A novel hybrid and publicly available model for spur gear vibrations

User Manual

Authors:

Lior Bachar

Omri Matania

All rights reserved to the PHM-BGU Laboratory,
Ben-Gurion University of the Negev, Be'er Sheva, Israel.
2023

calc_alpha_cyc

This function calculates the angle of the pressure line based on the cycle. Except for the Through Face Fault, both the gear and pinion always have an involute profile, ensuring that the contact point lies on the nominal pressure line. However, in the case of a Through Face Fault, the faulty part lacks an involute profile, which causes the contact point to deviate from the pressure line.

In the case of the Through Face Fault, the function iterates over the cycle (the "*for loop*" starting from line 55) and calculates the current cycle fraction ("*curr_rem_cyc*") based on the transmission ratio ("*tr*"). This is necessary because the pressure angle varies periodically in relation to the transmission ratio. If the current cycle fraction lies between the beginning of the fault and its midpoint, the function calculates the pressure angle using the following formula:

$$\alpha = \alpha_{\text{nominal}} + \frac{\sqrt{R_s^2 - R_b^2} - \sqrt{R_{p_1}^2 - R_b^2}}{R_b}$$

If the current cycle fraction is between the middle and the end of the fault, the pressure angle is calculated using the following formula:

$$\alpha = \alpha_{\text{nominal}} + \frac{\sqrt{R_e^2 - R_b^2} - \sqrt{R_{p_2}^2 - R_b^2}}{R_b}$$

The function utilizes two running indices, "*jj*" and "*kk*", within the "*for loop*" (which starts on line 55) to extract the relevant values of R_{p_1} and R_{p_2}. When the current cycle fraction is within the first half of the fault, "*jj*" increases monotonically until the current cycle reaches the middle of the fault. Then, "*kk*" increases monotonically until the current cycle fraction reaches the end of the fault. Once the current cycle fraction exits the fault, the function reinitializes the indices "*jj*" and "*kk*" to 1 in preparation for their use when the current cycle fraction reaches the fault again. Since the current cycle fraction is only inside the fault once during each transmission ratio period, resetting "*jj*" and "*kk*" to 1 is appropriate as they will be needed again only when the current cycle fraction completes a full transmission ratio period.

`calc_cont_line_properties`

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 1**. 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:

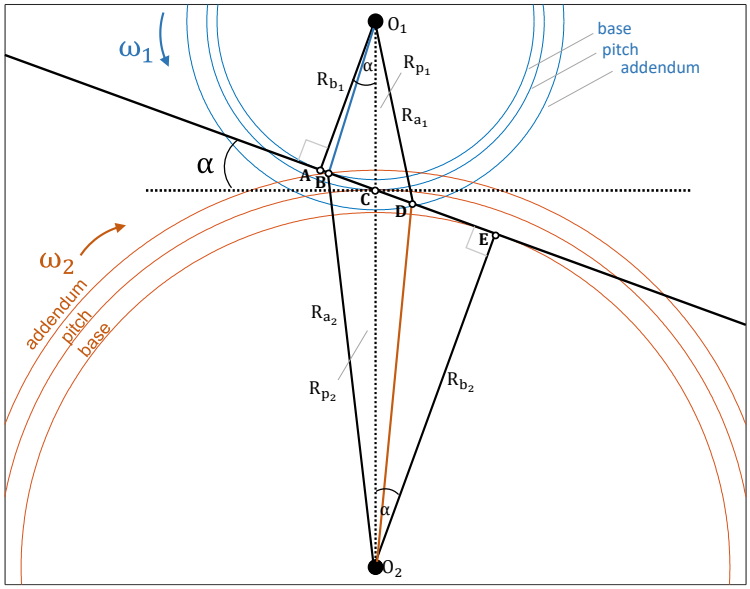


Figure 1 - Illustration for the parameters of the initial contact radius calculation

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

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 1**, 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.

`calc_Euler_Lagrange_components`

This function utilizes the Euler-Lagrange model to calculate the components of the equations of motion describing the simulated system presented in **Figure 2**, 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 3** demonstrates the methodology of this function.

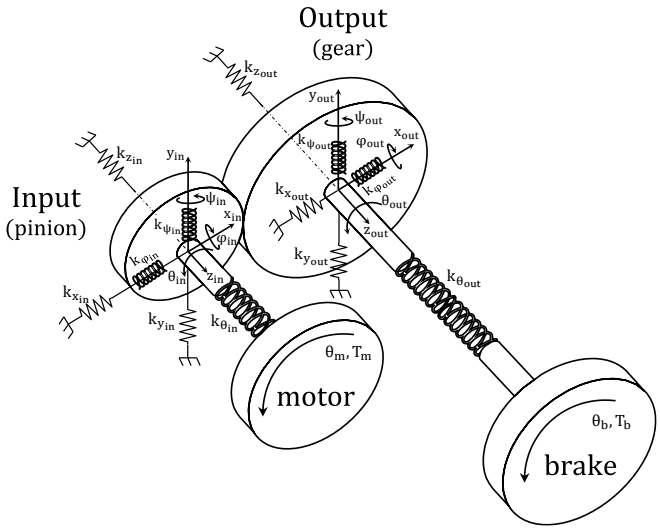


Figure 2 – Qualitative sketch of the simulated system

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 3**. 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:

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

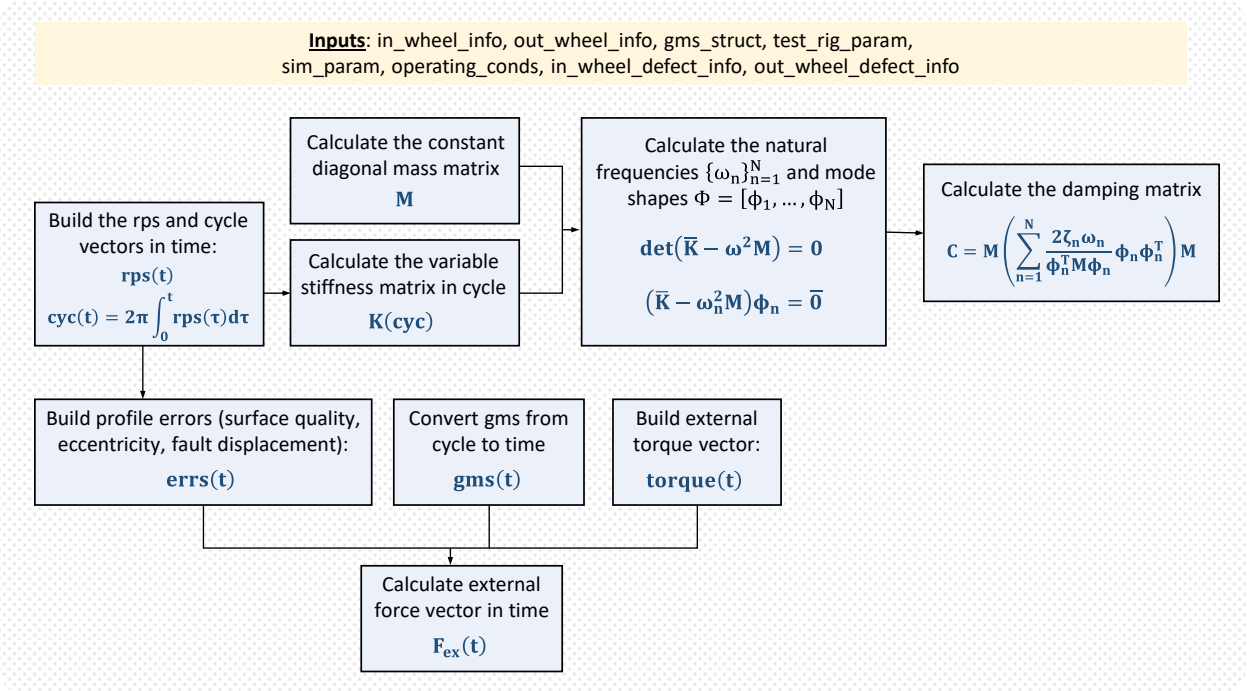


Figure 3 - Methodology of calculating the Euler-Lagrange components.

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 $\bar{\mathrm{K}}$ as follows:

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

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

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

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

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

The damping matrix C can be rewritten as:

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

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.

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

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

`calc_fault_displacement`

This function calculates the displacement caused by local or distributed tooth faults, which lead to deviations from the involute tooth profile. This calculation takes into account both the gear and pinion wheels.

In all cases except for the Through Face Fault, both the gear and pinion have an involute profile, resulting in a displacement of 0. However, for the Through Face Fault, the function calculates the intersection between the healthy tooth and the red line shown in **Figure 4**, as well as the intersection between the original tooth of the faulty tooth and the red line. The displacement is then calculated as a function of the cycle, using the following formula, where h represents the maximum deviation.

$$\text{fault displacment} = -\text{h} \cdot \frac{1 - \cos(2\pi \cdot \text{realative location})}{2}$$

An example of the fault displacement as function of the cycle for Through Face Fault is depicted in **Figure 5**.

