The Pillar Model PyRolL Plugin

Max Weiner, Christoph Renzing

May 26, 2025

The Pillar Model plugin serves as a base package for other plugins using the division of the profile in collinear pillars along width direction. It is mainly intended for calculation of material flow, groove filling and spread.

1 Model Approach

1.1 Discretization of Profile Cross-Sections

The pillar model introduces a discretization of the profile cross-section into distinct equidistant pillars as shown in Figure 1. Another possibility is the discretization into uniform pillars shown in Figure 2.

Equidistant pillars The pillars' positions are defined by their center points z_i. $i \in [0, n-1]$ is the index of the pillar, with n as the count of pillars. Each pillar has a defined width w_i and height h_i. The height is always measured at z_i.

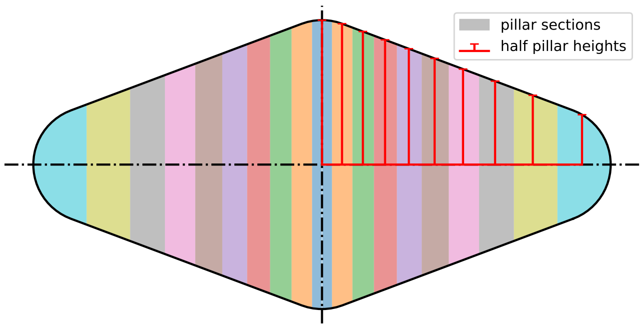

Figure 1: Example of Pillar Discretization for a Diamond Profile With 5 Pillars (Symmetrical, equidistant)

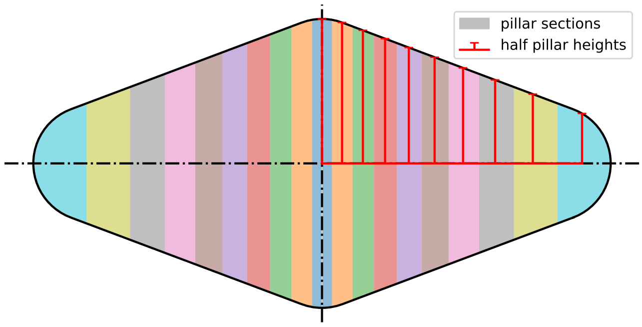

Figure 2: Example of Pillar Discretization for a Diamond Profile With 5 Pillars (Symmetrical, uniform)

When discretizing an existing profile, the z_i are calculated as in Equation 1, with w as the maximum profile width. However, during deformation the positions may change, so one shall not depend on equidistant pillars.

$$z_i = i\Delta z \qquad \text{with } \Delta z = \frac{w}{n - \frac{1}{2}} \tag{1}$$

Uniform pillars The pillars' positions are defined by their center points z_i. $i \in [0, n-1]$ is the index of the pillar, with n as the count of pillars. Each pillar has a defined width w_i and height h_i. The height is always measured at z_i.

When discretizing an existing profile, the z_i are calculated by solving the system of equations in Equation 2, with w as the maximum profile width.

$$\begin{aligned} 0 &= A_i - A_{i+1} \\ 0 &= \frac{w}{2} - \sum_i^n w_i \end{aligned} \tag{2}$$

The boundaries Z_j with $j \in [0, n]$ of the pillars are located halfway between their positions, but the innermost boundary is identical with the position $z_0 = Z_0 = 0$.

$$Z_j = \frac{z_j - z_{j-1}}{2} \tag{3}$$

The pillar widths w_i are defined as the distance between their boundaries. Note, that the numerical width of the inner pillar w_0 is initially the half of the others due to $z_0 = Z_0 = 0$.

