pysces Documentation

Release 0.1

Clancy Rowley

December 03, 2015

Contents

This package implements a boundary element method for solving for the fluid flow around streamlined bodies moving in a potential flow. The goal is to provide a clean interface so that different methods may be easily swapped out, while being computationally efficient.

Testing and coverage

Automated tests are provided, and may be run with:

```
$ python runtests.py
```

You can also generate a coverage report as follows:

```
$ coverage run runtests.py
$ coverage report -m
```

To generate a nice html report, use:

```
$ coverage html
```

and then open the files generated in the directory *htmlcov/*.

CHAPTER 1

Documentation

1.1 Function reference

The *pysces* package implements a boundary element method for inviscid fluid flows.

1.1.1 Generating bodies

Body(points)	Base class for representing bodies
TransformedBody(body[, angle, displacement])	Base class for rigid (Euclidean) transformations of existing bodies
Pitching(body, amplitude, frequency[, phase])	Sinusoidal pitching for an existing body
Heaving(body, displacement, frequency[, phase])	Sinusoidal heaving for an existing body
cylinder(radius, num_points)	Return a circular Body with the given radius and number of points
flat_plate(num_points)	Return a flat plate Body with the given number of points.
joukowski_foil([xcenter, ycenter, a, numpoints])	Return a Joukowski foil Body.
karman_trefftz_foil([xcenter, ycenter, a, ...])	Return a Karman-Trefftz foil Body.
naca_airfoil(code, num_points[, ...])	Return a NACA 4-digit series airfoil
van_de_vooren_foil([semichord, thickness, ...])	Return a van de Vooren foil Body.

pysces.Body

class pysces.**Body**(*points*)

Base class for representing bodies

__init__(*points*)

Create a body with nodes at the given points

Parameters **points** : 2d array, shape (n,2)

Array of points defining the boundary of the body For a closed body, the boundary curve should be positively oriented (counter-clockwise around outside of body), starting from trailing edge

Methods

__init__(points)	Create a body with nodes at the given points
get_body()	Return the Body object in the body-fixed frame

Continued on next page

Table 1.2 – continued from previous page

get_motion()	Return the transformation from the body-fixed to inertial frame
get_points([body_frame])	

Attributes

time	The time used to specify the body's motion

get_body()
Return the Body object in the body-fixed frame

get_motion()
Return the transformation from the body-fixed to inertial frame

time
The time used to specify the body's motion

pysces.TransformedBody

class pysces.**TransformedBody**(*body*, *angle=0*, *displacement=(0, 0)*)
Base class for rigid (Euclidean) transformations of existing bodies

__init__(*body*, *angle=0*, *displacement=(0, 0)*)
angles are clockwise, in degrees

Methods

__init__(body[, angle, displacement])	angles are clockwise, in degrees
get_body()	
get_motion()	
get_points([body_frame])	
set_motion(value)	

Attributes

time

pysces.Pitching

class pysces.**Pitching**(*body*, *amplitude*, *frequency*, *phase=0.0*)
Sinusoidal pitching for an existing body

__init__(*body*, *amplitude*, *frequency*, *phase=0.0*)
amplitude and phase given in degrees

Methods

`__init__`(body, amplitude, frequency[, phase])	amplitude and phase given in degrees
`get_body`()	
`get_motion`()	
`get_points`([body_frame])	
`set_motion`(value)	

Attributes

`time`

pysces.Heaving

class `pysces.`**`Heaving`**(*body*, *displacement*, *frequency*, *phase=0.0*)

Sinusoidal heaving for an existing body

`__init__`(*body*, *displacement*, *frequency*, *phase=0.0*)

Methods

`__init__`(body, displacement, frequency[, phase])
`get_body`()
`get_motion`()
`get_points`([body_frame])
`set_motion`(value)

Attributes

`time`

pysces.cylinder

`pysces.`**`cylinder`**(*radius*, *num_points*)

Return a circular Body with the given radius and number of points

pysces.flat_plate

`pysces.`**`flat_plate`**(*num_points*)

Return a flat plate Body with the given number of points.

In body coordinates the plate runs from (0,0) to (1,0).

pysces.joukowski_foil

`pysces.`**`joukowski_foil`**(*xcenter=-0.1*, *ycenter=0.1*, *a=1*, *numpoints=32*)

Return a Joukowski foil Body.

The foil has its trailing edge at (2a,0). The foil has a total of numpoints along the boundary. Refer to chapter 4 of *[R1]* for details.

Parameters **xcenter, ycenter** : float

(xcenter,ycenter) is the center of the Joukowski preimage circle. xcenter should be negative and small; its magnitude determines the bluffness of the foil. ycenter should be small; it determines the magnitude of the camber (positive gives upward camber, and negative gives downward camber).

a : float

radius of the Joukowski preimage circle

numpoints : int

number of points along the boundary

References

[R1]

pysces.karman_trefftz_foil

pysces.**karman_trefftz_foil**(*xcenter=-0.1*, *ycenter=0*, *a=0.1*, *angle_deg=10*, *numpoints=32*)

Return a Karman-Trefftz foil Body.