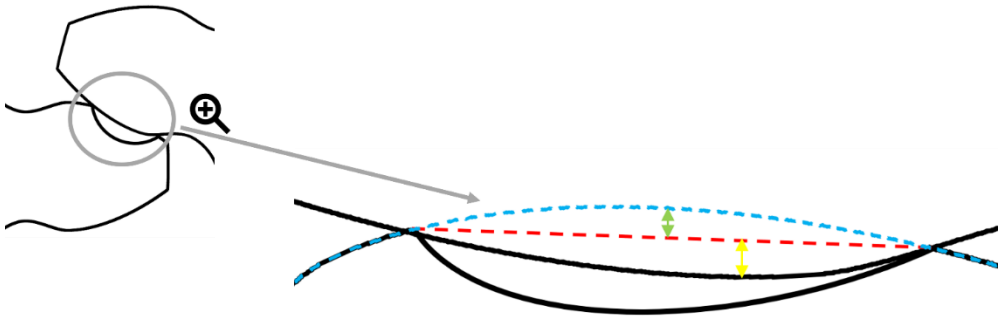


Figure 4 - The assumed deflections of the healthy and faulty tooth due to a Through Face Fault. The deflection of the healthy tooth is marked by yellow and of the faulty tooth by green.

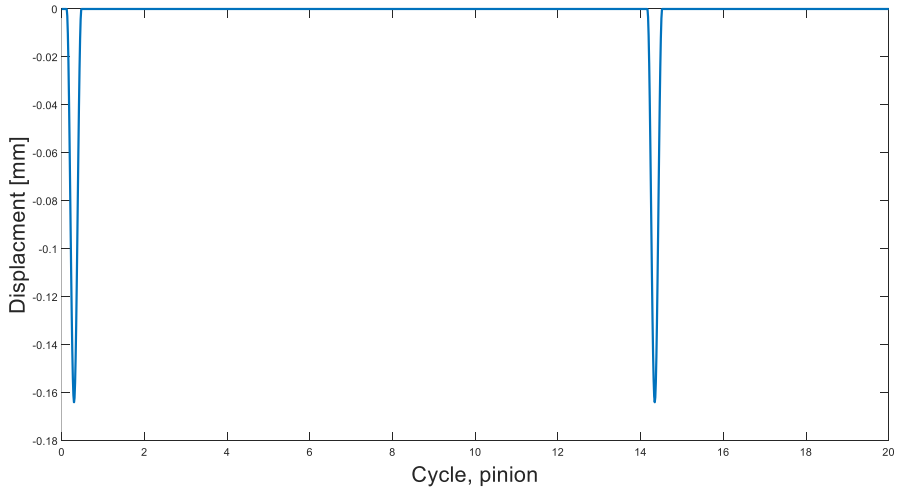


Figure 5 - Displacement between the teeth of the gear and the pinion as function of the 20 first cycles.

calc_forces_vctr_t

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 6**.

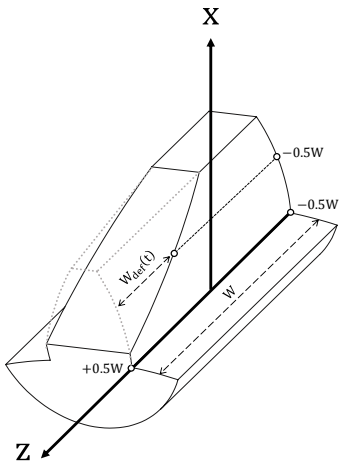


Figure 6 - Illustration of z-axis dependency in a tooth breakage scenario

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 6**):

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

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

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 $\mathbb{E}[z] \neq 0$.

`calc_gearmesh_stiffness`

This function calculates the gear mesh stiffness (gms) as a function of the rotating cycle of the input shaft. The equivalent stiffness of a single tooth pair is calculated with the function The shape of the gms is a square-wave-like curve, as illustrated in **Figure 7**.

Figure 7 - An illustration of the several first periods of the gms.

There are two types of interactions: between healthy teeth and between a faulty tooth and a healthy tooth. Thus, the function calculates the equivalent stiffness for a healthy teeth pair and for a faulty-healthy teeth pair. The function calculates the GMS in the following steps, as also depicted in **Figure 8**:

1. Calculation of the equivalent stiffness of a healthy tooth pair.
2. Calculation of the equivalent stiffness of three following teeth pairs, including the resulting stiffness of the former pair and the following pair. This period represents the basic period of the equivalent stiffness.
3. Generation of the equivalent stiffness as a function of the cycle based on the equivalent stiffness of the three following pairs.
4. Calculation of the stiffness for a faulty-healthy pair.
5. Calculation of the equivalent stiffness of the faulty-healthy pair including the resulting stiffness of the former pair and the following pair.
6. Replacement of the equivalent stiffness of the first teeth interaction with the equivalent stiffness of the faulty-healthy pair.

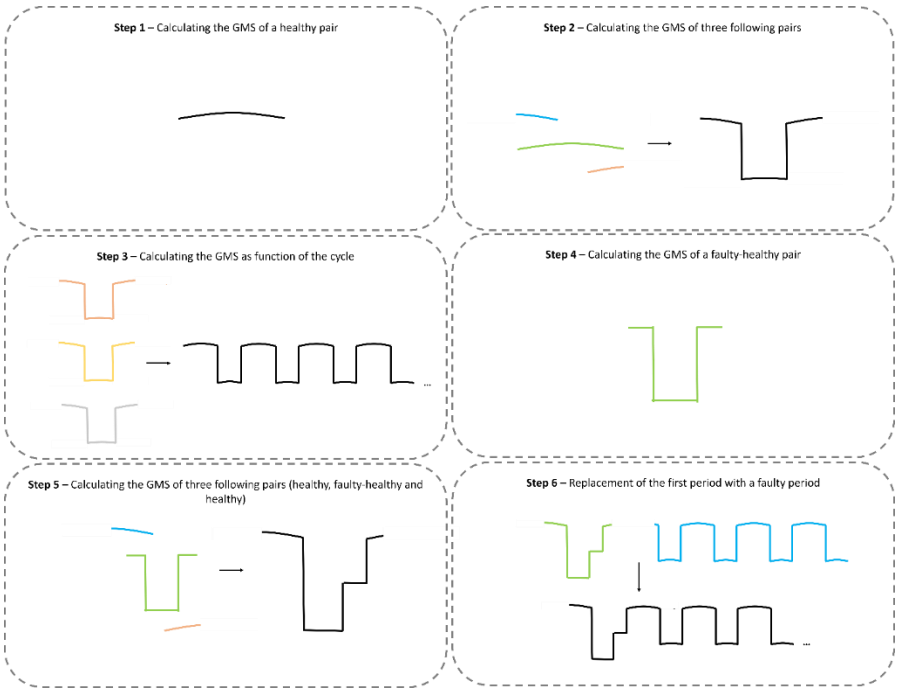


Figure 8 - An illustration of the several first periods of the gms.

calc_healthy_tooth_profile

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 9** 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.

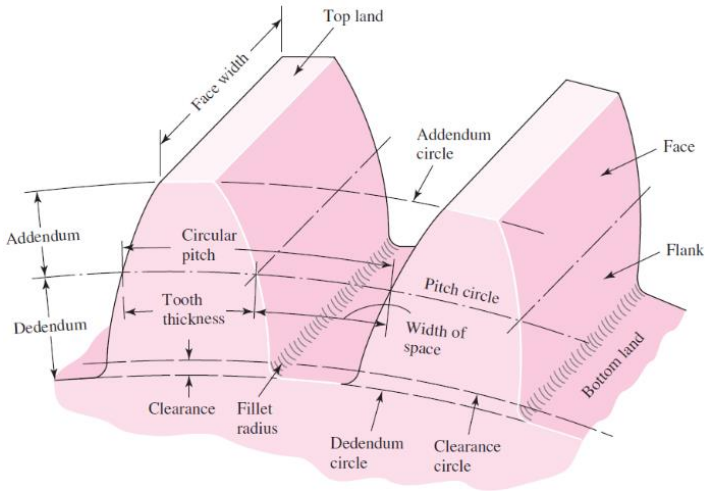


Figure 9 - Nomenclature of spur gear teeth [1]

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:

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

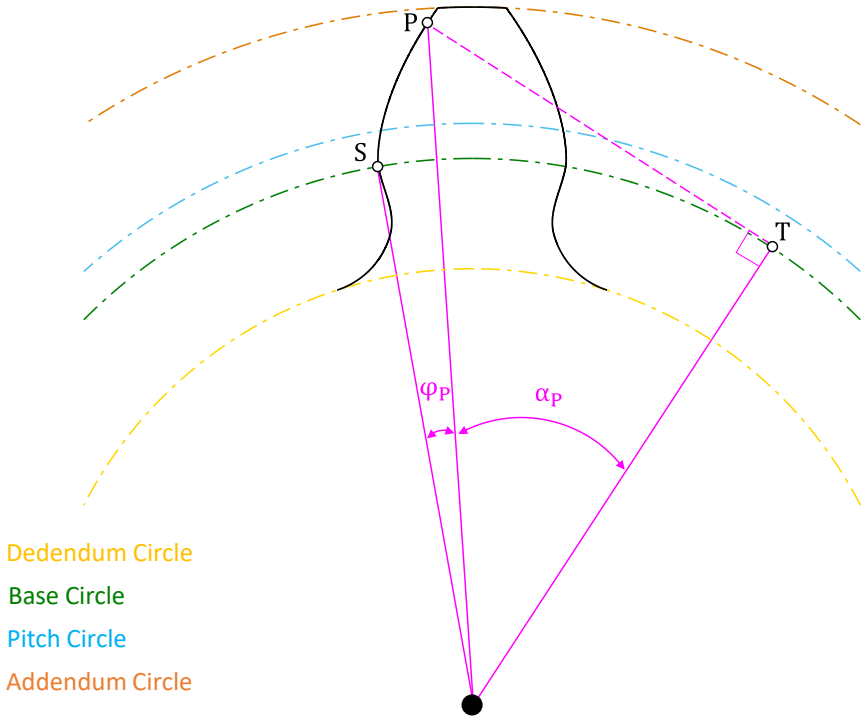


Figure 10 - Illustration of the involute profile properties in spur gears.

