is the shear modulus, and dV is differential element of the volume. The terms for the differential element of the volume and the shear stress can be written as follows:

$$\tau = \frac{F_s(X_i) \cdot Q_z(x,y)}{W(x,y) \cdot I_z(x)} \qquad dV = dAdx = W(x,y)dydx$$

In these formulas $F_s(X_i)$ is the shear force at $x = X_i$, $Q_z(x,y)$ is the first moment of the area, $W(x,y)$ is the varying tooth width and $I_z(x)$ is the area moment of inertia. By substituting these relations, the following term is obtained:

$$U_s(x = X_i) = \int_0^{X_i} \int_{Y_{IF}(x)}^{Y_{OF}(x)} \frac{F_s^2(X_i)\, Q_z^2(x,y)}{2GW(x)\, I_z^2(x)} \cdot dy\, dx$$

The first moment of the area is dependent in (x,y) and can be written as follows:

$$Q_z(x,y) = \int_A y dA = \int_y^{Y_{OF}(x)} yW(x,y)dy$$

Where y is the distance of an element dA from the neutral axis z, and the operational face $Y_{OF}(x)$ is the upper boundary of the cross-section area in the y-z plane. An alternative form after rearrangement using cumulative integration and vector multiplication can be expressed as follows:

$$U_s(X) = F_s^2 \odot \int_0^X \frac{1}{2GI_z^2(x)} \cdot \int_{Y_{IF}(x)}^{Y_{OF}(x)} \frac{Q_z^2(x,y)}{W(x,y)} dy\, dx$$

The shape of the cross-section in the z-y plane plays a pivotal role in calculating the shear strain energy, which is expressed through the utilization of the first and second moments of the area, and the tooth width. Two scenarios are examined: the typical case of a uniform rectangular cross-section and the atypical case of an irregular polygonal cross-section, denoted as Case I and Case II, respectively.

Case I: Uniform rectangular cross-section in the z-y plane

This case applies to the `'Healthy'` status, and can be extended to the `'Tooth Breakage'` status, where the tooth width varies only along the x-axis ($W(x)$), as well as the `'Through Face Fault'` status, characterized by a defect in the operational face. In all three of these statuses, the tooth width W remains independent of y, and the integration boundaries along the y-axis exhibit symmetry. To account for this, the distance from the neutral axis to both the operational and inactive faces is denoted as $\pm h(x)$, as illustrated in **Figure 31**.

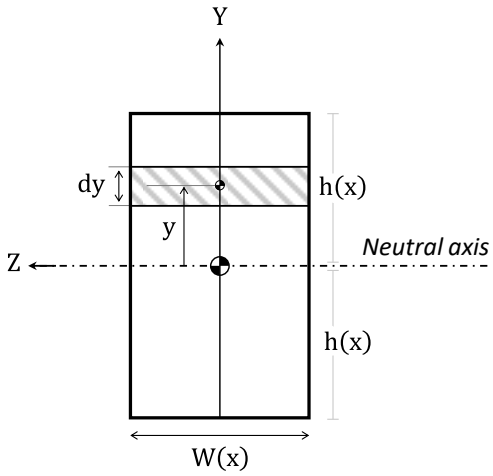

Figure 31 - An illustration of a rectangular cross-section in the z-y plane

In this case, the expression for the first moment of the area can be formulated as follows:

$$Q_z(x,y) = \int_y^{h(x)} yW(x)dy = \frac{W(x)}{2}[h^2 - y^2]$$

The inner integral along the y-axis in the expression for shear strain energy becomes:

$$U_s(X_i) = \frac{1}{8}\int_0^{X_i} \frac{F_s^2}{GI_z^2} \int_{-h}^{h} W(h^4 - 2y^2h^2 + y^4)dy\,dx \qquad \rightarrow U_s(X_i) = \frac{2}{15}\int_0^{X_i} \frac{F_s^2}{GI_z^2} Wh^5 dx$$

The area moment of inertia for a rectangle at the center of the area is calculated as follows:

$$I_z = \frac{1}{12}W \cdot (2h)^3 \rightarrow \quad I_z^2 = \frac{4}{9}W^2h^6$$

Finally, the expression for shear strain energy is derived and computed as follows:

$$U_s(X_i) = \frac{3}{10}\int_0^{X_i} \frac{F_s^2(X_i)}{GW(x)h(x)}dx$$

An alternative form using the cross-section area, and vector multiplication can be expressed as follows:

$$U_s(X) = F_s^2 \odot \int_0^X \frac{0.6}{GA(x)} dx$$

Case II: Inconsistent cross-section in the z-y plane

This case applies to the `'Partial Face Fault'`, wherein the tooth width varies along both the x and y axes ($W(x, y)$, as depicted in **Figure 32**). The operational face exhibits partial defects, resulting in a cross-sectional shape that alternates between a rectangle and a polygon, as illustrated in **Figure 33**. Consequently, the center of area in the z-y plane becomes x-dependent, leading to adjustments in the integration boundaries along the y-axis in relation to the center of area, as follows:

$$[Y'_{IF}(x), Y'_{OF}(x)] = [-Y(x) - Y_c(x), +Y(x) - Y_c(x)]$$

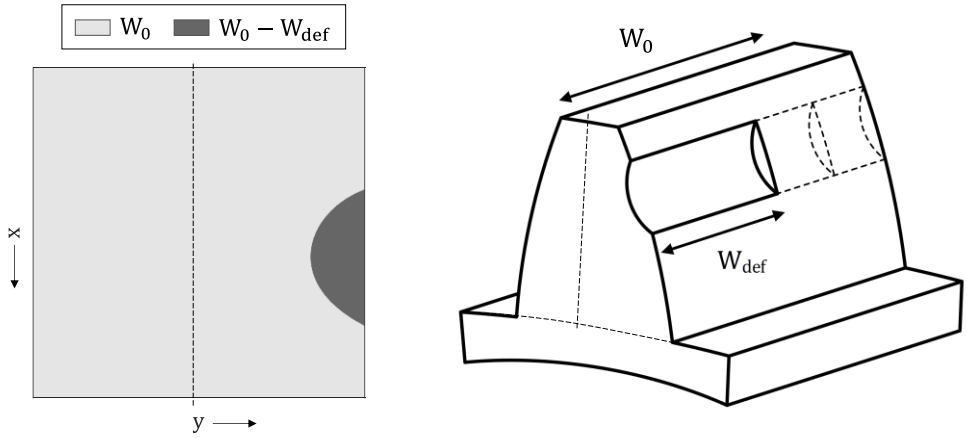

Figure 32 - An illustration of the tooth width variation along x and y axes in case of a partial tooth face fault

The process of computing shear strain energy involves the evaluation of the term within the integral along the x-axis, namely $\mathrm{integAlongX}$, initially configured in accordance with Case I. Subsequently, only the indices associated with the fault are updated before cumulative integration as follows:

$$\mathrm{integAlongX} = \frac{1}{2GI_z^2(x)} \cdot \int_{Y'_{IF}(x)}^{Y'_{OF}(x)} \frac{Q_z^2(x, y)}{W(x, y)} dy$$

To facilitate this, a y-matrix is computed. For each point in the x-axis among N, a vector with a predefined length L is generated to establish linear spacing between the inactive face $Y'_{IF}(x_{ind})$ and the operational face $Y'_{OF}(x_{ind})$. This vector is necessary for computing integrals along the y-axis for each fault index. The default number of points in the linearly spaced vector used for y-axis integration is set to $L = 100$; however, users have the flexibility to adjust this value when specifying parameters. The y-matrix can be expressed as follows:

$$y_{N\times L}=\begin{bmatrix} Y'_{IF}(x_1) & \cdots & Y'_{OF}(x_1) \\ Y'_{IF}(x_2) & \cdots & Y'_{OF}(x_2) \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ Y'_{IF}(x_N) & \cdots & Y'_{OF}(x_N) \end{bmatrix}$$

A for loop iterates only through the fault indices along the x-axis for computational efficiency. During each iteration (x_{ind}), both the first moment of the area, $Q_z(x_{ind}, y_{ind,1:L})$, and variable tooth width, $W(x_{ind}, y_{ind,1:L})$, are computed. Given that the for loop iterates over indices in the x-axis, integration along the y-axis is performed obviously without cumulative integration (i.e., using `trapz` function instead of `cumtrapz`). Finally, the potential energy is calculated through cumulative integration along the x-axis after updating fault information.

$$U_s(X)=F_s^2\odot\int_0^X \text{integ\_along\_x}\cdot dx$$

To summarize, the parameters that may vary with health status are the shear force $F_s(X_i)$, the cross-section shape, and consequently the tooth width $W(x,y)$. **Table 3** outlines the parameters subject to health status variation and those that remain constant according to the nominal healthy status.

Table 3 - Key parameters and their health status-related dependency in shear strain energy

Health status	Shear force $F_s(X_i)$	Cross-section shape	Tooth width $W(x,y)$	
			x-axis	y-axis
`'Healthy'`	Constant	Rectangle	Constant	Constant
`'ToothBreakage'`	Constant	Rectangle (defected)	Variable	Constant
`'PartialFaceFault'`	Constant	Inconsistent polygon	Variable	Variable
`'ThroughFaceFault'`	Variable	Rectangle (defected)	Constant	Constant
`'ToothDestruction'`	Constant	Rectangle (defected)	Constant	Constant

`calcInvBendingStiff`

This function calculates the inverse stiffness derived from the potential strain energy of the tooth due to bending stress. The bending strain energy is calculated as follows:

$$U_b(X_i) = \int_0^{X_i} \frac{M_z^2 I_y + 2M_z M_y I_{yz} + M_y^2 I_z}{2E\left(I_y I_z - I_{yz}^2\right)} dx$$

where M_z, M_y are the bending moments along z-axis and y-axis, respectively. E represents the modulus of elasticity. I_y, I_z are, respectively, the area momenta of inertia along the z-axis and y-axis, and I_{yz} is the product moment of inertia. Recall that the tooth is treated as a cantilever beam, subjected to axial and shear forces at the edge, located at distances $(X_i, Y_i, \bar{Z}_i)$ from the origin, where (X_i, Y_i) are the coordinates of the tooth profile in the x-y plane, and $\bar{Z}_i$ is the z-coordinate of the resultant axial force in the z-x plane, as referenced in **Figure 33**.