The Karman-Trefftz foil is a modified version of the Joukowski foil but with a nonzero interior angle — rather than a cusp — at the trailing edge. Refer to *[R2]* for details.

Parameters **xcenter, ycenter, a** : float.

The same as in joukowski_foil().

angle_deg : float

The interior angle, in degrees, at the trailing edge.

numpoints : int

Number of points along the boundary

See also:

joukowski_foil

References

[R2]

pysces.naca_airfoil

pysces.**naca_airfoil**(*code*, *num_points*, *zero_thick_te=False*, *uniform=False*)

Return a NACA 4-digit series airfoil

pysces.van_de_vooren_foil

pysces.**van_de_vooren_foil**(*semichord=1.0*, *thickness=0.15*, *angle_deg=5*, *numpoints=32*)

Return a van de Vooren foil Body.

Refer to section 6.6 of *[R3]*

Parameters **semichord** : float

half the chord c, so c=2*semichord

thickness : float

vertical thickness as a fraction (0 < thickness < 1) of the semichord

angle_deg : float

interior angle, in degrees, at the trailing edge

numpoints : int

number of points along the boundary

References

[R3]

1.1.2 Panel methods

Vortices([positions, strengths])	
BoundVortices(body[, Uinfty])	A class for bound vortex panels
BoundSourceDoublets(body)	

pysces.Vortices

class pysces.**Vortices**(*positions=None*, *strengths=None*)

__init__(*positions=None*, *strengths=None*)

Methods

__init__([positions, strengths])	
append(position, strength)	
induced_velocity([x, motion])	Compute the induced velocity at the given point(s)
induced_velocity_single(x, xvort, gam)	Compute velocity induced at points x by a single vortex

Attributes

circulation
core_radius

Continued on next page

Table 1.12 – continued from previous page

positions
strengths

induced_velocity(*x=None*, *motion=None*)
Compute the induced velocity at the given point(s)

induced_velocity_single(*x*, *xvort*, *gam*)
Compute velocity induced at points x by a single vortex

This method returns a vector of velocities at the points x induced by a single vortex of strength gam located at xvort.

Parameters **x** : 2d array

Locations at which to compute induced velocity. Expressed as column vectors (i.e., shape should be (n,2))

xvort : 1d array

Location of vortex (shape should be (2,))

gam : float

Strength of vortex

Notes

Induced velocity is

$$u_\theta = \frac{\Gamma}{2\pi r}$$

where r is the distance between the point and the vortex. If this distance is less than `core_radius` r_0, the velocity is regularized as solid-body rotation, with

$$u_\theta = \frac{\Gamma r}{2\pi r_0^2}$$

pysces.BoundVortices

class pysces.**BoundVortices**(*body*, *Uinfty=(1, 0)*)
A class for bound vortex panels

__init__(*body*, *Uinfty=(1, 0)*)

Methods

__init__(body[, Uinfty])	
compute_rhs([Uinfty, wake])	
get_newly_shed()	Return newly shed wake vortex in the inertial frame
induced_velocity(x)	
update_positions()	
update_strengths([Uinfty])	Update vortex strengths
update_strengths_unsteady(dt[, Uinfty, ...])	Update strengths for unsteady calculation

Attributes

collocation_pts
influence_matrix
normals
num_panels
tangents
time
vortices

get_newly_shed()
Return newly shed wake vortex in the inertial frame

Returns **x_shed** : 1d array, shape (2,)

Location of newly shed wake vortex, in inertial frame

gam_shed : float

Strength of newly shed vortex

update_strengths(*Uinfty=(1, 0)*)
Update vortex strengths

update_strengths_unsteady(*dt*, *Uinfty=(1, 0)*, *wake=None*, *circ=None*, *wake_fac=0.25*)
Update strengths for unsteady calculation

Shed a new wake panel (not added into wake)

Parameters **dt** : float

Timestep

Uinfty : array_like, optional

Farfield fluid velocity (default (1,0))

wake : wake panel object, optional

Induces velocities on the body (default None)

circ : float, optional

Total bound circulation, for enforcing Kelvin's circulation theorem. If None (default), obtain the total circulation from the wake, assuming overall circulation (body + wake) is zero

wake_fac : float, optional

New wake vortex is placed a distance wake_fac * Uinfty * dt from trailing edge (see Katz & Plotkin, p390).

pysces.BoundSourceDoublets

class pysces.**BoundSourceDoublets**(*body*)

__init__(*body*)

Methods

`__init__`(body)	
`get_wake_panel`()	
`update_positions`()	
`update_strengths`(wake, Uinfty, dt)	

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Bibliography

[R1] Acheson, D. J., “Elementary Fluid Dynamics”, Oxford, 1990.

[R2] https://en.wikipedia.org/wiki/Joukowsky_transform

[R3] Katz, Joseph and Plotkin, Allen, “Low-Speed Aerodynamics”, 2nd Ed., Cambridge University Press, 2001.

p

p

Symbols

B

C

F

G

H

I

J

K

N

P

T

U

V