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 10**. 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 11**. The involute function is employed to calculate the angle θ_P for any point P along the involute profile.

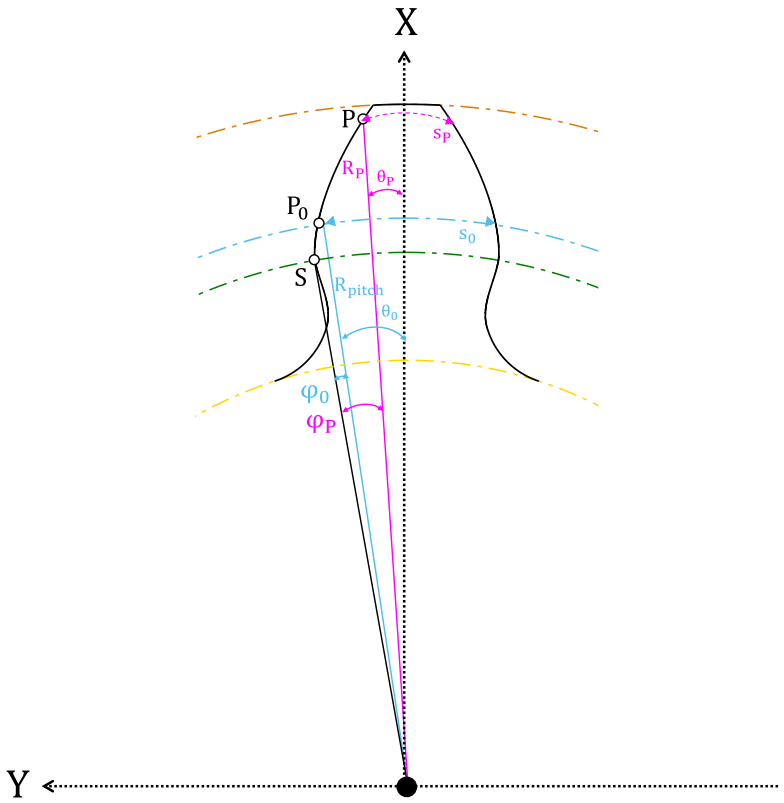


Figure 11 - Illustration of the involute profile calculation

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 10** 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 12**. The Cartesian coordinates of the involute profile are:

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

Where

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

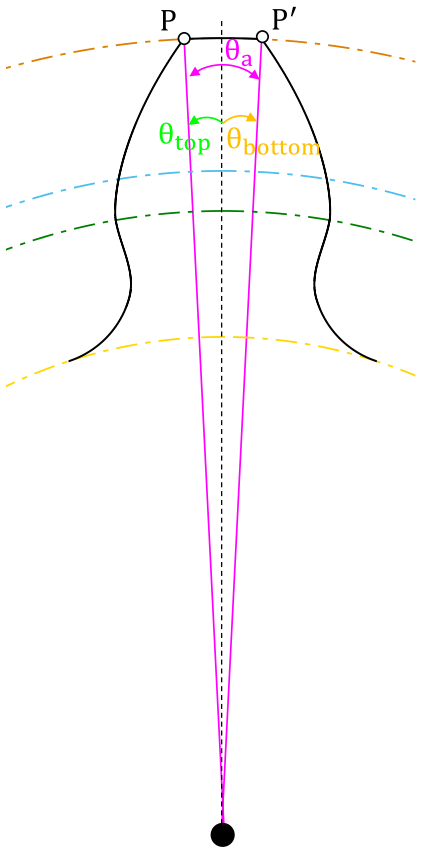


Figure 12 - Illustration of the tooth tip curve parameters

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 13**.

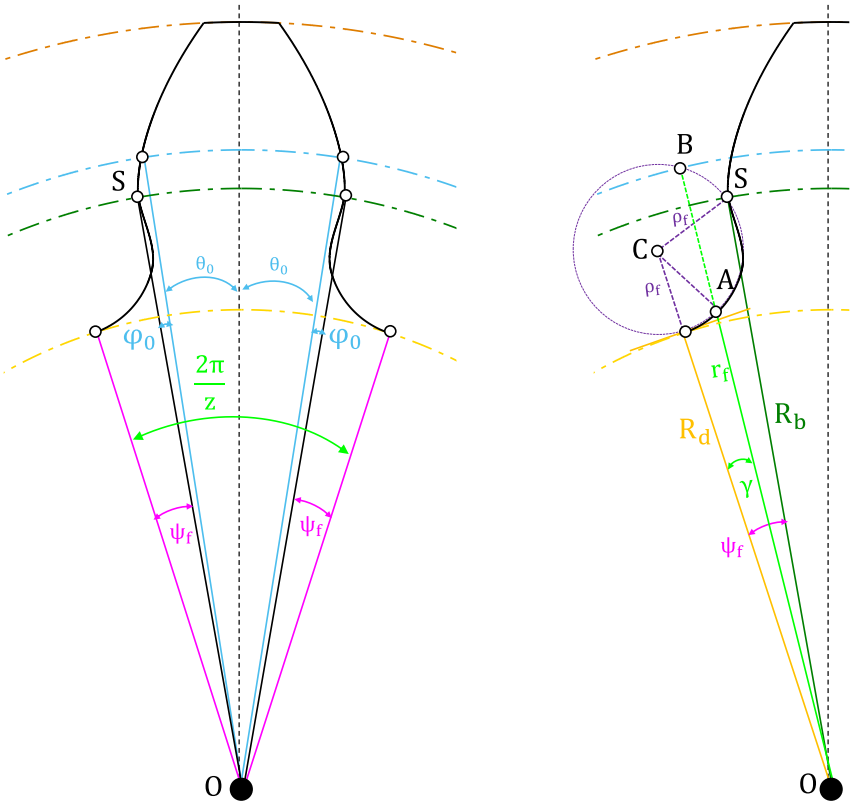


Figure 13 - Illustration of the fillet curve parameters

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 13**, 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

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

`calc_inv_(axial/bending/shear)_stiffness`

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 "**`calc_Keq`**". 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 14**:

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

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:

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

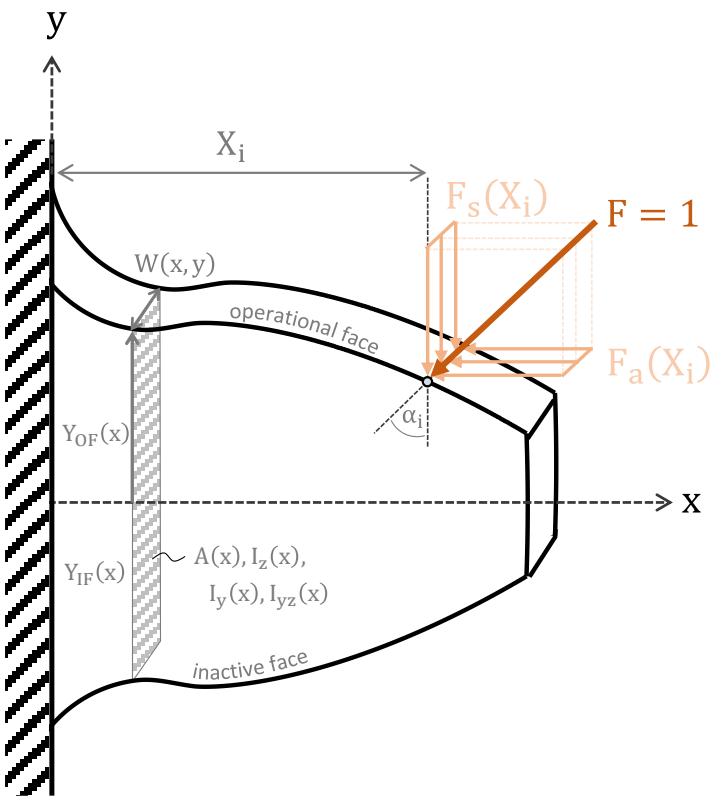


Figure 14 - A qualitative three-dimensional free-body diagram of a tooth in operation

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 ($X_i, Y_i, \overline{Z}_i$) 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 $\overline{F}(X_i), X_i, Y_i, \overline{Z}_i$) 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.

calc_inv_axial_stiffness

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 14**. 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 1** outlines the parameters subject to health status variation and those that remain constant according to the nominal healthy status.

Table 1 - 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 ($Y_{def}(x)$)

calc_inv_shear_stiffness

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 15**.

