

---

# PyStokes

*Release 2.1.1*

**Jun 30, 2020**



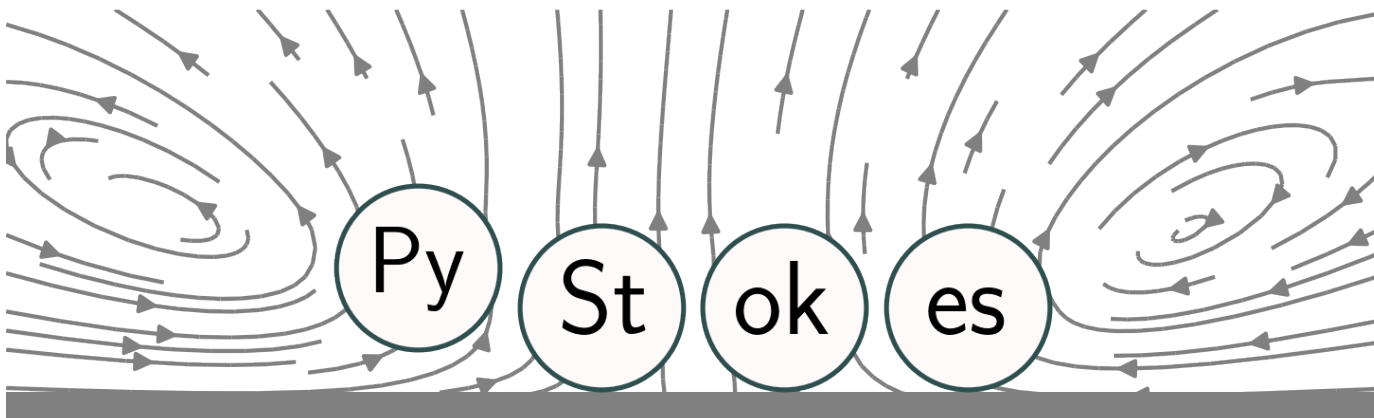
---

## Contents

---

<b>1</b>	<b>API Reference</b>	<b>3</b>
	<b>Index</b>	<b>31</b>





**PyStokes** is a numerical library for Phoresis and Stokesian hydrodynamics in Python. It uses a grid-free method, combining the integral representation of Stokes and Laplace equations, spectral expansion, and Galerkin discretization, to compute phoretic and hydrodynamic interactions between spheres with slip boundary conditions on their surfaces. The library also computes suspension scale quantities, such as rheological response, energy dissipation and fluid flow. The computational cost is quadratic in the number of particles and upto  $1e5$  particles have been accommodated on multicore computers. The library has been used to model suspensions of **microorganisms**, **synthetic autophoretic particles** and **self-propelling droplets**.

Please see installation instructions and more details in the [README.md](#) on GitHub.



### 1.1 Stokes flow in an unbounded domain

Hydrodynamic interactions of active particles in an unbounded domain.

#### 1.1.1 RBM (Rigid Body Motion: Velocity and Angular Velocity)

**class** `pystokes.unbounded.Rbm`

Rigid body motion (RBM) - velocity and angular velocity

Methods in this class update velocities or angular velocities using the inputs of - arrays of positions, velocity or angular velocity, along with an array of forces or torques or a slip mode

The array of velocity or angular velocities is then update by each method.

...

**radius: float** Radius of the particles (a).

**particles: int** Number of particles (Np)

**viscosity: float** Viscosity of the fluid (eta)

**mobilityRR()**

Compute angular velocity due to body torques using  $\mathbf{o} = \mu^{RR} \cdot \mathbf{T} \dots$

**Parameters**

- $\mathbf{o}$  (`np.array`) - An array of angular velocities An array of size  $3 \cdot N_p$ ,
- $\mathbf{r}$  (`np.array`) - An array of positions An array of size  $3 \cdot N_p$ ,
- $\mathbf{T}$  (`np.array`) - An array of forces An array of size  $3 \cdot N_p$ ,

**mobilityRT()**

Compute angular velocity due to body forces using  $\mathbf{o} = \mu^{RT} \cdot \mathbf{F} \dots$