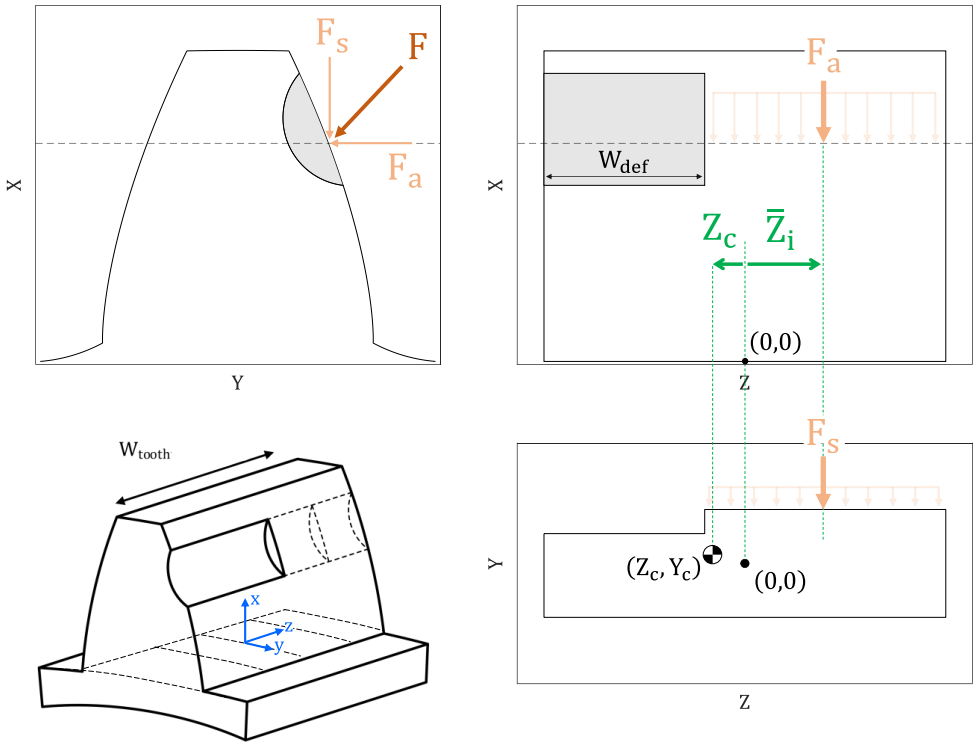

Figure 33 - An illustration of the tooth geometry and the meshing force distribution in three views

Assuming that the coordinate of the center of area in the y-z plane are x-dependent, represented as $Y_c(x)$ and $Z_c(x)$, the functions for the bending moments M_z, M_y are separately derived in relation to the center of area, using internal moments analysis from beam theory. The sections in the x-y plane and x-z plane are illustrated in **Figure 34** and are employed to calculate the profiles for M_z, M_y, respectively. The expressions of the internal moments in relation to x and the i^{th} index would be:

$$M_z(x,i) = F_s(x - X_i) + F_a(Y_i - Y_c)$$

$$M_y(x,i) = F_a(\bar{Z}_i - Z_c)$$

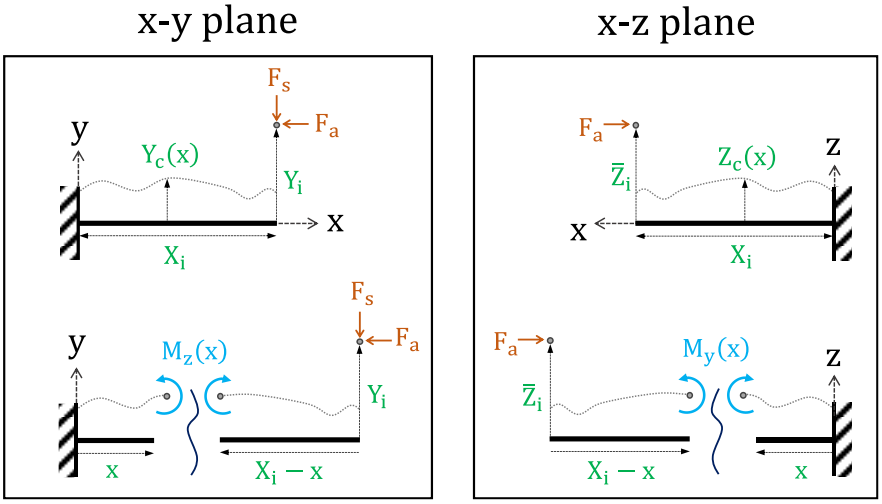

Figure 34 - Internal bending moments in the x-y and x-z planes

Each bending mode, including bending along the z-axis, bending along the y-axis, and mixed-mode bending, is analyzed separately. Within each bending mode, the flexural rigidity, denoted as D, is defined and independently computed. The total bending strain energy is divided into three distinct terms, each corresponding to one of the bending modes, and is computed separately. This process can be summarized as follows:

$$U_b(X_i) = \int_0^{X_i} \frac{M_z^2}{2D_z}dx + \int_0^{X_i} \frac{M_z M_y}{D_{yz}}dx + \int_0^{X_i} \frac{M_y^2}{2D_y}dx = U_{b_z} + U_{b_{yz}} + U_{b_y}$$

$$D_z = E\left(I_y I_z - I_{yz}^2\right)/I_y \qquad D_{yz} = E\left(I_y I_z - I_{yz}^2\right)/I_{yz} \qquad D_y = E\left(I_y I_z - I_{yz}^2\right)/I_z$$

$$U_{b_z} = \sum_{n=1}^{10} U_{b_{z_n}} \qquad U_{b_{yz}} = \sum_{n=1}^{8} U_{b_{yz_n}} \qquad U_{b_y} = \sum_{n=1}^{3} U_{b_{y_n}}$$

It is important to recall that, within the cumulative integration, both the pressure angle α_i (and consequently, the axial and shear components) and the coordinates of the radius to the meshing force $(X_i, Y_i, \bar{Z}_i)$ remain independent of the x. For the sake of computational efficiency, these terms are extracted from the cumulative integral and replaced with vector multiplication, denoted $\odot$. By substituting the expressions of the internal bending moment into the strain energy formulas and implementing the efficiency procedure, 21 terms of bending strain energies are derived for all three bending modes, as summarized in **Table 4**. In **Table 4**, distinct cell colors indicate different combinations of multiplying components of the meshing force, i.e., $F_a^2, F_a F_s, F_s^2$. This separation is implemented in the code, and it is essential for handling variable pressure angles in case of a through-tooth face fault, as elaborated in detail in the function **`correctUForThroughFaceFault`**.

Table 4 - Terms of bending potential energy, divided into bending modes

$\mathbf{U_{b_{z_n}}(X)}$	$\mathbf{U_{b_{yz_n}}(X)}$	$\mathbf{U_{b_{y_n}}(X)}$
$U_{b_{z_1}} = F_s^2 \odot \int_0^X \frac{x^2}{2D_z} dx$	$U_{b_{yz_1}} = F_a F_s \odot \bar{Z} \odot \int_0^X \frac{x}{D_{yz}} dx$	$U_{b_{y_1}} = F_a^2 \odot \bar{Z}^2 \odot \int_0^X \frac{1}{2D_y} dx$
$U_{b_{z_2}} = F_a F_s \odot \int_0^X \frac{-xY_c}{D_z} dx$	$U_{b_{yz_2}} = F_a F_s \odot X \odot \bar{Z} \odot \int_0^X \frac{-1}{D_{yz}} dx$	$U_{b_{y_2}} = F_a^2 \odot \bar{Z} \odot \int_0^X \frac{-Z_c}{D_y} dx$
$U_{b_{z_3}} = F_a^2 \odot \int_0^X \frac{Y_c^2}{2D_z} dx$	$U_{b_{yz_3}} = F_a^2 \odot Y \odot \bar{Z} \odot \int_0^X \frac{1}{D_{yz}} dx$	$U_{b_{y_3}} = F_a^2 \odot \int_0^X \frac{Z_c^2}{2D_y} dx$
$U_{b_{z_4}} = F_s^2 \odot X \odot \int_0^X \frac{-x}{D_z} dx$	$U_{b_{yz_4}} = F_a^2 \odot \bar{Z} \odot \int_0^X \frac{-Y_c}{D_{yz}} dx$	
$U_{b_{z_5}} = F_a F_s \odot Y \odot \int_0^X \frac{x}{D_z} dx$	$U_{b_{yz_5}} = F_a F_s \odot \int_0^X \frac{-Z_c x}{D_{yz}} dx$	
$U_{b_{z_6}} = F_a F_s \odot X \odot \int_0^X \frac{Y_c}{D_z} dx$	$U_{b_{yz_6}} = F_a F_s \odot X \odot \int_0^X \frac{Z_c}{D_{yz}} dx$	
$U_{b_{z_7}} = F_a^2 \odot Y \odot \int_0^X \frac{-Y_c}{D_z} dx$	$U_{b_{yz_7}} = F_a^2 \odot Y \odot \int_0^X \frac{-Z_c}{D_{yz}} dx$	
$U_{b_{z_8}} = F_s^2 \odot X^2 \odot \int_0^X \frac{1}{2D_z} dx$	$U_{b_{yz_8}} = F_a^2 \odot \int_0^X \frac{Z_c Y_c}{D_{yz}} dx$	
$U_{b_{z_9}} = F_a F_s \odot X \odot Y \odot \int_0^X \frac{-1}{D_z} dx$		
$U_{b_{z_{10}}} = F_a^2 \odot Y^2 \odot \int_0^X \frac{1}{2D_z} dx$		

To summarize, the parameters that may vary with health status are the meshing force $\bar{F}(X_i)$, the coordinates of the cross-section area in the z-y plane (Z_c, Y_c), the z-coordinate of the resultant axial force in the z-x plane $\bar{Z}_i$, and consequently, the associated bending strain energies in all three modes U_{b_z}, $U_{b_{yz}}$, U_{b_y}. **Table 5** outlines the parameters subject to health status variation and those that remain constant or zero according to the nominal healthy status.

Table 5 - Key parameters and their health status-related dependency in bending strain energy

Health status	Mesh force $\bar{F}(X_i)$	Center of area		Z-center of resultant force $\bar{Z}_i$	Bending strain energy		
		Y_c	Z_c		U_{b_z}	$U_{b_{yz}}$	U_{b_y}
'Healthy'	Constant	0	0	0	Variable	0	0
'ToothBreakage'	Constant	0	Variable	Variable	Variable	0	Variable
'PartialFaceFault'	Constant	Variable	Variable	Variable	Variable	Variable	Variable
'ThroughFaceFault'	Variable	Variable	0	0	Variable	0	0
'ToothDestruction'	Constant	Variable	0	0	Variable	0	0

calcInvFilletFoundationStiff

This function calculates the inverse stiffness of the residual stress induced by the fillet foundation. The inverse stiffness is empirically calculated by dividing the displacement of the tooth by the contact force. The stiffness calculation is based on the work of Sainsot et al. [1]. The main concept, as illustrated in **Figure 35**, is that the force applied to the tooth causes a deflection in the tooth's foundation. The relationship between the force and the deflection is used to determine the corresponding stiffness. The foundation deflection can be calculated using the following equation:

$$\delta_f = \frac{F\cos^2\alpha}{EW}\left[L(h,\theta_f)\cdot\left(\frac{u}{S_f}\right)^2 + M(h,\theta_f)\cdot\frac{u}{S_f} + P(h,\theta_f)\cdot(1+Q(h,\theta_f)\cdot\tan^2\alpha)\right]$$

Here, L, M, P, and Q are polynomial functions represented as:

$$X(h,\theta_f) = \frac{A_i}{\theta_f^2} + B_i h^2 + \frac{C_i h}{\theta_f} + \frac{D_i}{\theta_f} + E_i h + F_i$$

The coefficients $A_i, B_i, C_i, D_i, E_i, F_i$ are empirical coefficients obtained from previous research [1] and are listed in **Table 6**.