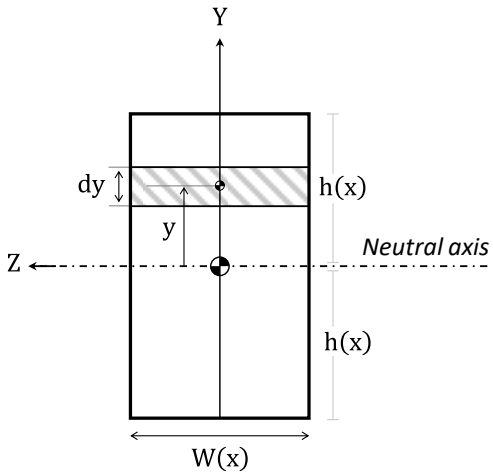


Figure 15 - An illustration of a rectangular cross-section in the z-y plane

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 16**). The operational face exhibits partial defects, resulting in a cross-sectional shape that alternates between a rectangle and a polygon, as illustrated in **Figure 17**. 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:

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

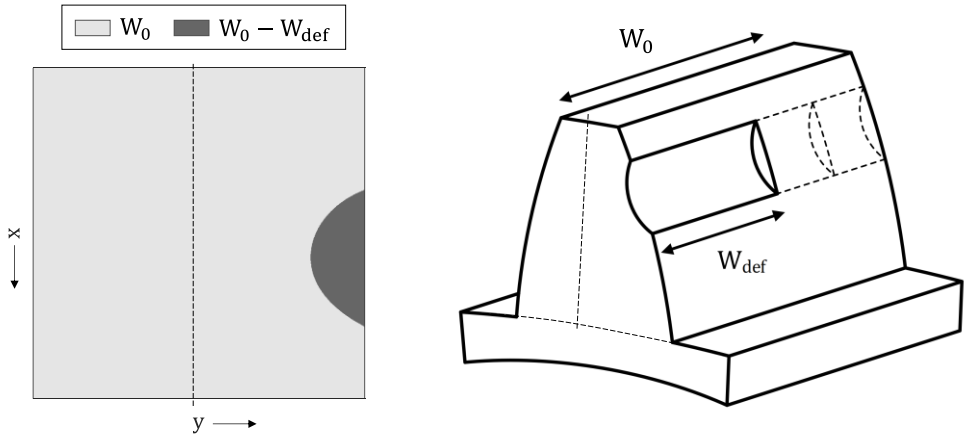


Figure 16 - An illustration of the tooth width variation along x and y axes in case of a partial tooth face fault

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

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

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 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 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

calc_inv_bending_stiffness

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:

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

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 17**.

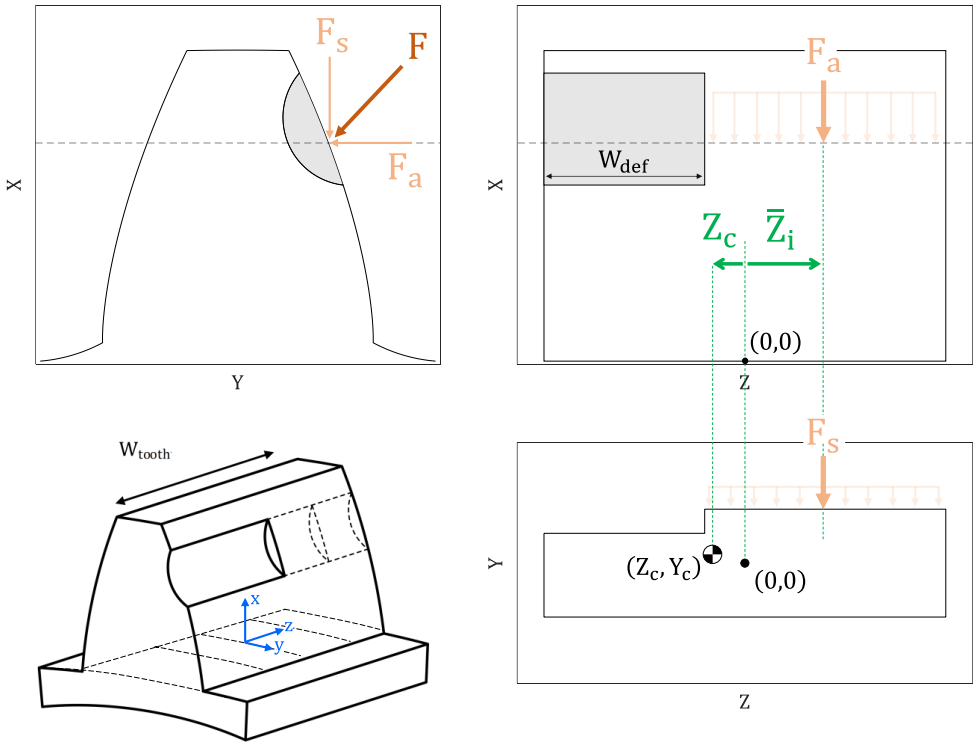


Figure 17 - An illustration of the tooth geometry and the meshing force distribution in three views

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 18** 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:

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

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

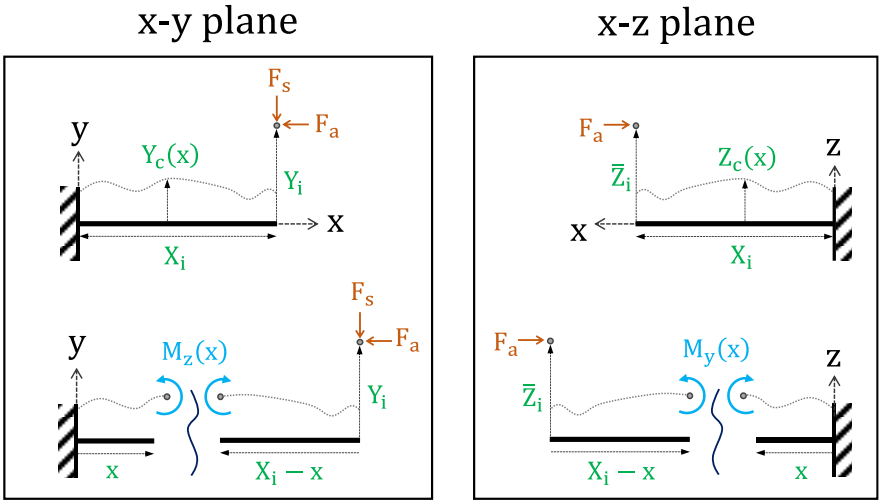


Figure 18 - Internal bending moments in the x-y and x-z planes

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 3**. In **Table 3**, 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 **`consider_varaible_alpha`**.

Table 3 - 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 4** outlines the parameters subject to health status variation and those that remain constant or zero according to the nominal healthy status.

Table 4 - 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

calc_inv_fillet_foundation_stiffness

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 19**, 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 5**.

Table 5 - 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

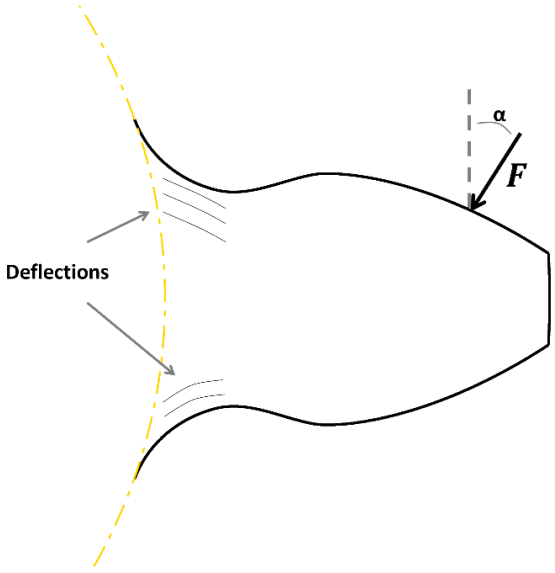


Figure 19 - Deflections in the foundation of the tooth due to the force

calc_inv_hertzian_cont_stiffness

This function calculates the inverse stiffness derived from the residual stresses induced by the Hertzian contact stresses. The function calculates the stiffness using the formula:

$$K_h = \frac{\pi \cdot E \cdot W_x}{4(1 - v^2)}$$

Where E is Young's modulus, W_x is the tooth width and v is Poisson's ratio.

`calc_Keq`

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 20**.

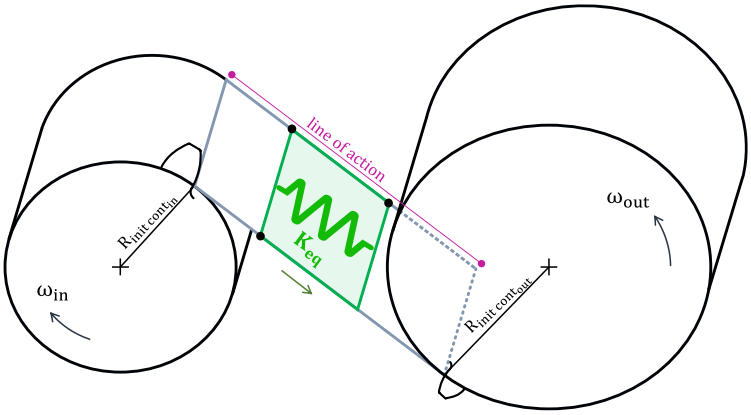


Figure 20 - Illustration for the equivalent stiffness of a tooth pair along the line of action

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 21** and calculated as follows:

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

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

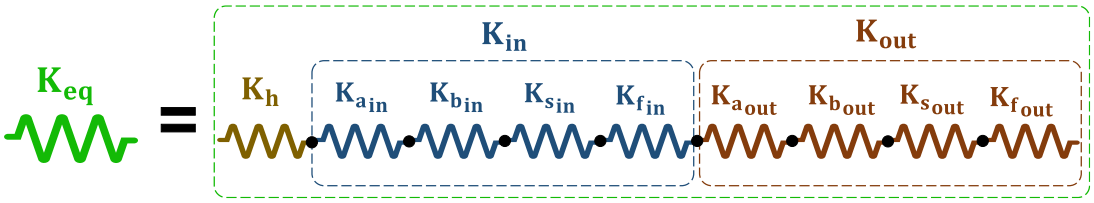


Figure 21 - The equivalent stiffness K_{eq} as a combination of nine springs connected in series.

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 22**.

