MR Sim

Nov 07, 2022

Contents:

MR Sim is a library for simulating material removal from abrasive material removal processes. The library is built to be modular to allow different models, tool types, tool shapes, and pressure distributions to be simulated and included.

CHAPTER 1

Example 1

This example simulates a round orbital sander on a flat surface with a sinsoidal motion in the X direction. The sander is set to have orbital speed, but no rotational speed.

```python
from context import mr_sim
import numpy as np
import matplotlib.pyplot as plt

R = 3.5 / 2 * 25.4 / 1000
length = 208 / 1000
period = 0.2 * 20
dt = period / 1000
dt = dt / 4
amp = length / 2

Simulation = mr_sim.create_simulation(
    mr_sim.Round, mr_sim.Flat, mr_sim.Orbital, mr_sim.Preston
)

simulation = Simulation(
    length + 2 * R,
    2 * R,
    kp=1.362e-9,
    radius=R,
    eccentricity=0.1875 * 25.4 / 1000,
    dt=dt,
    auto_velocity=True,
)

simulation.set_speed(620)
simulation.set_force(15)

for t in np.arange(0, period / 2, dt):
    simulation.set_location(amp * np.cos(2 * t * np.pi / period))
    simulation.step()
```

(continues on next page)

(continued from previous page)

```

plt.figure()
simulation.plot()
plt.show()
```

CHAPTER 2

Example 2

This example simulates a purely rotary tool on a flat surface traveling on a Peano Weaving path.

```
from context import mr_sim
import numpy as np
from hilbertcurve.hilbertcurve import HilbertCurve
import matplotlib.pyplot as plt

R = 0.1
spacing = 1.75 * R
velocity = 0.1
dt = 0.05

Simulation = mr_sim.create_simulation(
    mr_sim.Round, mr_sim.Flat, mr_sim.Rotary, mr_sim.Preston
)

curve = HilbertCurve(3, 2)

points = np.array(
    [curve.coordinates_from_distance(d) for d in range(curve.max_h + 1)], np.float
)
points -= curve.max_x / 2
points *= spacing

t_final = curve.max_h * spacing / velocity
point_times = np.linspace(0, t_final, points.shape[0])
t = np.arange(0, t_final, dt)

path = np.vstack(
    (np.interp(t, point_times, points[:, 0]), np.interp(t, point_times, points[:, 1]))
).T

size = spacing * curve.max_x + 2 * R
```

(continues on next page)

(continued from previous page)

```
simulation = Simulation(
    size, size, radius=R, dt=dt, dx=0.005, dy=0.005, auto_velocity=True
)

simulation.set_speed(100)
simulation.set_force(5)

for location in path:
    simulation.set_location(*location)
    simulation.step()

plt.figure()
simulation.plot()
plt.plot(points[:, 0], points[:, 1], ":w", linewidth=0.5)
plt.show()
```

CHAPTER 3

Util

`mr_sim.`**`create_simulation`**`(`*`*classes`*`)`

A helper method to combine classes.

This method combines classes representing different elements of a material removal simulation. This allows different simulations to be easily constructed.

Parameters **`*classes`** – Variable length argument list of classes to subclass.

Returns A class which is a subclass of all classes provided as arguments.

Examples

For example, to use the Preston equation to simulate a round orbital sander sanding a flat surface, the following can be used.

```
>>> from mr_sim import *
>>> Simulation = create_simulation(Flat, Round, Orbital, Preston)
>>> simulation = Simulation(0.3, 0.1, radius=0.05, eccentricity=0.005, dt=0.001,␣
↪kp=1e-9)
```

In this example `radius` sets the radius of the sander defined in the `Round` class, `eccentricity` sets the eccentricity of the orbital sander defined in the `Orbital` class, `dt` sets the timestep in the `Base` class, which is a superclass of all other classes, and `kp` sets the constant in the Preston Equation class, `Preston`.