**Parameters**

- $\mathbf{o}$  (*np.array*) – An array of angular velocities An array of size  $3*Np$ ,
- $\mathbf{r}$  (*np.array*) – An array of positions An array of size  $3*Np$ ,
- $\mathbf{F}$  (*np.array*) – An array of forces An array of size  $3*Np$ ,

**mobilityTR()**

Compute velocity due to body torque using  $v = \mu^{TR} \cdot T \dots$

**Parameters**

- $\mathbf{v}$  (*np.array*) – An array of velocities An array of size  $3*Np$ ,
- $\mathbf{r}$  (*np.array*) – An array of positions An array of size  $3*Np$ ,
- $\mathbf{T}$  (*np.array*) – An array of torques An array of size  $3*Np$ ,

**mobilityTT()**

Compute velocity due to body forces using  $v = \mu^{TT} \cdot F \dots$

**Parameters**

- $\mathbf{v}$  (*np.array*) – An array of velocities An array of size  $3*Np$ ,
- $\mathbf{r}$  (*np.array*) – An array of positions An array of size  $3*Np$ ,
- $\mathbf{F}$  (*np.array*) – An array of forces An array of size  $3*Np$ ,

## Examples

An example of the RBM

```
>>> import pystokes, numpy as np, matplotlib.pyplot as plt
>>> # particle radius, self-propulsion speed, number and fluid viscosity
>>> b, vs, Np, eta = 1.0, 1.0, 128, 0.1

>>> #initialise
>>> r = pystokes.utils.initialCondition(Np) # initial random distribution of
↪positions
>>> p = np.zeros(3*Np); p[2*Np:3*Np] = -1 # initial orientation of the
↪colloids
>>>
>>> rbm = pystokes.unbounded.Rbm(radius=b, particles=Np, viscosity=eta)
>>> force = pystokes.forceFields.Forces(particles=Np)
>>>
>>> def rhs(rp):
>>>     # assign fresh values at each time step
>>>     r = rp[0:3*Np]; p = rp[3*Np:6*Np]
>>>     F, v, o = np.zeros(3*Np), np.zeros(3*Np), np.zeros(3*Np)
>>>
>>>     force.lennardJonesWall(F, r, lje=0.01, ljr=5, wlje=1.2, wljr=3.4)
>>>     rbm.mobilityTT(v, r, F)
>>>     return np.concatenate( (v,o) )
>>>
>>> # simulate the resulting system
>>> Tf, Npts = 150, 200
>>> pystokes.utils.simulate(np.concatenate((r,p)),
>>>     Tf,Npts,rhs,integrator='odeint', filename='crystallization')
```

**noiseRR()**

Compute rotational Brownian motion ...



**Parameters**

- $\mathbf{o}$  (*np.array*) – An array of angular velocities An array of size  $3*N_p$ ,
- $\mathbf{r}$  (*np.array*) – An array of positions An array of size  $3*N_p$ ,

**noiseTT()**

Compute translation Brownian motion ...

**Parameters**

- $\mathbf{v}$  (*np.array*) – An array of velocities An array of size  $3*N_p$ ,
- $\mathbf{r}$  (*np.array*) – An array of positions An array of size  $3*N_p$ ,

**propulsionR2s()**

Compute angular velocity due to 2s mode of the slip  $v = \pi^{R,2s} \cdot S \dots$

**Parameters**

- $\mathbf{o}$  (*np.array*) – An array of angular velocities An array of size  $3*N_p$ ,
- $\mathbf{r}$  (*np.array*) – An array of positions An array of size  $3*N_p$ ,
- $\mathbf{S}$  (*np.array*) – An array of 2s mode of the slip An array of size  $5*N_p$ ,

**propulsionR3a()**

Compute angular velocity due to 3a mode of the slip  $v = \pi^{R,3a} \cdot V \dots$

**Parameters**

- $\mathbf{o}$  (*np.array*) – An array of angular velocities An array of size  $3*N_p$ ,
- $\mathbf{r}$  (*np.array*) – An array of positions An array of size  $3*N_p$ ,
- $\mathbf{V}$  (*np.array*) – An array of 3a mode of the slip An array of size  $5*N_p$ ,

**propulsionR3s()**

Compute angular velocity due to 3s mode of the slip  $v = \pi^{R,3s} \cdot G \dots$