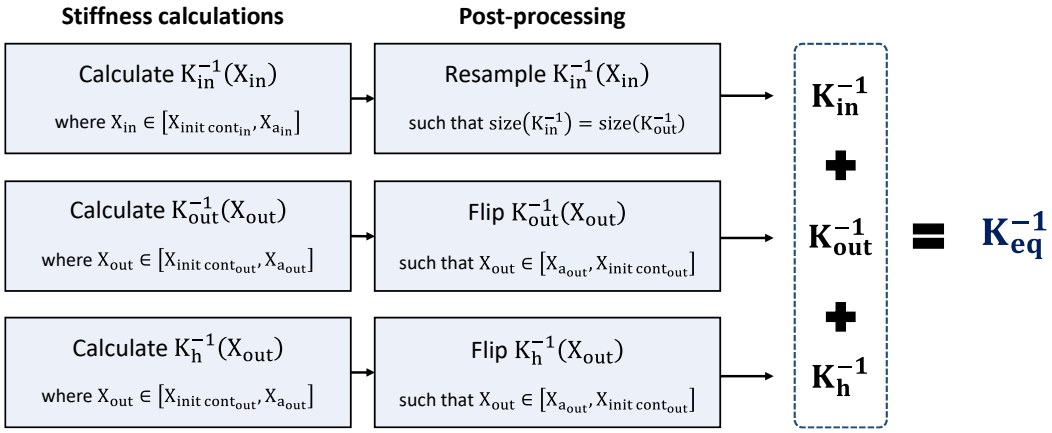


Figure 22 - Calculation procedure for inverse equivalent stiffness

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 defect_info structure, a conflict arises as it does not exist in this scenario. The function handles the absence of defect information and generates a degenerated defect_info 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.

`calc_linspace_matrix`

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 23**. 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.

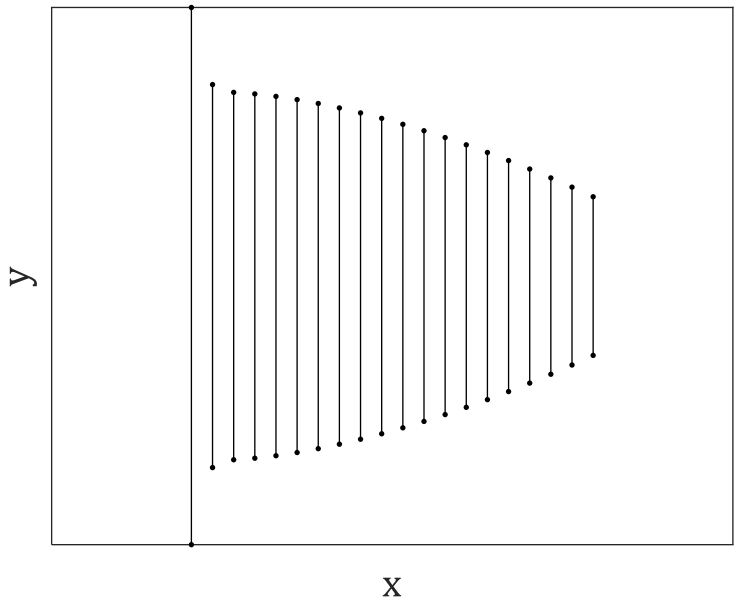


Figure 23 - An illustration of linearly spaced vectors in y for each x-point along the tooth profile

`calc_mass_properties`

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 24**:

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

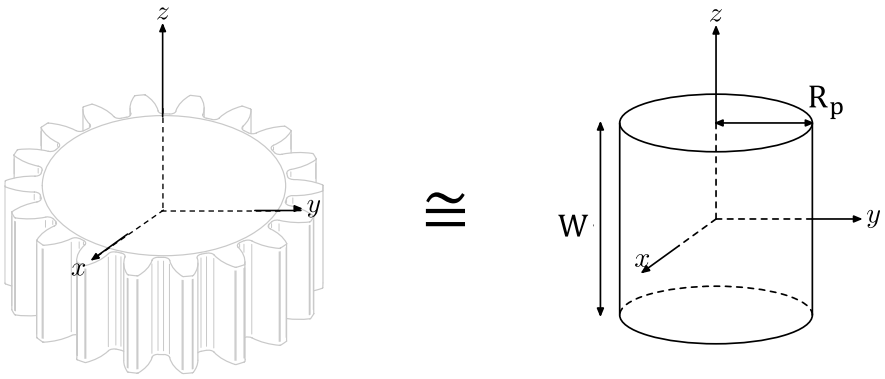


Figure 24 - Approximation of a spur gear as a cylinder with pitch radius illustration

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 24**, 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:

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

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

calc_shaft_stiffness

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 25**.

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:

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

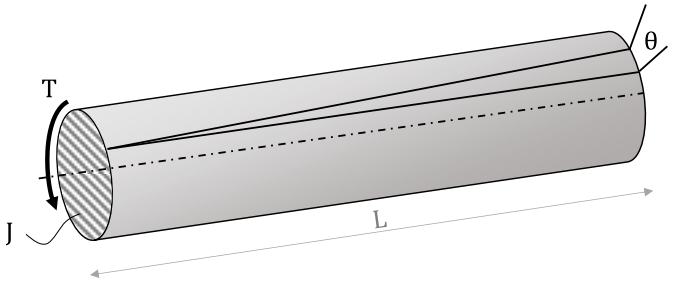


Figure 25 - Illustration of torsional shaft parameters

calc_tooth_face_fault_geometry

This function calculates key geometric properties of the cross-section in the z-y along the x-axis in case of a 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 26**. This defective area is characterized by three parameters illustrated in **Figure 26**, 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.

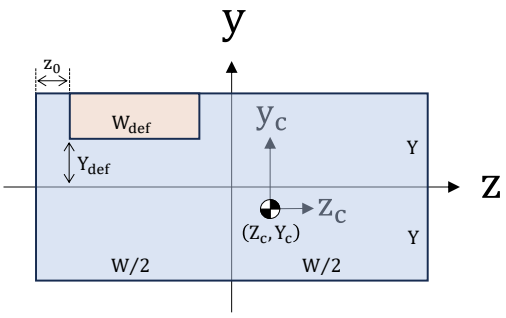


Figure 26 - An illustration of the cross-section in the y-z plane in the tooth face fault status

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 26**.

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

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 6**, derived from **Figure 26.**

Table 6 - 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

calc_variable_stiffness_mtx

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 2**. 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 7**.

Table 7 - 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:

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

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

consider_variable_alpha

This function compensates for the varying pressure angle caused by the Through Tooth Fault by fixing the potential energy vector within the boundaries of the fault segment. It is assumed that during a Through Tooth Fault, the healthy tooth rotates along the fault edges, which act as a "rotation axis" as shown in **Figure 27**. Therefore, the potential energy should only be calculated in that region, taking into account the modification of the variable alpha.

Since the healthy tooth rotates along both fault edges, the function compensates for the varying pressure angle in two parts: between the start and the middle (line 34) and between the middle and the end (line 34). To compensate for the varying pressure angle, the function utilizes the "*modify_U*" function, where the modified U is updated by multiplying the original potential energy at the edge by:

$$U_{modified}(i) = U_{edge}\frac{F_1(i) \cdot F_2(i)}{F_1(i_{edge}) \cdot F_2(i_{edge})}$$

I.E., the original potential energy U assumes that the potential energy at the edge U_{edge} is proportional to the forces $F_1(i_{edge})$ and $F_2(i_{edge})$, as well as the potential energy corresponding to the healthy state. However, due to the Through Tooth Fault, the potential energy of the corresponding indexes inside the fault is altered because the contact point is no longer at the edges and the angle is different. As a result, for each index within the fault, the potential energy is updated by modifying the potential energy at the edge using the ratio between the test forces $F_2(i_{edge})$. This adjustment accounts for changes in the pressure angle.

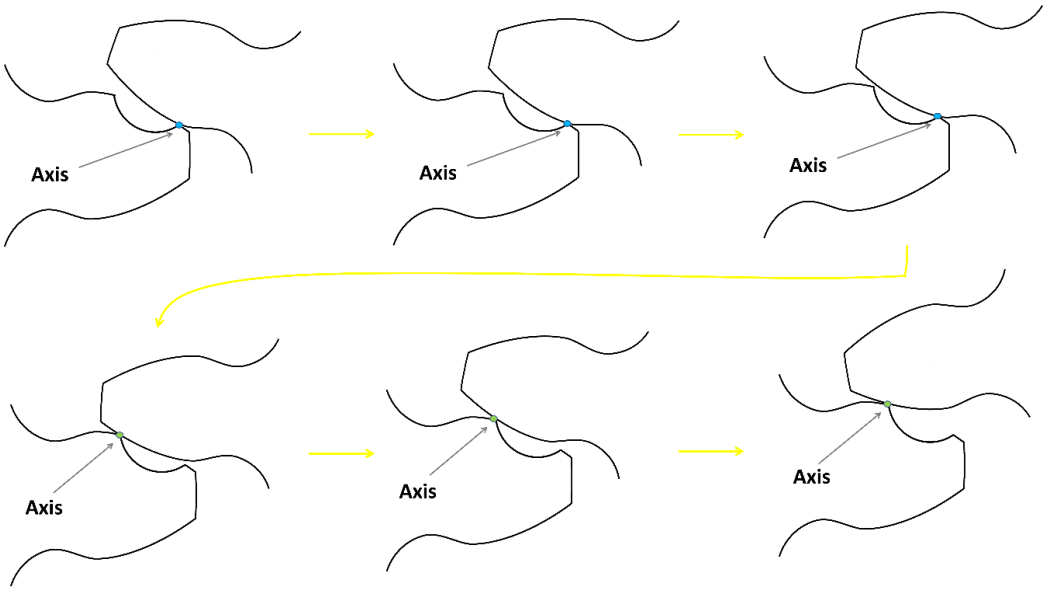


Figure 27 - Interaction of the healthy tooth with the Through Face Fault

convert_cyc2t

This function gets any structure (vector/matrix/list) "*x_cyc*" in the cycle domain, which is periodic as a function of the cycle and corresponds to the cycle vector with a resolution of "*d_cyc*". It returns the equivalent representation of "*x_cyc*" (or a portion of it) in the time domain.