Table 6 - Coefficients of the polynomial functions

	A_i	B_i	C_i	D_i	E_i	F_i
L	$-5.574\cdot10^{-5}$	$-1.9986\cdot10^{-3}$	$-2.3015\cdot10^{-4}$	$4.7702\cdot10^{-3}$	0.0271	6.8045
M	$60\cdot111^{-5}$	$28.100\cdot10^{-3}$	$-83.431\cdot10^{-4}$	$-9.9256\cdot10^{-3}$	0.1624	0.9086
P	$-50.952\cdot10^{-5}$	$185.50\cdot10^{-3}$	$0.0538\cdot10^{-4}$	$53.300\cdot10^{-3}$	0.2895	0.9236
Q	$-6.2042\cdot10^{-5}$	$9.0889\cdot10^{-3}$	$-4.0964\cdot10^{-4}$	$-7.8297\cdot10^{-3}$	−0.1472	0.6904

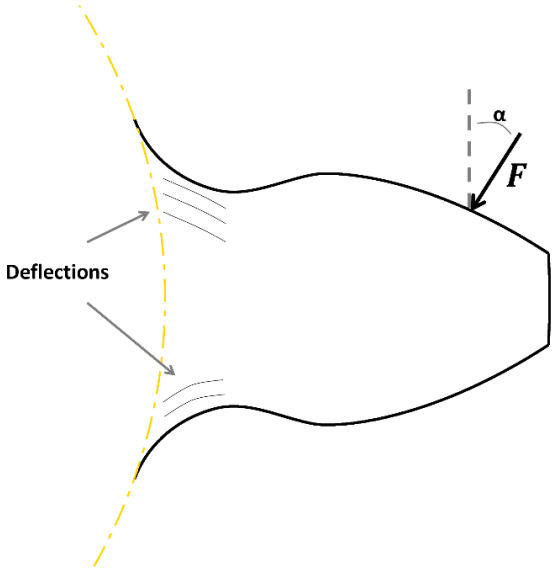

Figure 35 - Deflections in the foundation of the tooth due to the force

`calcInvHertzianContStiff`

This function calculates the inverse stiffness derived from the elastic force induced by the Hertzian contact stresses. Contact mechanics is a field within mechanical engineering that examines how solid objects behave when they come into contact. It explores how components such as gears and bearings interact when their surfaces meet. A key concept in contact mechanics is Hertzian contact, which deals with the stress that occurs when curved surfaces touch and slightly deform under pressure. This stress depends on factors such as the force pressing them together, the curvature of the surfaces, and the materials they're made of.

The model considers Hertzian contact stress when calculating the equivalent stiffness of a tooth pair, following Yang and Sun's paper published in 1985 []. They derived an analytic formula for spur gear elastic force, treating gear teeth contact as two cylinders with parallel axes under compressive force, as illustrated in **Figure 36**. Using principles of contact mechanics and geometry, they approximate the elastic force as follows:

$$F = \frac{\pi \cdot E \cdot W(x)}{4(1-\nu^2)} \cdot \delta$$

Where F is the meshing force, $W(x)$ is the tooth width, E is the modulus of elasticity, ν is Poisson's ratio, and δ is the deflection (or indentation) due to Hertzian contact stress.

Hertzian contact between two cylinders with parallel axes

Contact between a tooth pair in spur gears

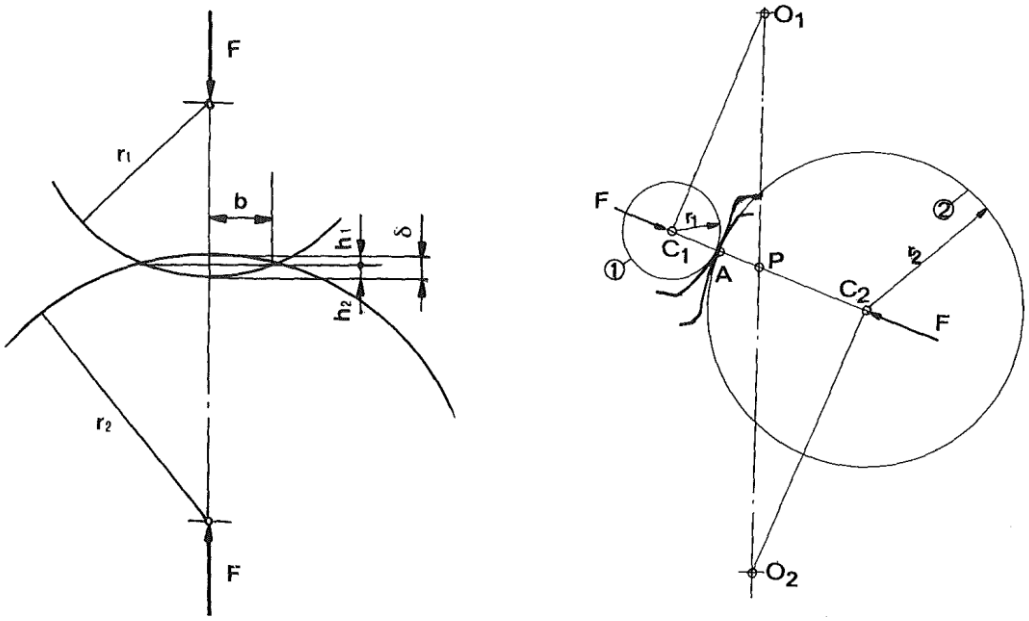

Figure 36 - Hertzian contact analysis between two cylinders with parallel axes (left); Contact between a tooth pair is spur gears (right) [5]

The inverse Hertzian contact stiffness $k_h^{-1}(x)$ can be approximated as the inverse of the ratio between the compressive meshing force and the deflection as follows:

$$k_h(x) = \frac{F}{\delta} = \frac{\pi \cdot E \cdot W(x)}{4(1-\nu^2)}$$

$$\rightarrow k_h^{-1}(x) = \frac{4(1-\nu^2)}{\pi \cdot E \cdot W(x)}$$

The only variable parameter in the inverse Hertzian stiffness is the tooth width $W(x)$, while E and ν remain constant and independent of x for all health statuses. **Table 7** provides an overview of how $W(x)$ and, consequently, $k_h^{-1}(x)$, depend on the health status.

Table 7 - Dependency of the Hertzian contact stress on tooth width for different health statuses.

Health status	Tooth width $W(x)$
`'Healthy'`	Constant
`'ToothBreakage'`	Variable
`'PartialFaceFault'`	Variable
`'ThroughFaceFault'`	Constant
`'ToothDestruction'`	Constant

correctUForThroughFaceFault

This function adjusts the terms of potential strain energies attributed to axial (a), shear (s), and bending (b) stresses, specifically addressing the variation in pressure angle (α) and geometry caused by through tooth face faults. The general expression for the strain energy term $U_n(X)$ can be stated as follows:

$$U_n(X) = F_1F_2(\alpha) \odot \rho(X) \odot \int_0^X \phi(x)dx \quad (n = a, s, b)$$

where $F_1F_2(\alpha)$ stands for combinations of meshing force components, i.e., axial-axial (F_a^2), axial-shear (F_aF_s), or shear-shear (F_s^2), all are α-dependent. $\rho(X)$ represents geometrical terms related to the radius to the meshing force, typically equaling 1 except for specific bending strain energy terms listed in **Table 3**. The function $\phi(x)$ denotes x-dependent material and geometric terms within integration boundaries. For more details, please see `calcInv(Axial/Bending/Shear)Stiff`.

In modeling through tooth face faults, we assume that the fault edges on the damaged wheel profile serve as rotating axes in two stages: initiation-transition and transition-exit, as depicted in **Figure 3**. In the initial stage, the undamaged wheel engages the starting edge of the fault and rotates along it until a transition occurs in the middle of the fault. In the second stage, the undamaged wheel rotates along the ending edge of the fault until it exits. Therefore, any geometric terms in the potential energy vector (including $\rho(X)$ and $\phi(x)$) should adopt the value of the relevant fault edge within the fault boundaries while maintaining the variable pressure angle $\alpha(X)$ in the term $F_1F_2(\alpha)$. To achieve this, the potential strain energy is adjusted in the following manner:

1. Overwrite the original U_{fault} with the value of the proper fault edge U_{edge} for all fault indices.
2. Retain the initially calculated value of the α-dependent term $F_1F_2(\alpha)$ by

$$U_{fault} = U_{edge} \times \frac{F_1F_{2_{fault}}}{F_1F_{2_{edge}}}$$

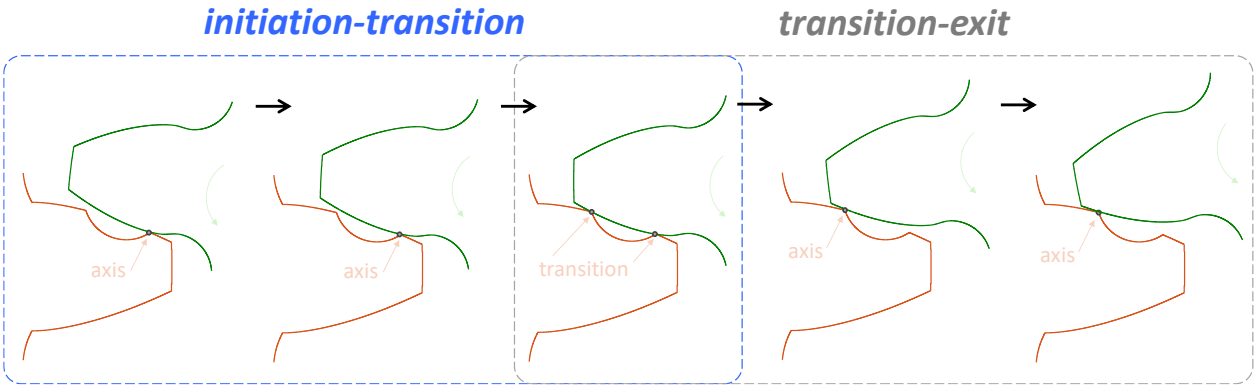

Figure 37 - Interaction of the healthy tooth with the Through Face Fault

`calcLinspaceMtx`

This function serves as an extension to the existing `linspace` function in MATLAB. Its purpose is to construct a matrix in which each row comprises a vector created with linear spacing between the corresponding values in the 'starts' and 'ends' vectors. Users are required to provide two input vectors: 'starts,' a vector containing starting values, and 'ends,' a vector containing ending values. Additionally, users specify the number of points they wish to generate for each range.