$$w_i = Z_{i+1} - Z_i \tag{4}$$

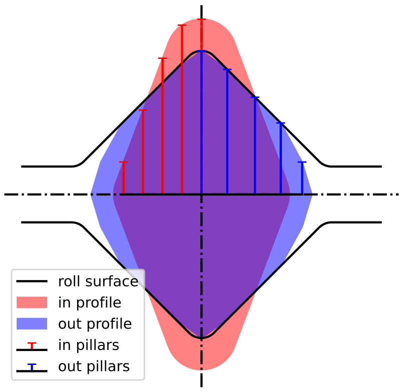

Figure 3: Deformation of a Pillared Profile in a Disk Element of a Roll Pass

1.2 Pillar Behavior in Roll Passes

The main purpose of the pillar model is to model geometry evolution in the disk elements of roll passes. In the following, the upper indizes 0 and 1 denote the incoming resp. outgoing profile's values.

Figure 3 shows the intended deformation of a pillar-discretized profile within a roll pass disk element. Note, that the spreading in the figure is heavily exaggerated for illustration purpose.

The incoming profile shows here a smooth surface, as it was generated by `Profile.diamond()`, the same way as the profile in Figure 1. The outgoing profile has lost its smooth surface, since the cross-section shape can only be described by the use of the pillars positions z_i^1 and heights h_i^1, so the cross-section becomes a linear line string. Additional points are added at $(Z_n, 0)$ and $(-Z_n, 0)$ to provide a senseful outer boundary and ensure consistency with the pillar creation described above. Note, that only the pillars 1–3 (from center to side) are here in contact with the roll surface, as their height is larger than the local roll pass height. Only they are experiencing deformation, with reduction in height and increase in width. The other pillars are shifted to the outside according to the others' widths, but maintain their heights.

The new pillar widths w_i^1 are calculated using their spread β_i. β_i defaults to 1, actual implementations are deferred to other plugins.

$$w_i^1 = \beta_i w_i^0 \tag{5}$$

The new boundaries result from this in a cumulative way as:

$$Z_{j+1}^1 = Z_j^1 + w_j \qquad \text{with } Z_0^1 = 0 \tag{6}$$

And therefore the pillar position z_i^1 as:

$$z_i^1 = \frac{Z_{i+1}^1 - Z_i^1}{2} \tag{7}$$

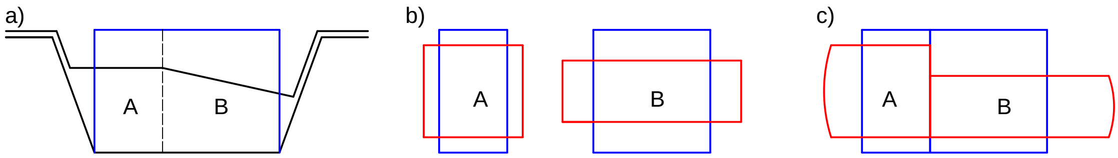

Figure 4: Schmeatic irregular roll pass, a) irregular pass; b) assumed deformation of two pillar elements; c) combined deformation of two pillar elements assuming unhindered spreading

1.3 Elongation Correction

As described in the previous subsections, the movement of one pillar is treated independent of its neighbouring pillars. Since this is not the case in realty due to the general material preservation, the movement of a pillar element also influences the movement of it's neighbouring elements. **Neumann1975** described the mean elongation as the resulting elongation of the profile when internal material cohesion is considered. The first investigation regarding this topic was done by **Lendl1948**. Resulting from his investigation was the already stated method of elongation correction. This method couples the elongation of the single pillar elements using a mean elongation λ_m. A comprehensive comparison and analysis for calculation of the mean elongation, has been done by **Wusatowski1957**. The author investigated seven different equations as well as the underlying assumptions and compared the resulting elongations with rolling trials. Schematically, this is represented in the following Figure 4.