The function calculates the cycle as a function of time in line 30 based on the rotational speed, and in line 31, it converts it to cycle indexes by dividing the cycle by the cycle resolution. Since "*x_cyc*" can be periodic with respect to the cycle as a function of time, the modulo operation is used in line 31 to consider the indexes within the periodic nature of "*x_cyc*". Subsequently, the function calculates the signal in the time domain "*x_t*" in lines 34-45 by extracting the relevant values from "*x_cyc*".

`find_circle_center`

This function calculates the center of a circle based on its radius and two points on its circumference. The function consists of the following steps, as also illustrated in **Figure 28**:

1. Calculating the values of the middle between the points and the distance.
2. Calculating the half chord size, and then the radius size.
3. Calculating the perpendicular vector for reaching the optional upper center and its size by subtracting the depth from the radius size (two points and a radius can correspond to two different circles. The function chooses always the upper one because it is the relevant one for the Trough Tooth Fault calculation).

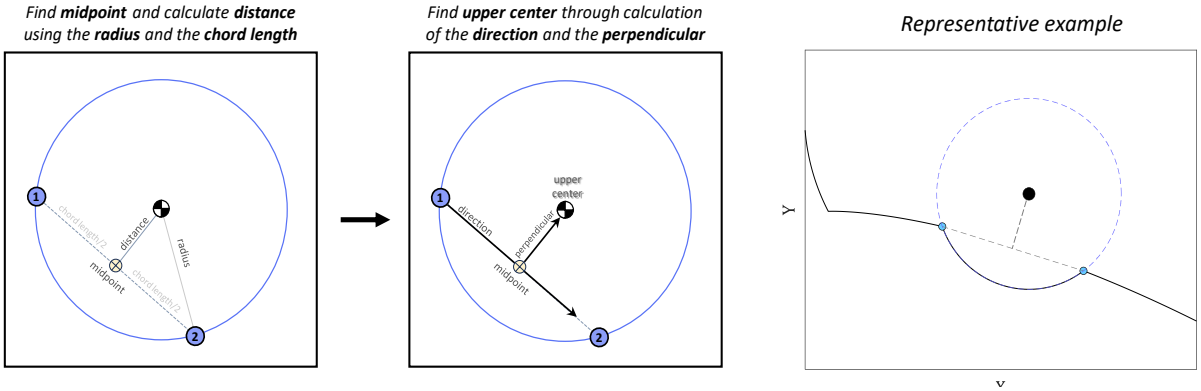


Figure 28 - Calculation of the upper center location of the circle based on the location of the points.

`generate_defect_info_struct`

This function generates the parameters for each fault, according to the fault's requirements. In lines 28-35, the function generates the tooth profile based on the fault type using other functions and then builds the wheel defect info structure based on the calculated wheel defect info.

generate_gear_simulated_data

This function is the main function of the Spur Gear Dynamic Model library. This is the function that the user needs to call in order to generate a simulated signal. The function consists of eight steps:

1. Setting the simulated parameters based on the default values and user inputs.
2. Generating the healthy tooth profile.
3. Generating the profile's errors.
4. Generating the defect information structure.
5. Calculating the gear mesh stiffness.
6. Calculating Euler-Lagrange components.
7. Solving Euler-Lagrange equations.
8. Building the simulated signal based on the numerical solution.

generate_teeth_profile_errors

This function is designed to generate teeth profile errors resulting from manufacturing imperfections, 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 29**. 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.

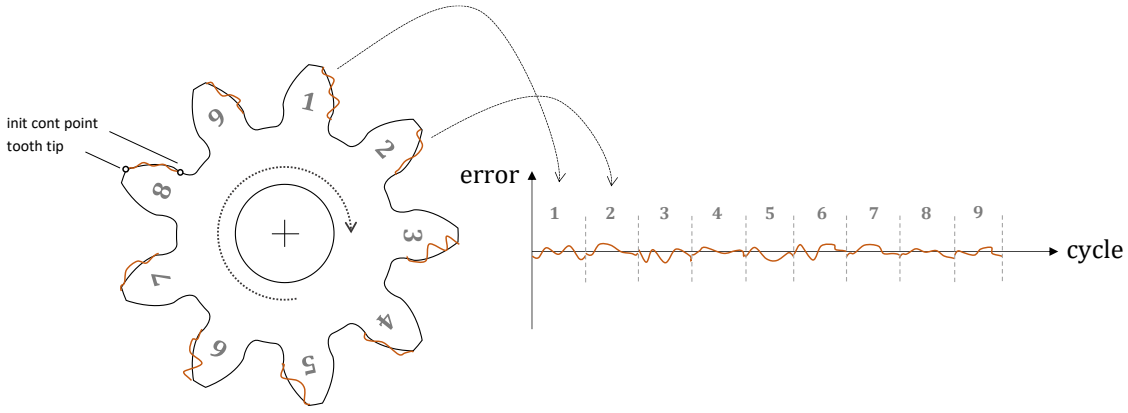


Figure 29 - A graphic illustration for the profile errors vector in cycle

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 30**. 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 30**. 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.

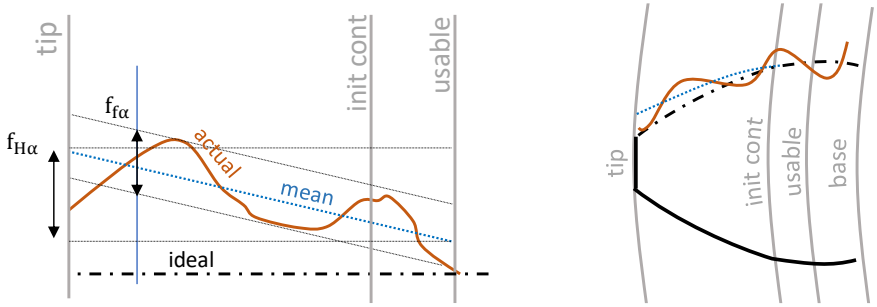


Figure 30 - Relevant parameters of the surface roughness according to DIN-3962 standard

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 30**). 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 1**. 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.

generate_tooth_breakage_profile

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 31**, 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.

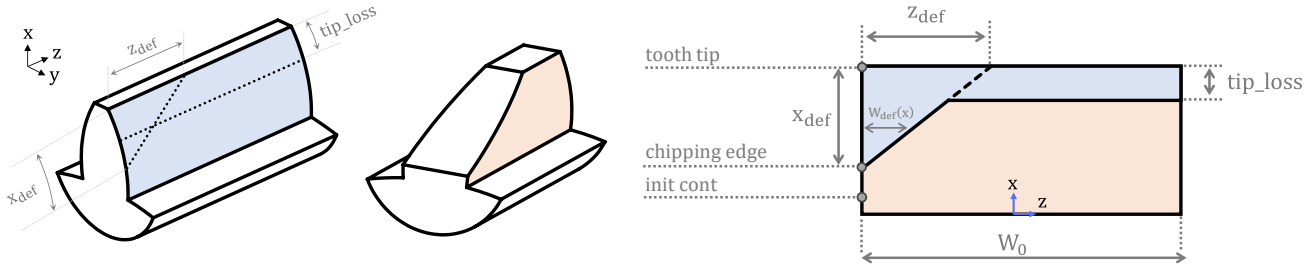


Figure 31 - Illustrative sketches and definitions of the simulated tooth breakage faults

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 31**. 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 31**. 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.

generate_tooth_face_fault_profile

This function calculates the profile of a tooth with a face fault, given the healthy tooth profile and the defect parameters. Additionally, the function calculates the deflection (h) of the teeth in the case of a Through Face Fault. The function generates a tooth profile with a partial circle fault as depicted in **Figure 32**. The fault can be designated by a percentage using the initial location of the fault (blue point), the end of the fault (red point), and depth. Alternatively, the fault can be designated by subtracting the circle from the healthy tooth profile, where the center of the circle and its radius should be defined (see **Figure 32**). The function generates the tooth profile in five steps:

1. Calculation of the X and Y coordinates of the lower part of the circle.
2. Calculation of the intersection points between the circle and the tooth profile.
3. Interpolation of the tooth profile and calculation of the edge points of the fault.
4. Replacement of Y values of the fault indexes with the values of the circle to generate the defect profile of the tooth and find the middle of the fault.
5. Calculation of the deflection that is relevant to a more advanced calculation of the tooth through fault displacement. The variable h is defined in **Figure 33**. For a partial fault, the function calculates relevant information for future calculations, such as the era of the fault.