An application of this function in the dynamic model is illustrated in **Figure 38**. Here, a linearly spaced vector in the y-axis is generated for each point along the tooth profile in the x-axis, spanning from the inactive to the operational tooth face.

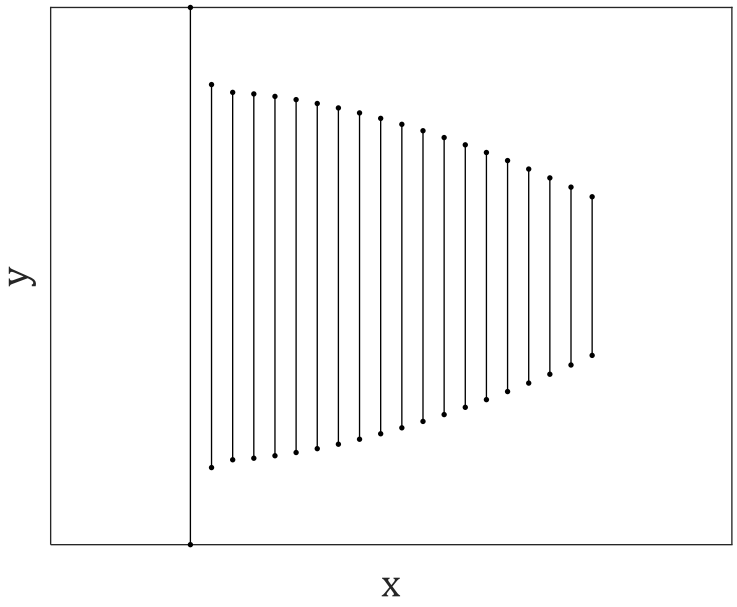

Figure 38 - An illustration of linearly spaced vectors in y for each x-point along the tooth profile

`calcEulerLagrangeComponents`

This function utilizes the Euler-Lagrange model to calculate the components of the equations of motion describing the simulated system presented in **Figure 39**, including the mass, damping, and stiffness matrices, and the external forces vector. The Euler-Lagrange model is expressed as follows:

$$M\ddot{u} + C\dot{u} + K(u)u = F_{ex}$$

$$u = \{x_{out} \quad y_{out} \quad x_{in} \quad y_{in} \quad \theta_{out} \quad \theta_{in} \quad \theta_b \quad z_{out} \quad z_{in} \quad \varphi_{out} \quad \varphi_{in} \quad \psi_{out} \quad \psi_{in}\}^T$$

Where u is the vector of generalized coordinates, M is the mass matrix, C is the damping matrix, $K(u)$ is the non-linear stiffness matrix, and F_{ex} is the external force vector. The block diagram in **Figure 40** demonstrates the methodology of this function.

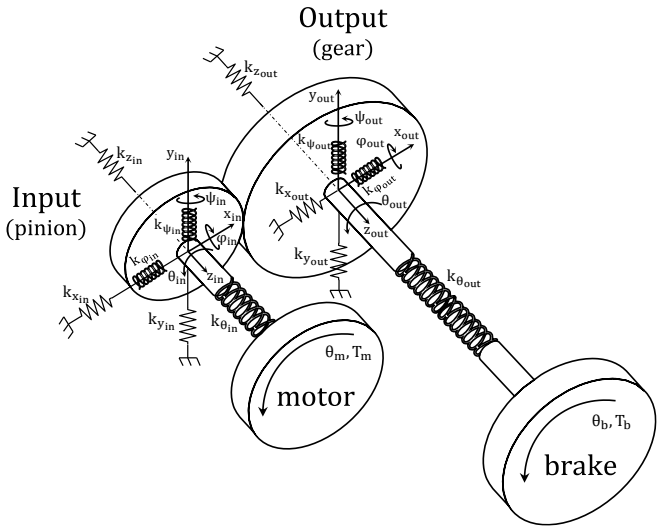

Figure 39 – Qualitative sketch of the simulated system

Variable speed handling:

The function is able to generate a synthetic profile of the rotational speed of the input shaft on demand, taking into account the nominal rps and the statistical error associated with the nominal value. The time-varying cycle of the motor is obtained by performing a cumulative integration on the fluctuating speed signal over time, as depicted in **Figure 40**. The rps signal is included in the output structure of the Euler-Lagrange components, allowing for additional post-processing capabilities such as angular resampling.

Calculating the constant diagonal mass matrix M:

The mass matrix is a square symmetrical matrix that describes the relationship between the acceleration forces and the system. When multiple masses are connected at a rigid contact point, it can result in mass coupling, which is reflected by non-zero off-diagonal elements in the mass matrix. However, the contact between the wheels is assumed to be elastic. In the case of elastic contact between the wheels, the interaction can be represented as a series of springs connecting the wheels. Therefore, no mass coupling is anticipated, and the mass matrix can be represented as a diagonal matrix, as shown below:

$$M = \mathrm{diag}\left(m_{out}, m_{out}, m_{in}, m_{in}, J_{out}, J_{in}, J_b, m_{out}, m_{in}, I_{rot_{out}}, I_{rot_{in}}, I_{rot_{out}}, I_{rot_{in}}\right)$$

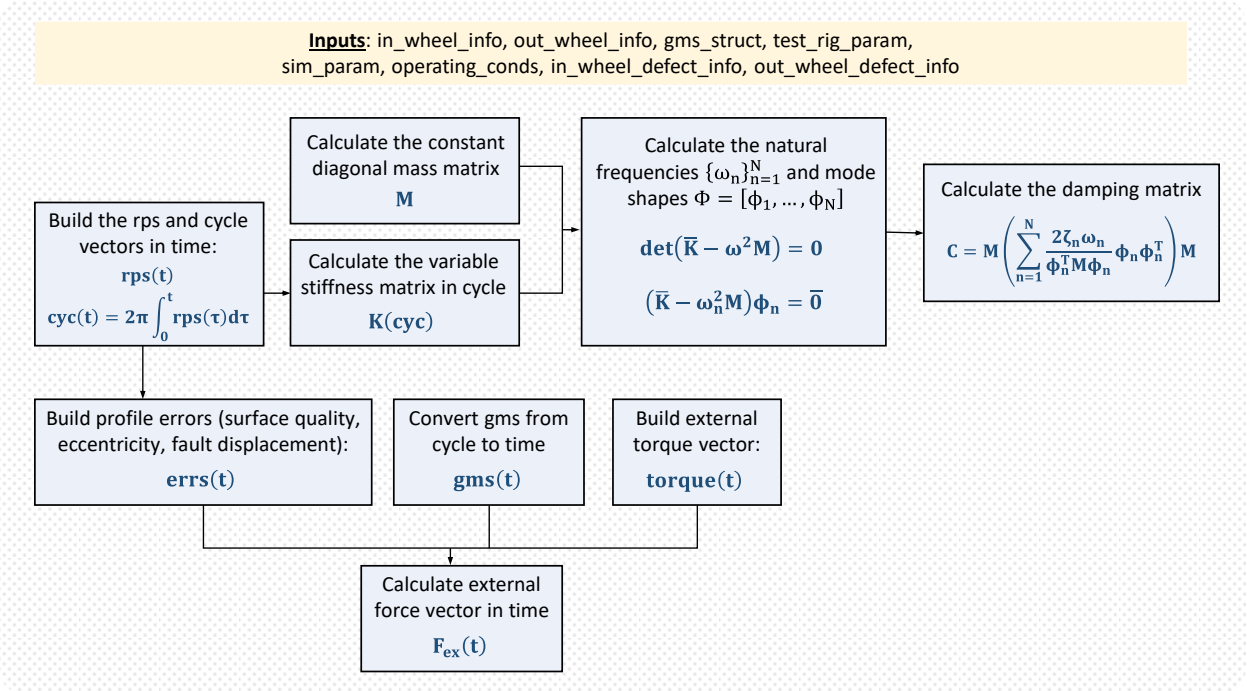

Figure 40 - Methodology of calculating the Euler-Lagrange components.

Calculating K matrix in cycle instead of time:

Calculating the K matrix in the cycle domain rather than the time domain offers a significant advantage in terms of time consumption. By performing the calculations in the cycle domain, the code optimizes efficiency and reduces computational resources. This approach eliminates the need for repetitive time iterations, allowing for faster and more streamlined processing. The cycle-based calculations enable accurate results within a shorter timeframe, enhancing productivity and enabling more extensive simulations and analysis. Later in the numerical solution, a dedicated function is used to map the index of the K matrix in the cycle domain to the relevant time step, ensuring accurate synchronization between cycle and time.

Calculating the damping matrix C:

The damping matrix C is calculated under the assumption that each mode of vibration is characterized by a constant damping ratio of $\{\zeta_n\}_{n=1}^{N}$. The calculation of the damping matrix follows the same procedure introduced in Chopra's book *'Dynamics of Structures, Theory and Applications to Earthquake Engineering, 4th edition'* (2013, p.462). First, the modal matrix $\Phi = [\phi_1 | \dots | \phi_N]$ and natural frequencies $\{\omega_n\}_{n=1}^{N}$ are obtained by formulating and solving an eigenvalues and eigenvectors problem of the mass matrix M and average stiffness matrix $\bar{K}$ as follows:

$$\bar{K}\phi_n = \omega_n^2 M\phi_n$$

The orthogonality of the natural modes $\{\phi_n\}_{n=1}^{N}$ ensures that the modal mass matrix $\hat{M}$, modal damping matrix $\hat{C}$, and modal stiffness matrix $\hat{K}$ are diagonal. The modal matrices are calculated as follows:

$$\widehat{M} \equiv \Phi^T M \Phi = \mathrm{diag}\{\widehat{m}_n\}_{n=1}^N \qquad \widehat{C} \equiv \Phi^T C \Phi = \mathrm{diag}\{\widehat{c}_n\}_{n=1}^N \qquad \widehat{K} \equiv \Phi^T \overline{K} \Phi = \mathrm{diag}\{\widehat{k}_n\}_{n=1}^N$$

Where the modal mass $\widehat{m}_n$, damping $\widehat{c}_n$, and stiffness $\widehat{k}_n$ of each modal coordinate can be expressed as:

$$\widehat{m}_n = \phi_n^T M \phi_n \qquad \widehat{c}_n = 2\xi_n \omega_n \widehat{m}_n \qquad \widehat{k}_n = \omega_n^2 \widehat{m}_n$$