The same result can be accomplished manually.

```
>>> from mr_sim import *
>>> Simulation(Flat, Round, Orbital, Preston):
>>>     pass
>>> simulation = Simulation(0.3, 0.1, radius=0.05, eccentricity=0.005, dt=0.001,␣
↪kp=1e-9)
```

This approach would be particularly useful if additional functionality needs to be added to the simulation class.

CHAPTER 4

Base

class mr_sim.base.**Base**(*size_x*, *size_y*, *dx=0.001*, *dy=0.001*, *dt=1*, *auto_velocity=False*)

The base simulation class.

This is the base simulation class. It is combined with other classes to define other attributes about the simulation.

X

A 2D array of the X coordinates of the part with the origin at the center of the part surface.

Type numpy.ndarray

Y

A 2D array of the Y coordinates of the part with the origin at the center of the part surface.

Type numpy.ndarray

profile

A 2D array of the depth of material removed from the part surface.

Type numpy.ndarray

dt

The timestep used in the simulation.

Type float

auto_vel

If `True` automatically calculate the linear velocity of the tool moving over the part surface.

Type bool

x

The current X location of the center of the tool in relation to the center of the part surface.

Type float

y

The current Y location of the center of the tool in relation to the center of the part surface.

Type float

`vl_x`

The current X direction velocity of the tool moving over the part surface.

Type float

`vl_y`

The current Y direction velocity of the tool moving over the part surface.

Type float

Note: Other classes must be used to include the `shape` and `mrr` methods needed by this class.

Parameters

- **`size_x`** (*`float`*) – The size of the part surface in the X direction.
- **`size_y`** (*`float`*) – The size of the part surface in the Y direction.
- **`dx`** (*`float`*) – The density of points to track in the X direction. Defaults to 0.001.
- **`dy`** (*`float`*) – The density of points to track in the Y direction. Defaults to 0.001.
- **`dt`** (*`float`*) – The simulation timestep. Defaults to 1.
- **`auto_velocity`** (*`bool`*) – If `True` automatically calculate the linear velocity of the tool over the part surface.

`local_grid`()

Returns a coordinate system centered at the tool origin.

Returns The X, Y coordinate system shifted so the origin is at the center of the tool.

Return type numpy.ndarray, numpy.ndarray

`plot`(*normalize=False*, ***kwargs*)

Plot the simulation result.

This function uses matplotlib to plot the result of the simulation in the current figure.

Parameters

- **`normalize`** (*`bool`*) – Normalize max depth of material removed to 1. Defaults to `False`.
- **`**kwargs`** – Keyword arguments to be included in the call to the `matplotlib.pyplot.imshow` function.

Returns The `AxesImage` object of the plotted heatmap.

Return type matplotlib.image.AxesImage

`set_location`(*x=0*, *y=0*)

Set the current location of the center of the tool.

Sets the current location of the center of the tool with respect to the middle of the part.

Parameters

- **`x`** (*`float`*) – The X location of the tool. Defaults to 0.
- **`y`** (*`float`*) – The Y location of the tool. Defaults to 0.

`set_velocity`(*x=0*, *y=0*)

Sets the current linear velocity of the tool over the part surface.

Parameters

- **`x`** (*`float`*) – The X velocity of the tool. Defaults to 0.
- **`y`** (*`float`*) – The Y velocity of the tool. Defaults to 0.

Note: If `auto_velocity` was set to `True` this method is not requried.

`step``()`

Move the simulation forward one timestep.

CHAPTER 5

Models

class mr_sim.models.**Preston**(**args*, *kp=1*, ***kwargs*)

A class implementing the Preston Equation for material removal simulation.

This class calculates material removal rate using the Preston Equation,

$\dot{h} = k_p p v$,

where $\dot{h}$ is the depth of material removed per unit time, k_p is a constant known as the Preston coefficient, p is the contact pressure between the tool and the workpiece, and v is the total speed of the tool rubbing the workpiece.

Note: This class must be subclassed by a class providing the `pressure` and `velocity` functions.

kp

The value of the Preston coefficient, k_p.

