# The Pillar Model PyRoll Plugin

Max Weiner

February 2, 2023

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 pillars as shown in Figure 1. 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 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.

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

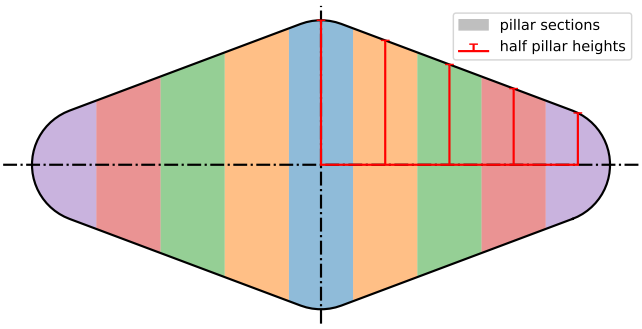


Figure 1: Example of Pillar Discretization for a Diamond Profile With 5 Pillars (Symmetrical)

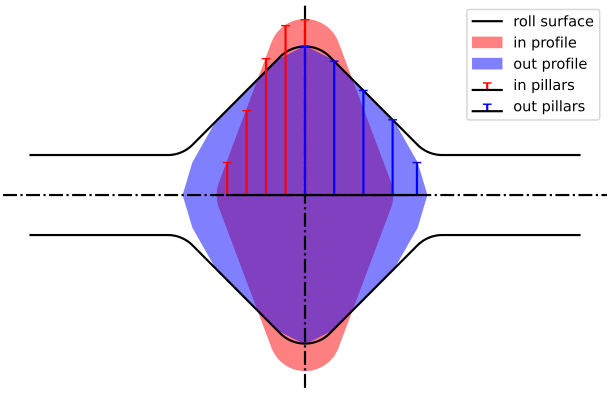


Figure 2: Deformation of a Pillared Profile in a Disk Element of a Roll Pass

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 $z_0 = Z_0 = 0$.

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

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 $z_0 = Z_0 = 0$.

$$w_i = Z_{i+1} - Z_i \tag{3}$$

## 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 2 shows the intended deformation of a pillar-discretized profile within a roll pass disk element. Note, that the spreading in the figure is heavily exagerated 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 $\beta_i$. $\beta_i$ defaults to 1, actual implementations are deferred to other plugins.

$$w_i^1 = \beta_i w_i^0 \tag{4}$$

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

$$Z_{j+1}^1 = Z_j^1 + w_j \qquad \text{with } Z_0^1 = 0 \tag{5}$$

And therefore the pillar position $z_i^1$ as:

$$z_i^1 = \frac{Z_{i+1}^1 - Z_i^1}{2} \tag{6}$$

## 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 1. The respective pillar boundaries' coordinates acc. to Equation 2 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 1, Equation 2 and Equation 3. 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 not rely on equidistant pillars, but calculate needed distances from the respective coordinates. Especially in roll passes the pillar posotions *will* change due to spreading calculation, if an respective plugin is loaded additionally.

Reagrding 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 $\beta_i$. Defaults to ones for all pillars.

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

`RollPass.DiskElement.pillar_spreads` returns an array with the pillars' elongations $\lambda_i$. 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 approad 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.