What can be seen in the figure, is the different draughts of the sub profiles A and B. Due the higher draught which subprofile A is exposed too, the elongation of this profile has to be higher according to the volume preservation. The investigations of **Wusatowski1957** resulted that the equation developed by Gorecki et al. yields the most accurate results, since it was directly mathematically derived to fulfill the condition of volume preservation.

$$\lambda_m = \frac{\sum A_i}{\sum \left(\frac{A_i}{\lambda_i}\right)} \tag{8}$$

To calculate the corresponding elongations for every pillar element in a disk element, a fix-point iteration scheme is applied. For the first iteration loop, the elongation coefficients are set to 1, so the volume preservation is evaluated globally.
In the next loop, the correction coefficients are updated using the following Equation 9:

$$\lambda_{a,i} = \frac{1}{\lambda_m \sum_{i=1}^{n} \gamma_i \sum_{i=1}^{n} \beta_i} \tag{9}$$

Calculation of the coefficients is finally done using Equation 10.

$$\lambda_{a,i} = \frac{\lambda_{a,j} + (\lambda_{a,j} \cdot \lambda_{a,i} - \lambda_{a,j}) \cdot A}{n} \tag{10}$$

It can be seen, that the correction coefficients of the previous iteration loop $\lambda_{a,j}$ are being updated by the coefficients of the current loop $\lambda_{a,i}$. Dividing the calculated coefficients using the total number of disk elements n yields the current local correction coefficients.
Utilizing a relaxation Factor A stabilizes the algorithm.

2 Usage Instructions and Implementation Details

Packages residing on this type of discretization shall depend on this plugin to create a common interface. In the following, the hooks defined by this plugin shall be described, and instructions shall be given, how to implement model equations for specific purposes.

The pillar model plugin defines hooks on `pyroll.core.Profile` for the purpose of pillar representation and initial creating as described in subsection 1.1. The central hook of this package is `Profile.pillars`. It returns a numpy array of the z coordinates representing the centers of the pillars acc. to Equation 7. The respective pillar boundaries' coordinates acc. to Equation 3 are provided by the `Profile.pillar_boundaries` hook. The heights h_i are provided by the `Profile.pillar_heights` hook, as the widths by the `Profile.pillar_widths` hook. General implementations of these hooks are provided according to Equation 7, Equation 3 and Equation 4. The heights are determined by intersection of the profiles cross-section polygon with vertical lines at the respective z_i. Polygons representing the pillars are provided by the `Profile.pillar_sections` hook, obtained by clipping the profile's cross-section.

To modify the initial pillar distancing, provide a new implementation of `Profile.pillars`. Users of this plugin must not rely on equidistant pillars, but calculate needed distances from the respective coordinates. Especially in roll passes the pillar positions *will* change due to spreading calculation, if a respective plugin is loaded additionally.

Regarding the behavior of pillars in roll passes the following hooks are defined:

`RollPass.DiskElement.pillars_in_contact` returns a boolean array indicating which pillars have contact to the roll surface in this disk element.

`RollPass.DiskElement.pillar_spreads` returns an array with the pillars' spreads β_i. Defaults to ones for all pillars.

`RollPass.DiskElement.pillar_draughts` returns an array with the pillars' draughts γ_i. Defaults to h_i^1/h_i^0.

`RollPass.DiskElement.pillar_spreads` returns an array with the pillars' elongations λ_i. Defaults to $(\beta_i \gamma_i)^{-1}$.

Implementations of the profile hooks above on `RollPass.DiskElement` are provided following the procedure in subsection 1.2.

Plugins aiming at spread calculation using the pillar approach should provide an implementation of `RollPass.DiskElement.pillar_spreads` yielding the respective values. Users of PyRolL (non-developers) generally do not need to provide anything for usage of this plugin, except they may set the `pyroll.pillar_model.PILLAR_COUNT` constant to a desired value. The value must be a non-negative integer, it is explicitly recommended to not change this during simulation runs, as it may corrupt data already generated.

For implementing additional model equations, define a new hook `Profile.pillar_*s` which shall return an array of the same length as `Profile.rings`. The `*` shall be replaced with the property name you want to represent, pay respect to the plural form. For hooks on units or disk elements act respectively. The elongation correction is enabled by default, one can control the behavior using the `ELONGATION_CORRECTION` variable in the plugin `CONFIG`.