The damping matrix C can be rewritten as:

$$C = \left(\Phi^T\right)^{-1} \widehat{C} \Phi^{-1}$$

In order to save computational time required for the inversion of two full square matrices, the following relations, obtained by the orthogonality relationship, are used:

$$\Phi^{-1} = \widehat{M}^{-1} \Phi^T M \qquad \left(\Phi^T\right)^{-1} = M \Phi \widehat{M}^{-1}$$

Since $\widehat{M}$ is diagonal, its inversion is significantly simpler. By substituting these relations, and utilizing that the modal matrices are diagonal, the following formula for calculating the damping matrix is obtained:

$$C = M\left(\sum_{n=1}^{N} \frac{2\zeta_n \omega_n}{\phi_n^T M \phi_n} \phi_n \phi_n^T\right) M$$

Profile errors:

The external force vector comprises three components: the torque applied by the brake, the reaction torque resulting from the motor's angular motion and the elasticity of the shaft, and the mesh forces (F_{mesh}). The formula for determining the mesh forces is provided below, which involves multiplying the gear mesh stiffness ($gms(t)$) by the profile errors ($errs(t)$). These profile errors serve as a displacement input and encompass various factors such as manufacturing surface quality errors of the input and output shafts, eccentricity errors, and displacements caused by faults that distort the nominal line of action. Each error is calculated independently with respect to time and subsequently summed together.

$$errs(t) = surf\_quality\_errs + eccentricity\_errs + fault\_displacement$$

$$F_{mesh} = gms(t) \cdot errs(t)$$

`calcVariableStiffMtx`

This function calculates the non-linear stiffness matrix as a list of square matrices in the cycle. Calculating the K matrix in the cycle domain rather than the time domain offers a significant advantage in terms of time consumption. By performing the calculations in the cycle domain, the code optimizes efficiency and reduces computational resources. This approach eliminates the need for repetitive time iterations, allowing for faster and more streamlined processing. The cycle-based calculations enable accurate results within a shorter timeframe, enhancing productivity and enabling more extensive simulations and analysis. Later in the numerical solution, a dedicated function is used to map the index of the K matrix in the cycle domain to the relevant time step, ensuring accurate synchronization between cycle and time.

The stiffness matrix, as written below, is the sum of a constant structural stiffness matrix $\mathrm{K}_{\mathrm{const}}$ and a stiffness matrix $\mathrm{K}_{\mathrm{var}}$ which is variable in cycle due to the non-linear gear mesh stiffness $\mathrm{gms}(\mathrm{cyc})$:

$$\mathrm{K}(\mathrm{cyc}) = \mathrm{K}_{\mathrm{const}} + \mathrm{K}_{\mathrm{var}}(\mathrm{cyc})$$

The constant stiffness matrix $\mathbf{K_{const}}$:

The constant stiffness matrix does not account for the non-linear interaction between the wheels along the line of action. As a result, it is almost entirely diagonal due to the majority of springs connecting each wheel to the ground, where no coupling is expected, as depicted in **Figure 39**. Additionally, the torsional shaft between the input wheel and the motor does not introduce any coupling since the angular position of the motor is known and does not serve as a degree of freedom. However, the torsional shaft between the output wheel and the brake does generate a coupling between the angular positions of θ_{out} and θ_{b}.

$$\mathrm{K}_{\mathrm{const}_{\mathrm{i,j}}} = \begin{cases} +\mathrm{k}_{\mathrm{n,n}} & \mathrm{i} = \mathrm{j} = \mathrm{n} \\ -\mathrm{k}_{\mathrm{tor}_{\mathrm{out}}} & (\mathrm{i},\mathrm{j}) = (5{,}7),(7{,}5) \\ 0 & \text{else} \end{cases}$$

$$\left\{\mathrm{k}_{\mathrm{n,n}}\right\}_{\mathrm{n}=1}^{\mathrm{N}} = \left\{\mathrm{k}_{\mathrm{bear}}, \mathrm{k}_{\mathrm{bear}}, \mathrm{k}_{\mathrm{bear}}, \mathrm{k}_{\mathrm{bear}}, \mathrm{k}_{\mathrm{tor}_{\mathrm{out}}}, \mathrm{k}_{\mathrm{tor}_{\mathrm{in}}}, \mathrm{k}_{\mathrm{tor}_{\mathrm{out}}}, \mathrm{k}_{\mathrm{bear}}, \mathrm{k}_{\mathrm{bear}}, \mathrm{k}_{\mathrm{rot}_{\mathrm{out}}}, \mathrm{k}_{\mathrm{rot}_{\mathrm{in}}}, \mathrm{k}_{\mathrm{rot}_{\mathrm{out}}}, \mathrm{k}_{\mathrm{rot}_{\mathrm{in}}}\right\}$$

Where $\mathrm{k}_{\mathrm{bear}}$ is the linear bearing stiffness, $\mathrm{k}_{\mathrm{tor}}$ is the torsional shaft stiffness (in/out), and $\mathrm{k}_{\mathrm{rot}}$ is the rotational stiffness of the wheel (in/out).

The variable stiffness matrix $\mathbf{K_{var}}$:

The variable stiffness matrix is a list of $\mathrm{N} \times \mathrm{N} \times \mathrm{len\_cyc}$ stiffness matrices in cycle $\left\{\mathrm{K}_{\mathrm{var}_{\mathrm{i}}}\right\}_{\mathrm{i}=1}^{\mathrm{len\_cyc}}$ which accounts only for the non-linear interaction between the wheels along the line of action. The variable stiffness matrix list is obtained by multiplying the gear mesh stiffness in cycle ($\mathrm{gms}(\mathrm{cyc})$) by a list of geometrical coefficient matrices ($\mathrm{geom}(\mathrm{cyc})$). The relationship can be expressed as follows:

$$\left\{\mathrm{K}_{\mathrm{var}_{\mathrm{i}}}\right\}_{\mathrm{i}=1}^{\mathrm{len\_cyc}} = \mathrm{gms}(\mathrm{cyc}_{\mathrm{i}}) \cdot \mathrm{geom}(\mathrm{cyc}_{\mathrm{i}})$$

The matrix of geometrical coefficient is separated into z-dependent and z-independent matrices, as follows:

$$\mathrm{geom}(\mathrm{cyc}) = \mathrm{geom}_{\mathrm{z\_dep}}\left(\{\mathrm{f}_{\mathrm{i}}\}_{\mathrm{i}=1}^{24}\right) \cdot \mathrm{E}(\mathrm{z}) + \mathrm{geom}_{\mathrm{z\_indep}}\left(\{\mathrm{f}_{\mathrm{i}}\}_{\mathrm{i}=1}^{24}\right)$$

The geometric terms $\{f_i\}_{i=1}^{24}$ are dependent on the pressure angle α, the base helix angle β, and the base radius of the wheel R_b. These terms encapsulating the geometrical characteristics are summarized in **Table 8**.

Table 8 - Geometric terms in the stiffness matrix

$f_1 = -R_{b_{out}} \sin(\alpha) \sin(\beta) + \cos(\alpha)\, \ell_{out} \sin(\beta)$	$f_{13} = \cos(\alpha) \cos(\beta)$
$f_2 = -R_{b_{out}} \cos(\alpha) \sin(\beta) - \sin(\alpha)\, \ell_{out} \sin(\beta)$	$f_{14} = R_{b_{out}} \cos(\alpha) \cos^2(\beta)$
$f_3 = -R_{b_{in}} \sin(\alpha) \sin(\beta) + \cos(\alpha)\, \ell_{in} \sin(\beta)$	$f_{15} = R_{b_{in}} \cos(\alpha) \cos^2(\beta)$
$f_4 = -R_{b_{in}} \cos(\alpha) \sin(\beta) - \sin(\alpha)\, \ell_{in} \sin(\beta)$	$f_{16} = R_{b_{out}} \cos(\beta) \sin(\beta)$
$f_5 = \cos^2(\beta) \sin^2(\alpha)$	$f_{17} = R_{b_{in}} \cos(\beta) \sin(\beta)$
$f_6 = \cos(\alpha) \cos^2(\beta) \sin(\alpha)$	$f_{18} = R_{b_{out}} \cos(\beta)$
$f_7 = \cos(\beta) \sin(\alpha) \sin(\beta)$	$f_{19} = R_{b_{in}} \cos(\beta)$
$f_8 = \cos(\beta) \sin(\alpha)$	$f_{20} = R_{b_{out}}^2 \cos^2(\beta)$
$f_9 = R_{b_{out}} \cos^2(\beta) \sin(\alpha)$	$f_{21} = R_{b_{out}} R_{b_{in}} \cos^2(\beta)$
$f_{10} = R_{b_{in}} \cos^2(\beta) \sin(\alpha)$	$f_{22} = R_{b_{in}}^2 \cos^2(\beta)$
$f_{11} = \cos^2(\alpha) \cos^2(\beta)$	$f_{23} = -\cos(\alpha)$
$f_{12} = \cos(\alpha) \cos(\beta) \sin(\beta)$	$f_{24} = +\sin(\alpha)$

The matrix $E(z)$ represents a list of expectation matrices in cycle, encompassing both the mean and squared mean values for each cycle point. The matrix is constructed using the following procedure:

$$E(z) = \begin{bmatrix} 0_{9\times9} & \mathbb{E}[z] \cdot 1_{9\times4} \\ \mathbb{E}[z] \cdot 1_{4\times9} & \mathbb{E}[z^2] \cdot 1_{4\times4} \end{bmatrix}$$

The structure of the expectation matrix can be explained as follows. The top left 9×9 matrix corresponds to the degrees of freedom that are not influenced by the position along the axial direction, including any linear displacement or axial rotation. Since these coordinates are not affected by the axial displacement, a matrix of zeros is obtained. On the other hand, the bottom right 4×4 matrix represents the degrees of freedom that are affected by the axial position. These coordinates are angular in nature, and the units of the stiffness exhibit a quadratic relationship with the distance along the z-axis. The top right 9×4 or bottom left 4×9 matrices are identical, indicating the stiffness coupling between the affected four angular coordinates and the linear motion or axial rotation coordinates. This coupling reflects the relationship between the angular displacements and the linear motions. The units of the geometry are also consistent with this coupling, ensuring compatibility between the different degrees of freedom.