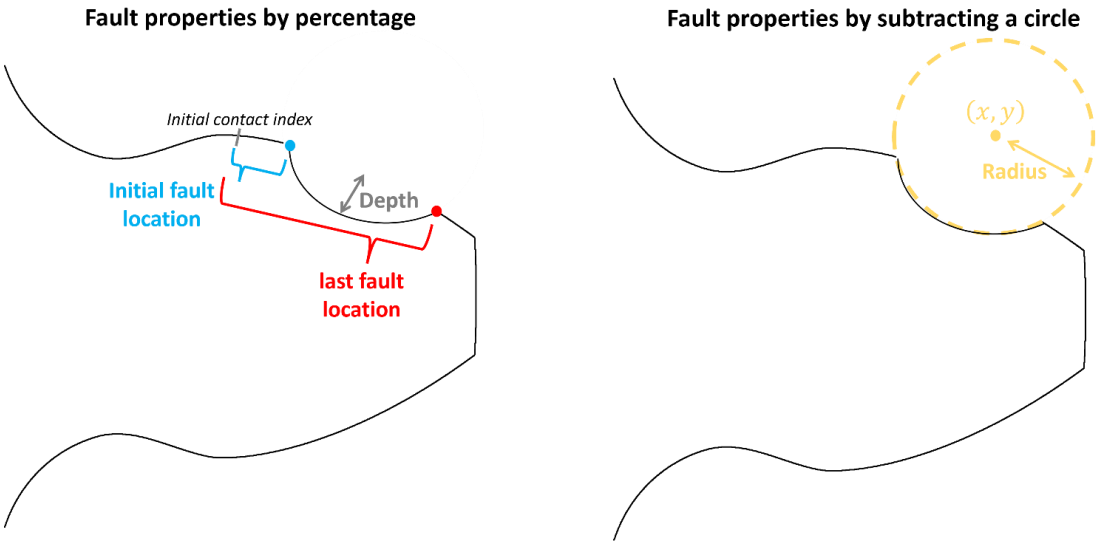


Figure 32 - The two relevant options for the input parameters of the Through Face Fault.

The inter-function "**calc_deviation_from_curve**" calculates the deviation from the curve, denoted as h. It achieves this by first rotating the Y values of the healthy profile so that the line connecting the fault edges is parallel to the horizon. Then, it calculates the deviation between the maximum and minimum values, as depicted in **Figure 33**. This distance corresponds to the distance between the maximum value of the healthy profile and the red line in **Figure 33**. This calculation is relevant for both the in and out wheels. In the out wheel, it calculates the portion that would have been deflected if the tooth were healthy, while in the in wheel, it determines the portion that penetrates the faulty gear.

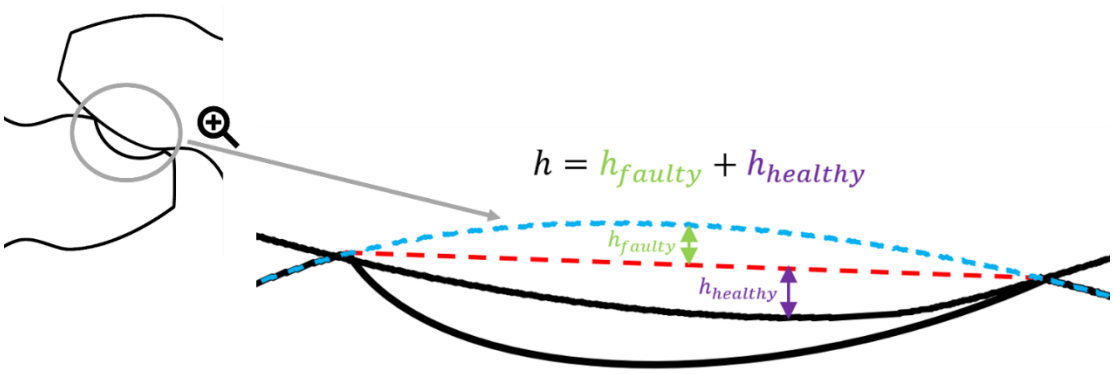


Figure 33 - The assumed deflections of the healthy and faulty tooth due to a Through Face Fault. The deflection of the healthy tooth is marked by purple and of the faulty tooth by green.

Get_material_properties

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.

`get_parameters`

This function serves two main purposes: setting the default parameters of the simulation and incorporating designated parameters from the user. The function always receives inputs such as “*module*”, “*z*”, “*tooth_width*”, “*surface_quality*”, “*shaft_dimensions*”, “*speed_in*”, and “*load_out*”. It can also accept additional parameters using “*varargin*”. If any parameters are provided through “*varargin*”, they will override the default parameters.
The function consists of nine steps:

1. Setting the operational conditions based on the user inputs.
2. Setting the initial conditions of the numerical solution.
3. Setting the default simulation parameters.
4. Setting the test rig parameters.
5. Setting the parameters of the wheels.
6. Updating the variable values based on the user input optional parameters. The function iterates over “*varargin*“ and translates the string "*key*" using the dictionary "*dict*". Then it sets the value of the relevant variable.
7. Generating the info structure for each wheel.
8. Calculating the simulation parameters.
9. Updating the rotational speed.

match_indexes

This function is used to convert the indexes of the in and out wheels. Since the in wheel and out wheel have a different number of points in the contact region, a conversion is needed between them for certain functions. It is assumed that there is a linear relationship between the indexes. For example, the first index of the gear (e.g., 2700) corresponds to the same index of the pinion (e.g., 3100), and the last index (e.g., 10,000) of the gear corresponds to the last index (e.g., 10,000) of the pinion. Between these indexes, the linear relationship should be maintained as depicted in **Figure 34** and described here:

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

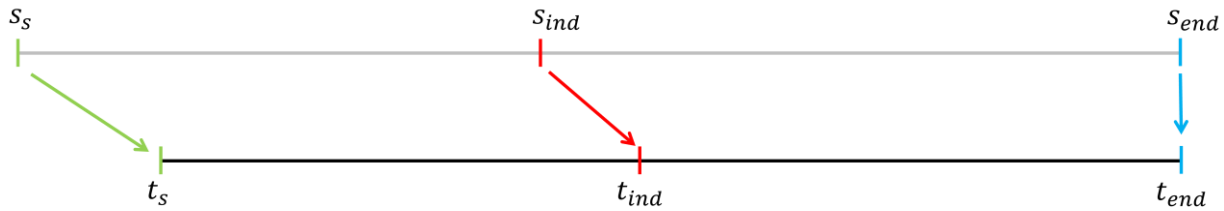


Figure 34 - Conversion of an index

numerical_solution_by_Newmark_and_Newton_Raphson_methods

This function uses the Newmark method and the Newton-Raphson method to solve the Euler-Lagrange system of equations which is written as follows:

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

Where u, $\dot{\mathrm{u}}$, $\ddot{\mathrm{u}}$ are, respectively, the positions, velocities, and accelerations of the system, M is the generalized mass matrix, C is the generalized damping matrix, $\mathrm{K}(\mathrm{u})$ is the generalized stiffness matrix which is dependent on the solution u, and $\mathrm{F}_{\mathrm{ex}}(\mathrm{u})$ is the external forces vector.

The numerical solution consists of four stages: (1) extraction of the relevant variables and basic processing (lines 51-78), (2) preprocessing of the variables to improve the running time (lines 80-87), (3) implementation of the Newmark method (lines 90-135), and (4) Newton-Raphson iterations within the Newmark method stage to refine the solution (lines 94-119).

Stage I:

In this stage, the previously calculated Euler-Lagrange components are extracted. Next, the basic parameters of the function are extracted, initialized, and calculated. For instance, the coefficients of the Newmark method, β (beta) and γ (gamma), are extracted (they always have the values $\beta = 0.25, \gamma = 0.5$, but are defined externally to support future updates). Additionally, the modified coefficients of the Newmark method, a and b, are calculated. The corresponding lines of stage (1) are presented in **Figure 35**.

```
51      %% step 1: load Euler-Lagrange components %%
52      [M, C, K_mtx_cyc, Fex_vctr_t] = deal(...
53          Euler_Lagrange_components.M,...
54          Euler_Lagrange_components.C,...
55          Euler_Lagrange_components.K_cyc_list,...
56          Euler_Lagrange_components.Fex_vctr_t) ;
57      [Fs, d_cyc_gms] = deal(sim_param.Fs, sim_param.d_cyc_gms_coarse) ;
58      dt = 1/Fs ;
59      [ndof, len_t] = size(Fex_vctr_t) ;
60
61      %% step 2: pre-allocation & initial conditions %%
62      [u_ddot, u_dot, u] = deal(zeros(size(Fex_vctr_t))) ;
63      [u_dot(:,1), u(:,1)] = deal(initial_conds.velocity, initial_conds.displacement) ;
64
65      %% stpe 3: initial calculations for average acceleration method %%
66      [beta, gamma] = deal(sim_param.numerical_sol.beta, sim_param.numerical_sol.gamma) ;
67      len_cyc = length(K_mtx_cyc) ;
68      cyc_in = u(6, 1) / (2*pi) ; % u(6,:) is the rotational angle of the driving wheel.
69      cyc_ind = mod(round(cyc_in/d_cyc_gms), len_cyc) + 1 ;
70
71      u_ddot(:,1) = inv(M)*(Fex_vctr_t(:,1) - C*u_dot(:,1)- ...
72          K_mtx_cyc(:, :, cyc_ind)*u(:,1)) ;
73      a = (1/(beta*dt))*M + (gamma/beta)*C ;
74      b = (1/(2*beta))*M + dt*((gamma/(2*beta))-1)*C ;
75
76      [convergence_criterion, max_NR_iterations] = deal(...
77          sim_param.numerical_sol.convergence_criterion, ...
78          sim_param.numerical_sol.max_NR_iterations) ;
```

Figure 35 - The code of stage (1) of the numerical solution

Stage II:

In this stage, preprocessing is used to improve the running time. Since the time vector contains multiple cycles, the stiffness matrix would exhibit periodicity if calculated for every time step. To avoid calculating

the modified stiffness matrix and its inverse for each time step, the modified stiffness matrix (line 81, **Figure 36**) and its inverse (lines 83-87, **Figure 36**) are calculated within the cycle. Subsequently, in stage (3), the Newmark method extracts the appropriate stiffness matrix and its inverse for each time step. This preprocessing stage significantly contributes to reducing computation time, as the inverse operation is computationally expensive. Moreover, the number of points in the cycle vector can be much smaller compared to the time step, for example, the cycle may contain 44,764 points while the time vector has 2,209,320 points - approximately 50 times larger.

```
80        %% step 4: prepare inverse modified K matrix %%
81        K_modified_cyc = K_mtx_cyc + repmat(((gamma/(beta*dt))*C + (1/(beta*dt^2))*M), 1, 1, len_cyc);
82        P0 = zeros(ndof,1) ;
83        inv_K_modified_cyc = zeros(size(K_modified_cyc)) ;
84
85        for ii = 1:len_cyc
86            inv_K_modified_cyc(:, :, ii) = inv(K_modified_cyc(:, :, ii)) ;
87        end % of for ii
```

Figure 36 - The code of stage (2) of the numerical solution

Stage III:

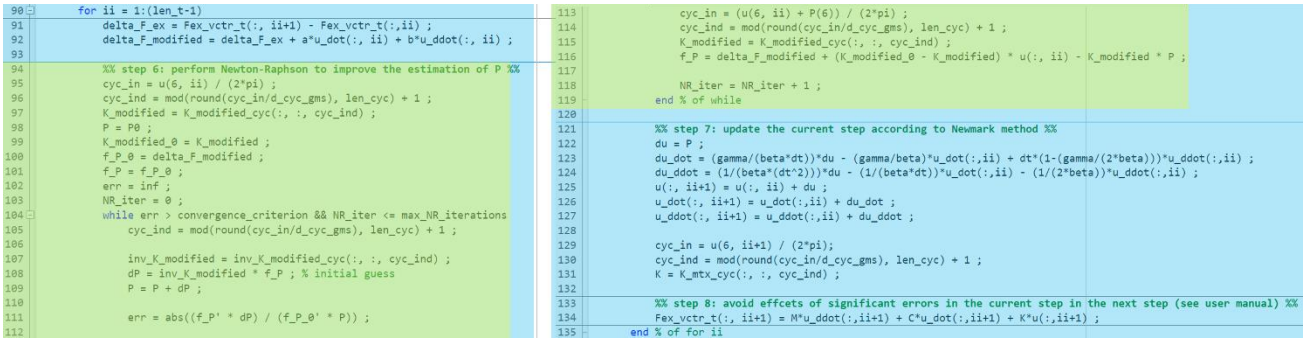
In this stage, the Newmark method is applied to calculate $u_{i+1}, \dot{u}_{i+1}, \ddot{u}_{i+1}$ given the previous values $u_i, \dot{u}_i, \ddot{u}_i$, the modified stiffness matrix of u_i (denoted as $\hat{K}_i$), and the modified external forces vector u_i (denoted as $\hat{F}_i$). Within stage (3), stage (4) involves the use of the Newton-Raphson method to improve the calculation of a specific step during the Newmark method process, where Δu_i is determined by the following relations:

$$\Delta u_i = \hat{K}_i^{-1} \Delta \hat{F}_i$$

As shown in **Figure 37**, the Newmark method iterates over time and calculates $u_{i+1}, \dot{u}_{i+1}, \ddot{u}_{i+1}$ using the following steps:

1. The function calculates the modified external force $\hat{F}_i$ (lines 91-92, **Figure 37**) and extracts the appropriate modified stiffness matrices $\hat{K}_i$ which was calculated in stage (2).
2. The function solves $\Delta u_i = \hat{K}_i^{-1} \Delta \hat{F}_i$ using Newton Raphson iterations [stage (4)].
3. The function calculates $\Delta u_i, \Delta \dot{u}_i, \Delta \ddot{u}_i$ and then update $u_{i+1}, \dot{u}_{i+1}, \ddot{u}_{i+1}$.
4. The function updates F_i to prevent cumulative errors in the solution of the Euler-Lagrange equations. Without this line, if there is a significant error in the solution of the equations at a former time step, it would persist throughout the remaining solution, leading to a highly inaccurate result. This line ensures that the solution "forgets" the error from the previous step, effectively eliminating accumulated errors and improving the accuracy of the solution.

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



```
for ii = 1:(len_t-1)
    delta_F_ex = Fex_vctr_t(:, ii+1) - Fex_vctr_t(:,ii) ;
    delta_F_modified = delta_F_ex + a*u_dot(:, ii) + b*u_ddot(:, ii) ;

    %% step 6: perform Newton-Raphson to improve the estimation of P %%
    cyc_in = u(6, ii) / (2*pi) ;
    cyc_ind = mod(round(cyc_in/d_cyc_gms), len_cyc) + 1 ;
    K_modified = K_modified_cyc(:, :, cyc_ind) ;
    P = P0 ;
    K_modified_0 = K_modified ;
    f_P_0 = delta_F_modified ;
    f_P = f_P_0 ;
    err = inf ;
    NR_iter = 0 ;
    while err > convergence_criterion && NR_iter <= max_NR_iterations
        cyc_ind = mod(round(cyc_in/d_cyc_gms), len_cyc) + 1 ;

        inv_K_modified = inv_K_modified_cyc(:, :, cyc_ind) ;
        dP = inv_K_modified * f_P ; % initial guess
        P = P + dP ;

        err = abs((f_P' * dP) / (f_P_0' * P)) ;

        cyc_in = (u(6, ii) + P(6)) / (2*pi) ;
        cyc_ind = mod(round(cyc_in/d_cyc_gms), len_cyc) + 1 ;
        K_modified = K_modified_cyc(:, :, cyc_ind) ;
        f_P = delta_F_modified + (K_modified_0 - K_modified) * u(:, ii) - K_modified * P ;

        NR_iter = NR_iter + 1 ;
    end % of while

    %% step 7: update the current step according to Newmark method %%
    du = P ;
    du_dot = (gamma/(beta*dt))*du - (gamma/beta)*u_dot(:,ii) + dt*(1-(gamma/(2*beta)))*u_ddot(:,ii) ;
    du_ddot = (1/(beta*(dt^2)))*du - (1/(beta*dt))*u_dot(:,ii) - (1/(2*beta))*u_ddot(:,ii) ;
    u(:, ii+1) = u(:, ii) + du ;
    u_dot(:, ii+1) = u_dot(:,ii) + du_dot ;
    u_ddot(:, ii+1) = u_ddot(:,ii) + du_ddot ;

    cyc_in = u(6, ii+1) / (2*pi);
    cyc_ind = mod(round(cyc_in/d_cyc_gms), len_cyc) + 1 ;
    K = K_mtx_cyc(:, :, cyc_ind) ;

    %% step 8: avoid effcets of significant errors in the current step in the next step (see user manual) %%
    Fex_vctr_t(:, ii+1) = M*u_ddot(:,ii+1) + C*u_dot(:,ii+1) + K*u(:,ii+1) ;
end % of for ii
```

Figure 37 - Stages (3) and (4). In stage (3), marked by blue, Newmark method is applied, and in stage (4), marked by yellow, Newton-Raphson iterations are applied.

Stage IV:

In this stage, Newton-Raphson iterations are employed to refine the solution of $\widehat{K}_i \Delta u_i - \Delta \widehat{F}_i = 0$, which corresponds to Step 2 of the Newmark method in stage (3). Due to the large values of $\widehat{K}_i$, directly solving $\Delta u_i = \widehat{K}_i^{-1} \Delta \widehat{F}_i$ can lead to inaccuracies. Thus, in the current case, the Newton-Raphson method calculates a solution P_i for the equation $f(P) = \widehat{K_1}\, P - \Delta \widehat{F}_i = 0$. In each iteration, it computes $dP_{i,j}$ to refine the equation, aiming to make $f(P_j + dP_j = P_{j+1}) = \widehat{K_1}\,(P_j + dP_j) - \Delta \widehat{F}_i$ closer to zero. The algorithm starts with P_1 set to zero and then refines the solution through the following iterations:

1. The function calculates $P_{j+1} = P_j + dP_j$ where $dP_j = \widehat{K}_i^{-1} f(P_j)$.
2. The function updates $\widehat{K}_i$ (denoted here as $\widehat{K}_{i,j}$). This update is necessary because Δu_i equals P_{end}, so any changes in P_j along the iterations will result in changes in Δu_i in the end, requiring the appropriate update of $\widehat{K}_{i,j}$.
3. The new residual $f(P_{j+1})$ is calculated as $f(P_j + dP_j = P_{j+1}) = \widehat{K}_{i,j}\,(P_j + dP_j) - \Delta \widehat{F}_i$. The term "(K_modified_0 - K_modified) * u(:, ii)" is used to update $\Delta \widehat{F}_i$ because changes in $\widehat{K}_{i,j}$ also affect it.
4. The function continues the iteration in the while loop until either the error of the solution is sufficiently small or the maximum number of Newton-Raphson iterations is reached.

It is worth mentioning that several algebraic developments are necessary to derive the Newton-Raphson equations implemented in the code lines of the numerical solution based on the standard steps on Newton-Raphson.

`resampling`

The function resampling resamples "*x_old*" with number of new points "*Npt_new*".

uniform_rand

The function uniform_rand generates a vector with a systematic error of "*systematic_err*" and random values distributed uniformly in $[-\text{statistic\_err}, +\text{statistic\_err}]$.