Type float

Parameters

- ***args** – Arguments to be passed on to superclasses.
- **kp** (*float*) – The Preston coefficient, k_p. Defaults to 1.
- ****kwargs** – Keyword arguments to be passed on to superclasses.

mrr(*x*, *y*)

Calculates the material removal rate.

This function returns the material removal rate for all locations on the part surface using the Preston Equation.

x (numpy.ndarray): A 2D array of the X coordinates of the part centered at the current tool location.

y (numpy.ndarray): A 2D array of the Y coordinates of the part centered at the current tool location.

Returns The material removal rate at all locations on the part surface.

Return type numpy.ndarray

CHAPTER 6

Pressure Distributions

class `mr_sim.pressure.`**`Flat`**(**args*, ***kwargs*)

A class used to calculate the pressure applied to a flat surface.

This class determines the pressure applied to a flat surface from the normal force and torques applied to the tool.

Note: This class assumes that all points of the tool are in contact with the part surface at all time.

`force`

The normal force applied to the tool.

Type float

`torque_x`

The torque about the X axis applied to the tool.

Type float

`torque_y`

The torque about the Y axis applied to the tool.

Type float

Note: This class requires `area`, `Ix`, and `Iy` to be set by a subclass.

Parameters

- **`*args`** – Arguments to be passed on to superclasses.
- **`**kwargs`** – Keyword arguments to be passed on to superclasses.

`pressure`(*x*, *y*)

Determine the pressure the tool applied to the part surface.

This function calculates the pressure applied at all points on the part surface.

Parameters

- **x** (*numpy.ndarray*) – A 2D array of the X coordinates of the part centered at the current tool location.
- **y** (*numpy.ndarray*) – A 2D array of the Y coordinates of the part centered at the current tool location.

Returns A 2D array of the pressure applied by the tool.

Return type numpy.ndarray

set_force (*force*)
Set the current normal force applied to the tool.

Parameters **force** (*float*) – The current normal force.

set_torque (*x=0*, *y=0*)
Set the current torques applied to the tool.

Parameters

- **x** (*float*) – The current torque about the X axis. Defaults to 0.
- **y** (*float*) – The current torque about the Y axis. Defaults to 0.

class mr_sim.pressure.**ConstantCurvature** (**args*, *kx=0*, *ky=0*, *stiffness=None*, *dx=0.001*, *dy=0.001*, ***kwargs*)
A class used to calculate the pressure applied to a surface with constant curvature.

This class determines the pressure applied to the surface from the normal force.

Note: This class models the contact pressure as a paraboloid with the specified curvature at the tool origin.

force
The normal force applied to the tool.

Type float

kx
The curvature in the x direction.

Type float

ky
The curvature in the y direction.

Type float

dx
The x direction spacing of the grid.

Type float

dy
The y direction spacing of the grid.

Type float

stiffness
The stiffness of the tool. This can be found as the Young's modulus of the tool divided by its thickness.

Type float

Parameters

- ***args** – Arguments to be passed on to superclasses.
- **kx** (*float*) – The curvature in the x direction. Defaults to 0.
- **ky** (*float*) – The curvature in the y direction. Defaults to 0.
- **stiffness** (*float*) – The stiffness of the sanding tool. Defaults to `None`. This can be found as the Young's modulus of the tool divided by its thickness.
- **dx** (*float*) – The x direction spacing of the grid. Defaults to 0.001.
- **dy** (*float*) – The y direction spacing of the grid. Defaults to 0.001.
- ****kwargs** – Keyword arguments to be passed on to superclasses.

Raises `ValueError` – If `stiffness` is `None`.

pressure (*x*, *y*)

Determine the pressure the tool applied to the part surface.

This function calculates the pressure applied at all points on the part surface.

Parameters

- **x** (*numpy.ndarray*) – A 2D array of the X coordinates of the part centered at the current tool location.
- **y** (*numpy.ndarray*) – A 2D array of the Y coordinates of the part centered at the current tool location.