The coordinate z_{max} depends on the nominal tooth width W and the defect width W_{def}. In case of a tooth breakage, the defect width is described by a linear line from the maximal width and decreasing to zero. In case of a partial tooth face fault the defect width remains constant. Recall that the gear mesh stiffness is assumed to be distributed uniformly along the tooth width, and therefore:

$$z_{min} = -0.5W \qquad \mathbb{E}[z] = \frac{1}{z_{max} - z_{min}} \int_{z_{min}}^{z_{max}} z\,dz = \frac{z_{max} + z_{min}}{2}$$

$$z_{max} = +0.5W - W_{def}(t) \qquad \mathbb{E}[z^2] = \frac{1}{z_{max} - z_{min}} \int_{z_{min}}^{z_{max}} z^2\,dz = \frac{z_{max}^2 + z_{min} z_{max} + z_{min}^2}{3}$$

`calcMeshForcesT`

This function is designed to compute mesh forces over time, and subsequently integrating them into the calculation of the generalized external forces vector (refer to `calcForcesVctrT`). The underlying assumption is that mesh forces arise from involute distortions, specifically profile errors, encompassing surface quality imperfections and fault displacements, all scaled by the gear mesh stiffness. A critical complexity arises from the contact ratio (ε), which necessitates a meticulous consideration of the intricate overlap among meshing tooth pairs along the line of action. This intricacy presents a significant challenge in calculating mesh forces, given that each tooth pair may exhibit distinct stiffness characteristics and unique profile errors. To address this challenge, this function divides the gear mesh stiffness into a matrix representing equivalent stiffness for each tooth pair, whether healthy or faulty. Subsequently, it multiplies each stiffness by its corresponding profile errors. The resultant mesh forces for each tooth pair are then meticulously concatenated in accordance with the contact ratio, ensuring a comprehensive and accurate representation of the forces involved.

The computation of mesh forces over time follows a four-step procedure:

1. Building `KeqMtx`: This initial step involves constructing the equivalent stiffness matrix for all tooth pairs. This matrix draws information from the function `calcGearMeshStiff`. It is formed by replicating the equivalent stiffness of a healthy tooth pair multiple times, aligning with the gear mesh stiffness signal. Additionally, it considers the scenario of a single-tooth fault.
2. Interpolating the profile errors matrix to match the finer resolution of the equivalent stiffness. Recall that the equivalent stiffness of each pair spans ε gear mesh cycles from engagement to separation.
3. Calculating mesh forces iteratively by scaling the profile errors of each tooth pair by its corresponding stiffness. Then, the results are concatenated in the cycle domain properly, as illustrated in **Figure 41**. In the case of a tooth fault, both a unique equivalent stiffness and fault displacement are present. It is crucial to ensure that only the faulty stiffness is multiplied by the corresponding fault displacement. This distinction is essential to prevent contributions from healthy pairs to the overall mesh forces.
4. Converting the mesh forces signal from the cycle domain to the time domain.

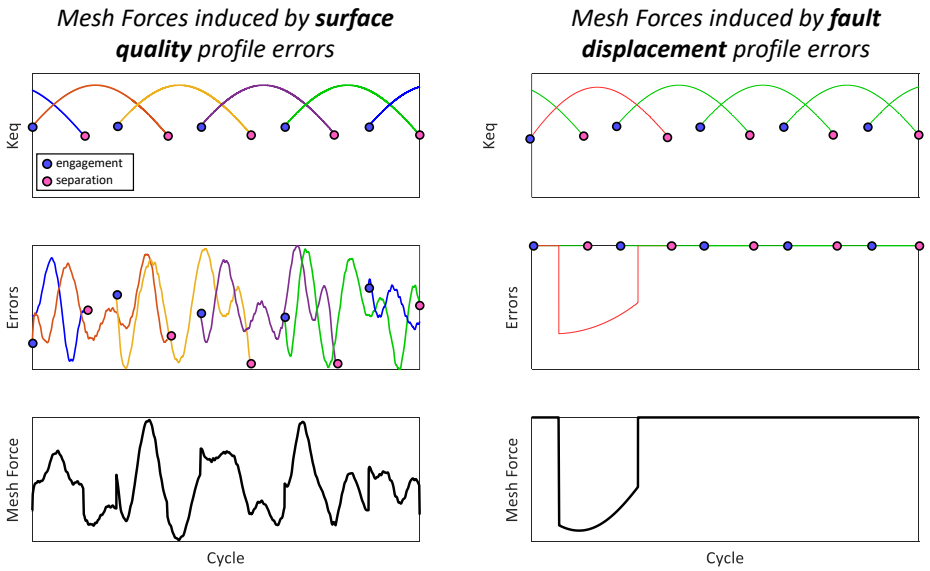

Figure 41 - Illustration of the iterative procedure of computing the mesh forces in cycle in two cases: left - surface quality, and right - fault displacement.

calcForcesVctrT

This function calculates the matrix of the external forces vector in time. The external forces comprise of the torque applied by the brake $\overline{T}_b$, the reaction torque resulting from the motor's angular motion and the elasticity of the shaft $\overline{T}_m$, and the mesh forces $\overline{F}_{mesh}$, as follows:

$$\overline{F}_{ex}(t) = \left\{F_{ex_n}(t)\right\}_{n=1}^{N} = \overline{F}_{mesh} + \overline{T}_m + \overline{T}_b$$

Recall that the vector of generalized coordinates u can be written as follows:

$$u = \begin{Bmatrix} x_{out} \\ y_{out} \\ x_{in} \\ y_{in} \\ \theta_{out} \\ \theta_{in} \\ \theta_b \\ z_{out} \\ z_{in} \\ \varphi_{out} \\ \varphi_{in} \\ \psi_{out} \\ \psi_{in} \end{Bmatrix}$$

Torques applied by the motor and brake:

The torques applied by the motor and the brake are associated with the angular displacements of the input shaft (θ_{in}) and the brake (θ_b), respectively. The brake torque signal is a user-defined operational parameter, while the motor torque is determined by multiplying the polar stiffness of the input shaft (k_{shaft}) by the motor's cycle in time $cyc_m(t)$ as follows:

$$T_m(t) = k_{shaft} \cdot cyc_m(t)$$

$$\overline{T}_m + \overline{T}_b = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ T_m(t) \\ T_b(t) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

Mesh Forces:

The mesh force $\bar{F}_{mesh} = \{F_{mesh_n}(t)\}_{n=1}^{N}$ can be calculated by multiplying the gear mesh stiffness $gms(t)$ with the profile errors $errs(t)$, and then further multiplied by a geometrical coefficient specific to each coordinate, denoted as $\{geom_n(t)\}_{n=1}^{N}$. The relationship can be expressed as follows:

$$F_{mesh_n}(t) = gms(t) \cdot errs(t) \cdot geom_n(t)$$

$$geom(t) = \{geom_n(t)\}_{n=1}^{N} = \begin{Bmatrix} -\cos(\beta)\sin(\alpha) \\ -\cos(\beta)\cos(\alpha) \\ +\cos(\beta)\sin(\alpha) \\ +\cos(\beta)\cos(\alpha) \\ -\cos(\beta)\,R_{b_{out}} \\ -\cos(\beta)\,R_{b_{in}} \\ 0 \\ -\sin(\beta) \\ +\sin(\beta) \\ +\cos(\alpha)\,\bar{z}(t) \\ -\cos(\alpha)\,\bar{z}(t) \\ +\sin(\beta)\,R_{p_{out}} - \sin(\alpha)\,\mathbb{E}[z] \\ +\sin(\beta)\,R_{p_{in}} + \sin(\alpha)\,\mathbb{E}[z] \end{Bmatrix}$$

where α, β are the pressure angle and base helix angle, respectively. R_b and R_p are, respectively, the base and pitch radii. $\mathbb{E}[z]$ is the expected z-coordinate of the tooth width. Recall that the z-axis corresponds to the axial direction. Unlike the geometric terms of the first nine coordinates, which are independent of z, the last four coordinates have a specific z-axis dependence, as depicted in **Figure 42**.

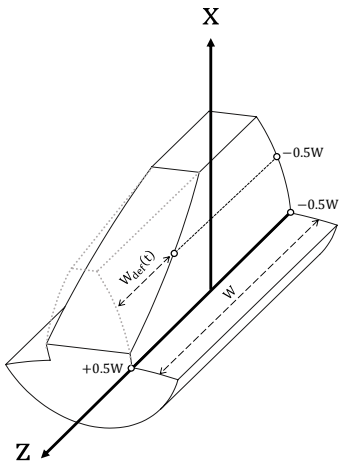

Figure 42 - Illustration of z-axis dependency in a tooth breakage scenario

Dependency on z-axis:

The last four coordinates represent the angular positions in the longitudinal and transverse directions, incorporating a geometric term that depends on the z-axis. This dependency is justified by the fact that the distance along the z-axis from mesh forces in the x-y plane generates moments in the mentioned

directions. The term $\mathbb{E}[z]$ represents the mean coordinate along the tooth width and can be calculated as follows (see **Figure 42**):

$$z_{min} = -0.5W \qquad z_{max} = +0.5W - W_{def}(t)$$

$$\mathbb{E}[z](t) = \frac{z_{min} + z_{max}}{2} = -\frac{W_{def}(t)}{2}$$

In a healthy scenario, $W_{def} = 0$ and consequently $\mathbb{E}[z] = 0$. However, it is important to note that the tooth width represents the contact surface, and certain deviations resulting from faults, such as breakage or tooth face faults, can disrupt the symmetry along the tooth width and then $\mathbb{E}[z] \neq 0$.

convertCyc2Time

This function takes any structure (vector/matrix/list) denoted as $x(cyc)$ in the cycle domain and returns the equivalent representation of $x(cyc)$ (or a portion thereof) in the time domain, denotes as $x(t)$. In the cycle domain, $x(cyc)$ exhibits periodic behavior with respect to the cycle and corresponds to the cycle vector, with a resolution of Δcyc, while the time resolution of $x(t)$ in the time domain is Δt.

Figure 43 provides a qualitative illustration of the process of converting $x(cyc)$ to $x(t)$, demonstrating the essence of this function. In the provided example, the signal $x(cyc)$ describes a single period of a simple sine wave, sampled at a resolution of $\Delta cyc = 0.1$ (this coarse resolution is intentionally set to facilitate better comprehension of the function). Additionally, a uniformly rotating speed signal with a value of 1rps is sampled in time at a rate of 250Hz, resulting in a time resolution of $\Delta t = 0.004s$. This sampling is carried out over a time duration of 2s, encompassing two periods of $x(cyc)$.

