The Ring Model PyRoll Plugin

Max Weiner

January 30, 2023

The Ring Model plugin serves as a base package for other plugin using an 1D finite differences approach for modelling inhomogeneous workpiece state.

1 Model approach

The ring model introduces a gradient in radial direction from core to surface of the profile while neglecting gradients in angular direction. The actual profile cross-section is mapped to an equivalent round profile, which is divided in concentric rings, as illustrated in Figure 1. In the implementation provided the equivalent radius r_e is defined by Equation 1 with the profile cross-section area A.

$$r_\mathrm{e} = \sqrt{\frac{A}{\pi}} \tag{1}$$

The equivalent round is divided in equidistant concentric rings. The ring distance is Δr. The core ring is a circle with center in $(0, 0)$ and radius $\frac{\Delta r}{2}$. Therefore, Δr is calculated as in Equation 2 with the count of rings n.

$$\Delta r = \frac{r_\mathrm{e}}{n - \frac{1}{2}} \tag{2}$$

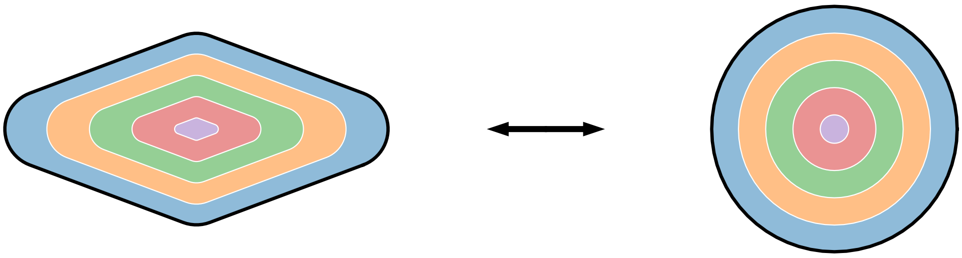

Figure 1: Example of Ring Discretization for an Diamond Profile and its Equivalent Round Profile

The ring center lines' radii are as in Equation 3 with the index of the ring $i \in [0, n-1]$.

$$r_i = i\Delta r \tag{3}$$

The radii of the ring boundary lines are as in Equation 4 with the index of the boundary $j \in [0, n]$.

$$R_j = \left(j - \frac{1}{2}\right)\Delta r \tag{4a}$$

$$R_0 = 0 \tag{4b}$$

The rings are mapped back to the actual cross-section by scaling the polar coordinates of the contour with the ring radii as in Equation 5, with the polar coordinates of the profile contour line (θ, ϱ). $r_{\mathrm{P}i}(\theta)$ are the polar coordinates of the ring center lines on the actual profile cross-section like illustrated on the left of Figure 1. The boundaries' coordinates are calculated accordingly.

$$r_{\mathrm{P}i}(\theta) = \varrho(\theta)\frac{r_i}{r_\mathrm{e}} \tag{5a}$$

$$R_{\mathrm{P}j}(\theta) = \varrho(\theta)\frac{R_j}{r_\mathrm{e}} \tag{5b}$$

The length of the mapped boundary lines is defined as the interface length I_j. The area between the mapped boundaries is the ring cross-section A_i.

2 Usage instructions

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 equation for specific purposes.

The ring model plugin only defines hooks on `pyroll.core.Profile`. The central hook of this package is `Profile.rings`. It returns a numpy array of radius coordinates representing the centers of the rings acc. to Equation 3. The respective ring boundaries' coordinates acc. to Equation 4 are provided by the `Profile.ring_boundaries` hook. The outermost boundary equals the equivalent radius acc. to Equation 1 of the profile cross-section, which is also available from the `Profile.equivalent_radius` hook. To modify the calculation of the equivalent radius, provide a new implementation of the `Profile.equivalent_radius` hook. To modify the ring distancing, both `Profile.rings` and `Profile.ring_boundaries` should be provided. Users of this plugin should not rely on equidistant rings, but calculate needed distances from the respective coordinates.

For geometric calculations, also two hooks returning shapely geometry objects are provided. `Profile.ring_contours` returns an array of `LineStrings` representing the mapping of the ring boundaries on the real profile cross-section. The interface lengths I_j can be obtained by using `Profile.ring_contours[j].length`. `Profile.ring_sections` returns, however, an array of `Polygon` objects representing the areas of the rings lying between those boundaries. The ring cross-sections A_i can be obtained by using `Profile.ring_sections[i].area`.

For implementing model equations, define a new hook `Profile.ring_*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. Ideally a hook named accordingly exists on `Profile` or is defined alongside, which returns a mean value of the inhomogeneous data. For example the core hook `Profile.temperature` may be accompanied by `Profile.ring_temperatures` to hold inhomogeneous temperature data.