Returns A 2D array of the pressure applied by the tool.

Return type numpy.ndarray

Note: This function uses closed form solutions if the shape is *Round*, otherwise it uses `scipy.optimize.minimize_scalar()` which is slow.

set_curvature (*kx=0*, *ky=0*)

Set the curvature of the surface.

Parameters

- **kx** (*float*) – The curvature in the x direction. Defaults to 0.
- **ky** (*float*) – The curvature in the y direction. Defaults to 0.

set_force (*force*)

Set the current normal force applied to the tool.

Parameters **force** (*float*) – The current normal force.

CHAPTER 7

Tool Shapes

class mr_sim.shapes.**Round**(**args*, *radius=None*, ***kwargs*)

A class representing a round tool for material removal simulations.

This class represents a round tool and calculates the section of the part surface which the tool is in contact with.

r

The tool radius.

Type float

area

The area of the tool.

Type float

Ix

The second moment of area of the tool in the X direction.

Type float

Iy

The second moment of area of the tool in the Y direction.

Type float

Note: The class attributes are set automatically when the class is initialized, and when the tool radius is set using the `radius` property. Setting these values manually could lead to simulation errors.

Parameters

- ***args** – Arguments to be passed on to superclasses.
- **radius** (*float*) – The radius of the tool. Defaults to `None`.
- ****kwargs** – Keyword arguments to be passed on to superclasses.

Raises `ValueError` – If `radius` is not set.

radius
Tool radius.

Type float

shape (*x*, *y*)
This function finds the section of the part the tool is in contact with.

Parameters

- **x** (*numpy.ndarray*) – A 2D array of the X coordinates of the part centered at the current tool location.
- **y** (*numpy.ndarray*) – A 2D array of the Y coordinates of the part centered at the current tool location.

Returns A 2D array where `True` indicates the tool is in contact with that portion of the part surface.

Return type numpy.ndarray

class mr_sim.shapes.**Rectangular**(**args*, *width=None*, *height=None*, ***kwargs*)
A class representing a rectangular tool for material removal simulations.

This class represents a rectangular tool and calculates the section of the part surface which the tool is in contact with.

width
The tool width, in the X direction.

Type float

height
The tool height, in the Y direction.

Type float

area
The area of the tool.

Type float

Ix
The second moment of area of the tool in the X direction.

Type float

Iy
The second moment of area of the tool in the Y direction.

Type float

Note: The class attributes are set automatically when the class is initialized, and when the tool size is set using the `set_size` method. Setting these values manually could lead to simulation errors.

Parameters

- ***args** – Arguments to be passed on to superclasses.
- **width** (*float*) – The width of the tool, in the X direction. Defaults to `None`.

- **`height`** (*`float`*) – The height of the tool, in the Y direction. Defaults to `None`.
- **`**kwargs`** – Keyword arguments to be passed on to superclasses.

Raises `ValueError` – If `width` or `height` is not set.

`set_size`(*width*, *height*)
Set the size of the tool.

Parameters

- **`width`** (*`float`*) – The width of the tool, in the X direction.
- **`height`** (*`float`*) – The height of the tool, in the Y direction.

`shape`(*x*, *y*)
This function finds the section of the part the tool is in contact with.

Parameters

- **`x`** (*`numpy.ndarray`*) – A 2D array of the X coordinates of the part centered at the current tool location.
- **`y`** (*`numpy.ndarray`*) – A 2D array of the Y coordinates of the part centered at the current tool location.

Returns A 2D array where `True` indicates the tool is in contact with that portion of the part surface.

Return type numpy.ndarray

class `mr_sim.shapes.`**`Square`**(**args*, *width=None*, ***kwargs*)
A class representing a square tool for material removal simulations.

This class represents a square tool and calculates the section of the part surface which the tool is in contact with.

Note: This class simply inherits *`Rectangular`* and sets both `width` and `height` to the same value.