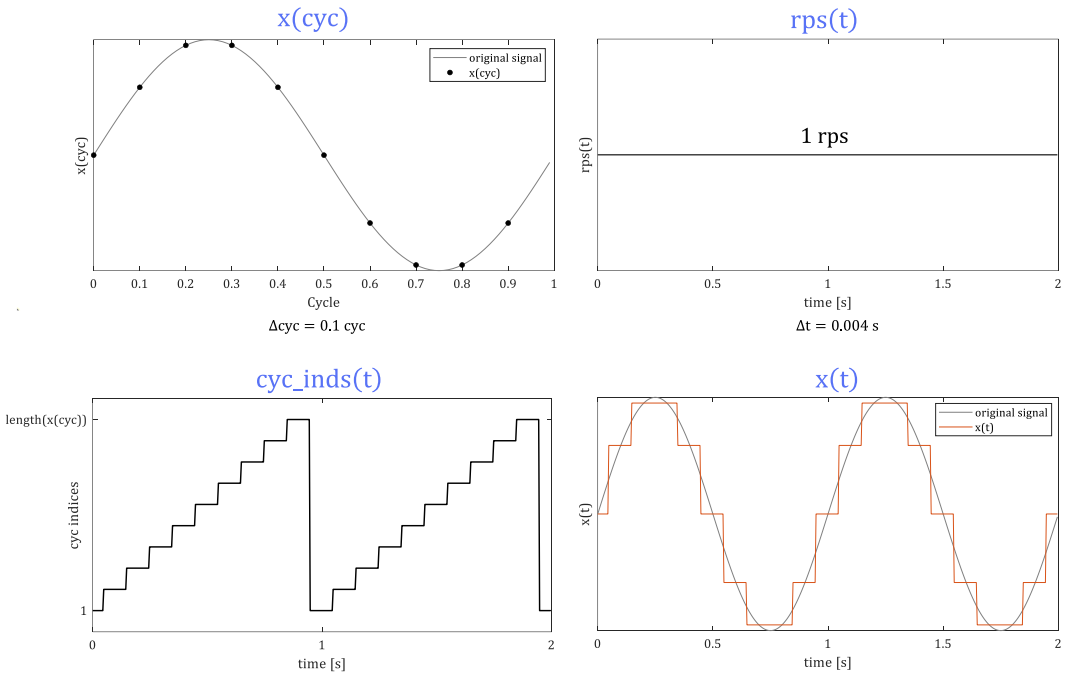

Figure 43 - A qualitative example for the process of converting x(cyc) in the cycle domain to x(t) in the time domain

The function initially computes the cycle vector in the time domain, referred to as $cyc(t)$. This cycle vector is derived from the phase signal $\varphi(t)$, which is normalized by 2π, achieved through cumulative integration utilizing the MATLAB function `cumsum`, as described below:

$$cyc(t) = \frac{\varphi(t)}{2\pi} = \int_0^t rps(\tau) \cdot d\tau$$

The function tackles two specific challenges:

First, there's often a notable difference in resolution between the time domain and the cycle domain, leading to a discretization effect where consecutive time points may share the same cycle index. This effect is visually depicted as the 'step' shape in **Figure 43**. The function addresses this by rounding the

cycle points corresponding to each time step, which are computed by dividing the cycle points in time by the cycle resolution.

Secondly, the desired time duration may encompass more cycles than the cycle duration of $\mathrm{x(cyc)}$. Assuming that $\mathrm{x(cyc)}$ exhibits periodic behavior with respect to the cycle concerning time, this issue is resolved within the function through a modulo operation applied to the cycle points with the length of $\mathrm{x(cyc)}$, as shown in **Figure 43**. To prevent zero indices, which are incompatible with the MATLAB language, the result is incremented by 1. The final formula for the cycle indices corresponding to time is:

$$\mathrm{cycInds} = \mathrm{mod}\left[\mathrm{round}\left(\frac{\mathrm{cyc(t)}}{\Delta\mathrm{cyc}}\right), \mathrm{length}\big(\mathrm{x(cyc)}\big)\right] + 1$$

As previously explained, the function is designed to handle various structures of $\mathrm{x(cyc)}$. Consequently, in the final phase of the function, it generates $\mathrm{x(t)}$ by selecting the appropriate cycle indices that correspond to the time points and adjusting them based on the dimension of $\mathrm{x(cyc)}$.

`resampling`

This function performs resampling to a signal x_{old}, either upsampling or downsampling, depends on the desired number of points Npt_{new} of the resampled signal x_{new}. The resampling is performed with interpolation, by default with spline method but the user has the flexibility to modify the interpolation method.

Figure 2 illustrates a segment displaying the variation in defect width along the x-axis in the event of a partial tooth face fault. The transition between the healthy and damaged states is notably abrupt. It becomes evident that employing `spline` interpolation for resampling in this scenario would introduce significant artifacts. Therefore, for this particular case, the `PCHIP` interpolation method is highly recommended.

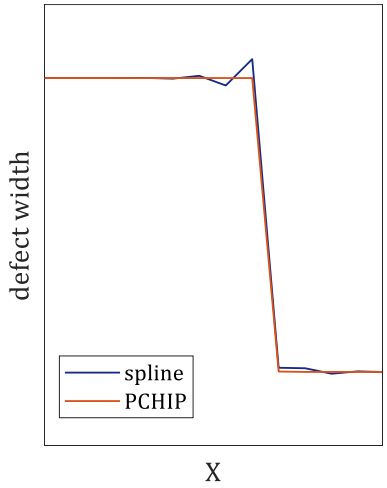

Figure 44 - Effects of the interpolation method on the resampled signal

`uniformRand`

The function generates a numeric array X of any size specified by the user, with values distributed uniformly with desired systematic error and statistic error, as illustrated in **Figure 45** and calculated as follows:

$$\mathrm{X} = \mathrm{systematic\_error} + 2 \times (\mathrm{rand} - 0.5) \times \mathrm{statistic\_error}$$

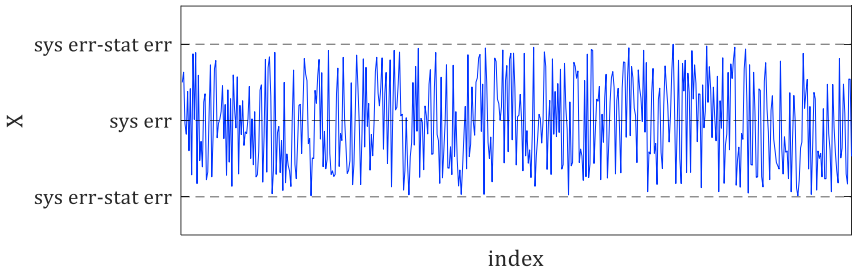

Figure 45 - Uniformly random vector with specified systematic and statistic errors

`numericalSolution`

This function uses the Newmark's method, along with Newton-Raphson iterations, to numerically solve the Euler-Lagrange equations of motion. **Figure 47** illustrates a general block diagram of the numerical solution algorithm. These equations are accompanied by initial conditions and external forces and are expressed as follows:

$$\mathrm{M}\ddot{\mathrm{u}} + \mathrm{C}\dot{\mathrm{u}} + \mathrm{K}(\mathrm{u})\,\mathrm{u} = \mathrm{F}_{\mathrm{ex}}(\mathrm{u})$$

$$\begin{cases} \mathrm{u}(\mathrm{t}=0) = \mathrm{u}_0 \\ \dot{\mathrm{u}}(\mathrm{t}=0) = \dot{\mathrm{u}}_0 \end{cases}$$

Where u, $\dot{\mathrm{u}}$, and $\ddot{\mathrm{u}}$ represent the positions, velocities, and accelerations of the system, M denotes the mass matrix, C represents the damping matrix, $\mathrm{K}(\mathrm{u})$ signifies the non-linear stiffness matrix which depends on the solution u. $\mathrm{F}_{\mathrm{ex}}(\mathrm{u})$ corresponds to the external forces vector, while u_0 and $\dot{\mathrm{u}}_0$ are vectors representing the initial conditions.

The function can be broadly divided into the following stages:

- Preliminary calculations based on Newmark's method.
- Preprocessing of the variable stiffness matrix to improve computational efficiency.
- Iterative solution over time using Newmark's method and Newton-Raphson iterations.
- Postprocessing of the solution to mitigate errors arising from the numerical convergence process.

In the upcoming subsections, we will introduce both numerical methods, discuss computational time optimization, and outline the postprocessing procedure.

Newmark's method

Newmark's method is a numerical technique widely used for solving 2nd-order ordinary differential equations, particularly in the fields of structural dynamics []. It is a common choice in engineering for studying the dynamic behavior of structures under various loads and boundary conditions. This method employs a set of integration algorithms characterized by two parameters, β and γ, which significantly influence the accuracy and stability of the solution. By default, we use β=0.25 and γ=0.5, a common choice for stability-critical problems. However, users have the flexibility to adjust these values. Newmark's method is implicit, taking into consideration acceleration at the current time step during the computations, as follows:

$$\mathrm{u}_{\mathrm{i+1}} = \mathrm{u}_{\mathrm{i}} + \Delta\mathrm{u}_{\mathrm{i}}$$

$$\dot{\mathrm{u}}_{\mathrm{i+1}} = \dot{\mathrm{u}}_{\mathrm{i}} + \frac{\gamma}{\beta\Delta\mathrm{t}}\Delta\mathrm{u}_{\mathrm{i}} - \frac{\gamma}{\beta}\dot{\mathrm{u}}_{\mathrm{i}} + \Delta\mathrm{t}\left(1 - \frac{\gamma}{2\beta}\right)\ddot{\mathrm{u}}_{\mathrm{i}}$$

$$\ddot{\mathrm{u}}_{\mathrm{i+1}} = \ddot{\mathrm{u}}_{\mathrm{i}} + \frac{1}{\beta\Delta\mathrm{t}^2}\Delta\mathrm{u}_{\mathrm{i}} - \frac{1}{\beta\Delta\mathrm{t}}\dot{\mathrm{u}}_{\mathrm{i}} - \frac{1}{2\beta}\ddot{\mathrm{u}}_{\mathrm{i}}$$

At each time step, $\Delta\mathrm{u}_{\mathrm{i}}$ is calculated by defining the modified stiffness matrix $\hat{\mathrm{K}}_{\mathrm{i}}$ and the modified forces difference $\Delta\hat{\mathrm{F}}_{\mathrm{i}}$ (using the specified parameters $\Delta\mathrm{F}_{\mathrm{i}}, \mathrm{a}, \mathrm{b}$) as follows:

$$\Delta\mathrm{u}_{\mathrm{i}} = \hat{\mathrm{K}}_{\mathrm{i}}^{-1}\Delta\hat{\mathrm{F}}_{\mathrm{i}}$$

$$\hat{\mathrm{K}}_{\mathrm{i}} = \mathrm{K}_{\mathrm{i}} + \frac{\gamma}{\beta\Delta\mathrm{t}}\mathrm{C} + \frac{1}{\beta\Delta\mathrm{t}^2}\mathrm{M}$$

$$\Delta\hat{\mathrm{F}}_{\mathrm{i}} = \underbrace{(\mathrm{F}_{\mathrm{i+1}} - \mathrm{F}_{\mathrm{i}})}_{\Delta\mathrm{F}_{\mathrm{i}}} + \underbrace{\left(\frac{1}{\beta\Delta\mathrm{t}}\mathrm{M} + \frac{\gamma}{\beta}\mathrm{C}\right)}_{\mathrm{a}}\dot{\mathrm{u}}_{\mathrm{i}} + \underbrace{\left[\frac{1}{2\beta}\mathrm{M} + \Delta\mathrm{t}\left(\frac{\gamma}{2\beta} - 1\right)\mathrm{C}\right]}_{\mathrm{b}}\ddot{\mathrm{u}}_{\mathrm{i}}$$

We incorporate an extra step in Newmark's method to update the next time step in $F(t)$. This prevents cumulative errors in solving the Euler-Lagrange equations. Without this step, significant errors from a previous time step would persist, causing inaccuracies in the final result. This ensures the solution "forgets" previous errors, eliminating cumulative inaccuracies and enhancing overall accuracy, as follows:

$$F_{i+1} = M\ddot{u}_{i+1} + C\dot{u}_{i+1} + K_{i+1}u_{i+1}$$

More detailed information about the Newmark method can be found in the following video: https://www.youtube.com/watch?v=doVJg4F264I

Newton-Raphson (NR) iterations

The Newton-Raphson (NR) method for systems of equations is a powerful numerical technique used to find the roots or solutions of a system of nonlinear equations $f(P) = 0$. It iteratively refines initial guesses for the unknowns P until it converges to a solution with high accuracy, as following:

$$f(P^n) = 0, P^0 \qquad \Delta P^n = -[J^n]^{-1} f(P^n) \qquad P^{n+1} = P^n + \Delta P^n$$

Where P^n is the solution vector at the n^{th} iteration, and J^n is the Jacobian matrix of $f(P^n)$.