Parameters

- **`*args`** – Arguments to be passed on to superclasses.
- **`width`** (*`float`*) – The width of the tool. Defaults to `None`.
- **`**kwargs`** – Keyword arguments to be passed on to superclasses.

Raises `ValueError` – If `width` is not set.

`set_size`(*width*)
Set the size of the tool.

Parameters **`width`** (*`float`*) – The width of the tool.

CHAPTER 8

Tool Types

class mr_sim.types.**Orbital**(**args*, *eccentricity=None*, ***kwargs*)
A class for simulating a random orbit sander.

This class includes the necessary methods for calculating the total velocity of a random orbit sander.

eccentricity
The eccentric distance of the sander.

Type float

orbital_speed
The current rotational speed of the eccentric link.

Type float

rotational_speed
The current rotational speed of the pad with with respect to the part surface.

Type float

Parameters

- ***args** – Arguments to be passed on to superclasses.
- **eccentricity** (*float*) – The eccentric distance of the sander. Defaults to None.
- ****kwargs** – Keyword arguments to be passed on to superclasses.

Raises `ValueError` – If `eccentricity` is not set.

set_speed(*orbital_speed=0*, *rotational_speed=0*)
Sets the orbital and rotational speed.

Parameters

- **orbital_speed** (*float*) – The current rotational speed of the eccentric link. Defaults to 0.

- **rotational_speed** (*float*) – The current rotational speed of the pad with with respect to the part surface. Defaults to 0.

velocity (*x*, *y*)
Determines the velocity of the tool

Determines the total velocity of the tool with respect to the part sufrace.

Note: This is an approximation that should be more rigerously verified.

Parameters

- **x** (*numpy.ndarray*) – A 2D array of the X coordinates of the part centered at the current tool location.
- **y** (*numpy.ndarray*) – A 2D array of the Y coordinates of the part centered at the current tool location.

Returns A 2D array of velocity on the part surface.

Return type numpy.ndarray

class mr_sim.types.**Belt** (**args*, ***kwargs*)
A class for simulating a belt sander.

This class includes the necessary methods for calculating the total velocity of a belt sander.

speed
The current speed of the belt.

Type float

Parameters

- ***args** – Arguments to be passed on to superclasses.
- ****kwargs** – Keyword arguments to be passed on to superclasses.

set_speed (*speed*)
Sets the speed of the belt.

Parameters **speed** (*float*) – The current speed of the belt.

velocity (*x*, *y*)
Determines the velocity of the tool

Determines the total velocity of the tool with respect to the part sufrace.

Note: This is an approximation that should be more rigerously verified.

Parameters

- **x** (*numpy.ndarray*) – A 2D array of the X coordinates of the part centered at the current tool location.
- **y** (*numpy.ndarray*) – A 2D array of the Y coordinates of the part centered at the current tool location.

Returns A 2D array of velocity on the part surface.

Return type numpy.ndarray

class `mr_sim.types.`**`Rotary`**(**args*, ***kwargs*)

A class for simulating a rotary abrasive tool.

This class includes the necessary methods for calculating the total velocity of a rotary tool.

`speed`

The current rotational speed of the tool.

Type float

Parameters

- **`*args`** – Arguments to be passed on to superclasses.
- **`**kwargs`** – Keyword arguments to be passed on to superclasses.

`set_speed`(*speed*)

Sets the rotational speed of the tool.

Parameters **`speed`** (*`float`*) – The current rotational speed of the tool.

`velocity`(*x*, *y*)

Determines the velocity of the tool

Determines the total velocity of the tool with respect to the part sufrace.

Note: This is an approximation that should be more rigerously verified.

Parameters

- **`x`** (*`numpy.ndarray`*) – A 2D array of the X coordinates of the part centered at the current tool location.
- **`y`** (*`numpy.ndarray`*) – A 2D array of the Y coordinates of the part centered at the current tool location.

Returns A 2D array of velocity on the part surface.

Return type numpy.ndarray

Python Module Index

Index