In the numerical solution algorithm, NR iterations refine Newmark's method calculations. Due to the high values of $\hat{K}_i$, direct solving $\Delta u_i = \hat{K}_i^{-1}\Delta\hat{F}_i$ can introduce inaccuracies. In this case, the NR method computes a solution P_i^n for the equation along with its associated Jacobian matrix:

$$P_i^n = \Delta u_i^n \qquad f(P_i^n) = \Delta\hat{F}_i - \hat{K}_i^n P_i^n = 0 \qquad J_i^n = -\hat{K}_i^n$$

The index i iterates over time, while the index n iterates over NR iterations. These iterations aim to minimize $f(P_i^n)$ towards zero. The process begins with an initial guess of zero ($P_i^0 = \bar{0}$), which is then refined through iterations until the agreed convergence criterion is met or the predefined maximum number of iterations is reached. The error is a scalar value that accounts for variations in both the residual and the solution, calculated as follows (using $\odot$ for scalar multiplication):

$$err = \left| \frac{f(P_i^{n+1}) \odot \Delta P_i^n}{f(P_i^0) \odot P_i^{n+1}} \right|$$

Since both ΔP_i^n and $f(P_i^{n+1})$ should approach the zero vector, and $f(P_i^0)$ and P_i^{n+1} aren't anticipated to be zero necessarily, the overall error decreases as we improve $f(P_i^{n+1})$ relative to $f(P_i^0)$ and ΔP_i^n in compared to P_i^{n+1}.

After each iteration, the function must update the stiffness matrix $\hat{K}_i^{n+1}$in accordance with the current cycle point, considering $P_i^n = \Delta u_i^n$ (where $cyc_{in_i} = u_{i,6}$ and similarly P_i^n). It also updates the residual $f(P_i^n)$ by both updating $\hat{K}_i^{n+1}P_i^{n+1}$ and adding the residual from the previous time iteration before the $\hat{K}_i$ update, as follows:

$$\hat{K}_i^{n+1} = \hat{K}(cyc_{in} = cyc_{in_i} + \Delta cyc_{in_i}^n)$$

$$f(P_i^{n+1}) = \Delta\hat{F}_i - \hat{K}_i^{n+1}P_i^{n+1} + (\hat{K}_i^0 - \hat{K}_i^{n+1})u_i$$

Computational efficiency optimization

The stiffness matrix $\mathrm{K}(\mathrm{cyc})$ is stored as a 3D list of square matrices corresponding to the gear mesh stiffness according to the cycle of the input shaft. During NR iterations, the Jacobian matrix inversion, equivalent to K inversion, is repeated. It is worth noting that the time vector is significantly longer than the cycle vector, often spanning multiple orders of magnitude. To optimize computation, instead of computing the inverse $\mathrm{K_i}$ matrix for each time step, we calculate the inverse $\mathrm{K}(\mathrm{cyc})$ matrix once for an entire cycle. We then select the corresponding index in the cycle that matches the current time step. This approach leverages the cyclic nature of the stiffness matrix over time. The index in the cycle is calculated similarly to the **`convertCyc2Time`** function, which isn't invoked in the numerical solution function to save processing time. The input shaft's cycle is determined by normalizing the 6th index in the generalized coordinates vector (representing the input shaft's rotation angle) by 2π. For more details, please refer to the **`convertCyc2Time`** function.

Postprocessing

The system's solution vectors, $\mathrm{u}(\mathrm{t}), \dot{\mathrm{u}}(\mathrm{t})$, and $\ddot{\mathrm{u}}(\mathrm{t})$, can exhibit numerical instability until convergence, as shown in the partial segment in **Figure 46**. To mitigate this, signals are trimmed, and numerical oscillations are removed. By default, $2 \times \mathrm{z_{out}}$ cycles of the input shaft are generated, and then the solution is cut by half. This ensures that the remaining signal encompasses a complete cycle of each tooth pair interaction combination. However, users have the option to adjust this value as needed. Please take this procedure into consideration for subsequent processing, such as angular resampling, synchronous averaging, etc.

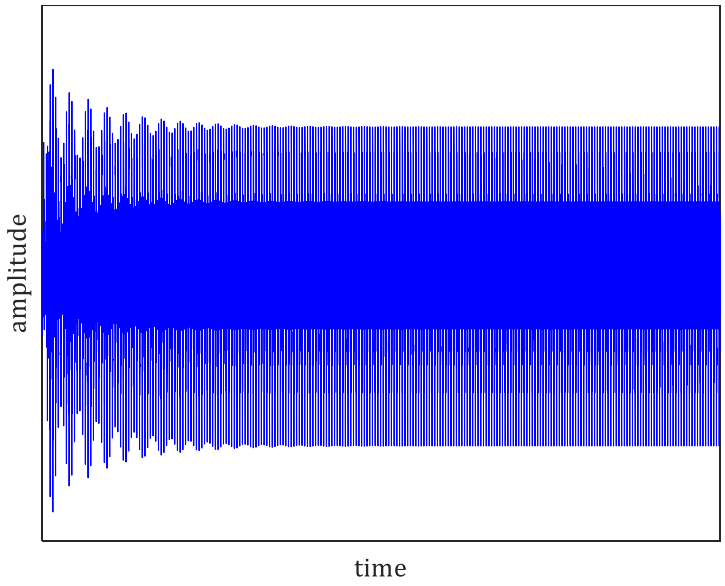

Figure 46 - A partial segment from the instable numerical solution

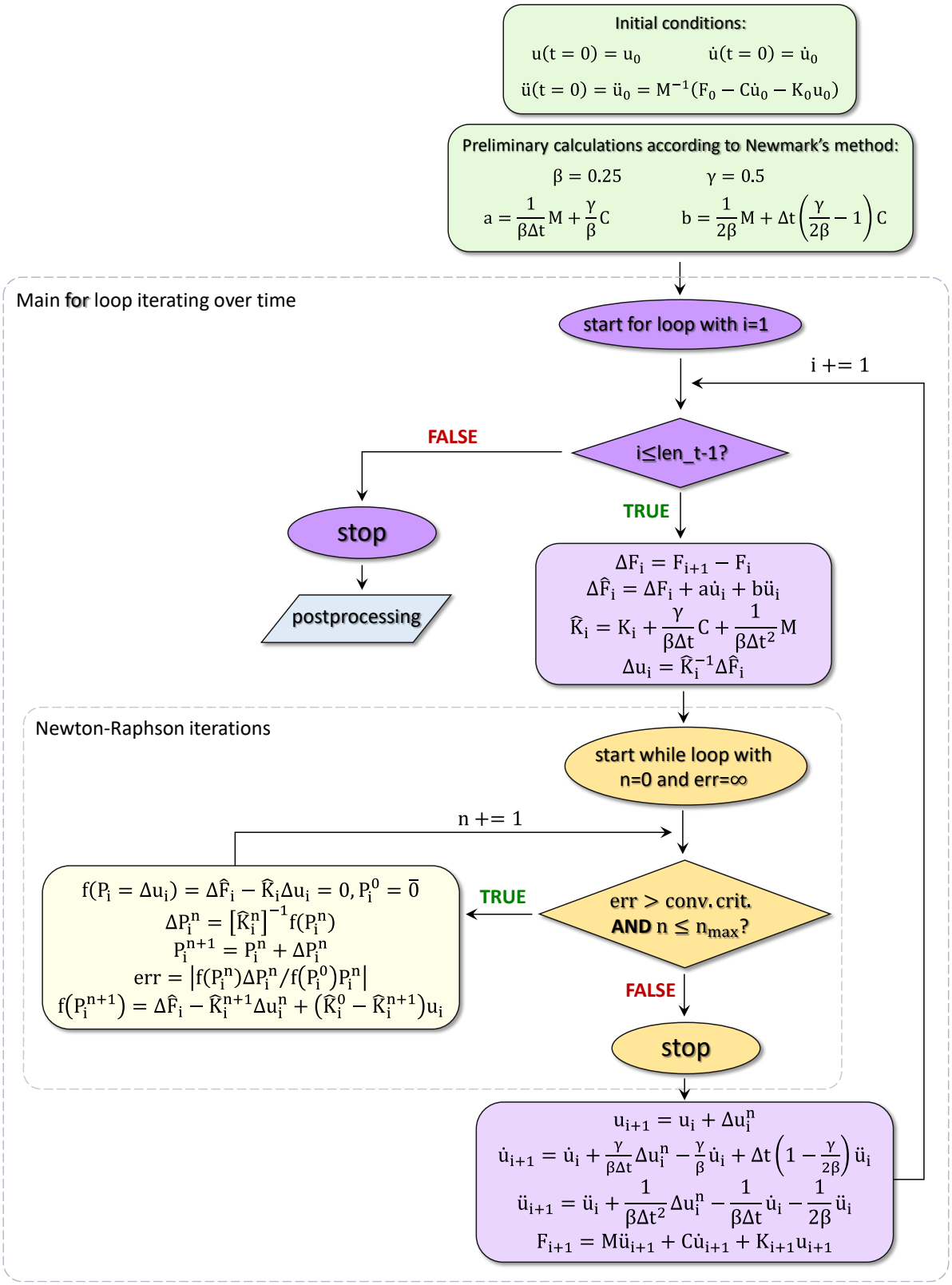

Figure 47 - A general block diagram of the numerical solution algorithm