**Parameters**

- $\mathbf{o}$  (*np.array*) – An array of angular velocities An array of size  $3*N_p$ ,
- $\mathbf{r}$  (*np.array*) – An array of positions An array of size  $3*N_p$ ,
- $\mathbf{G}$  (*np.array*) – An array of 3s mode of the slip An array of size  $7*N_p$ ,

**propulsionR4a()**

Compute angular velocity due to 4a mode of the slip  $v = \pi^{R,4a} \cdot M \dots$

**Parameters**

- $\mathbf{o}$  (*np.array*) – An array of angular velocities An array of size  $3*N_p$ ,
- $\mathbf{r}$  (*np.array*) – An array of positions An array of size  $3*N_p$ ,
- $\mathbf{M}$  (*np.array*) – An array of 4a mode of the slip An array of size  $7*N_p$ ,

**propulsionT2s()**

Compute velocity due to 2s mode of the slip  $v = \pi^{T,2s} \cdot S \dots$

**Parameters**

- $\mathbf{v}$  (*np.array*) – An array of velocities An array of size  $3*N_p$ ,
- $\mathbf{r}$  (*np.array*) – An array of positions An array of size  $3*N_p$ ,
- $\mathbf{S}$  (*np.array*) – An array of 2s mode of the slip An array of size  $5*N_p$ ,

**propulsionT3a()**

Compute velocity due to 3a mode of the slip  $v = \pi^{T,3a} \cdot V \dots$

**Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **V** (*np.array*) – An array of 3a mode of the slip An array of size 5\*Np,

**propulsionT3s()**

Compute velocity due to 3s mode of the slip  $v = \pi^{T,3s} \cdot G \dots$

**Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **G** (*np.array*) – An array of 3s mode of the slip An array of size 7\*Np,

**propulsionT3t()**

Compute velocity due to 3t mode of the slip  $v = \pi^{T,3t} \cdot D \dots$

**Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **D** (*np.array*) – An array of 3t mode of the slip An array of size 3\*Np,

**propulsionT4a()**

Compute velocity due to 4a mode of the slip  $v = \pi^{T,4a} \cdot M \dots$

**Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **M** (*np.array*) – An array of 4a mode of the slip An array of size 7\*Np,

## 1.1.2 Flow

**class** `pystokes.unbounded.Flow`

Flow at given points

...

**Parameters**

- **radius** (*float*) – Radius of the particles.
- **particles** (*int*) – Number of particles
- **viscosity** (*viscosity of the fluid*) –
- **gridpoints** (*int*) – Number of grid points

## Examples

An example of the RBM

**flowField1s()**

Compute flow field at field points due body forces ...

### Parameters

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **F** (*np.array*) – An array of body force An array of size 3\*Np,

## Examples

An example of the Flow field due to 1s mode of force per unit area

```
>>> import pystokes, numpy as np, matplotlib.pyplot as plt
>>>
>>> # particle radius, self-propulsion speed, number and fluid viscosity
>>> b, eta, Np = 1.0, 1.0/6.0, 1
>>>
>>> # initialize
>>> r, p = np.array([0.0, 0.0, 3.4]), np.array([0.0, 1.0, 0])
>>> Fls = pystokes.utils.irreducibleTensors(1, p)
>>>
>>> # space dimension , extent , discretization
>>> dim, L, Ng = 3, 10, 64;
>>>
>>> # instantiate the Flow class
>>> flow = pystokes.unbounded.Flow(radius=b, particles=Np, viscosity=eta,
↳ gridpoints=Ng*Ng)
>>>
>>> # create grid, evaluate flow and plot
>>> rr, vv = pystokes.utils.gridXY(dim, L, Ng)
>>> flow.flowField1s(vv, rr, r, Fls)
>>> pystokes.utils.plotStreamlinesXY(vv, rr, r, offset=6-1, density=1.4,
↳ title='1s')
```

**flowField2a()**

Compute flow field at field points due body Torque ...

### Parameters

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **T** (*np.array*) – An array of body torque An array of size 3\*Np,

## Examples

An example of the RBM

```
# Example 1: Flow field due to 2a mode of force per unit area >>> import pystokes, numpy as
np, matplotlib.pyplot as plt >>> >>> # particle radius, self-propulsion speed, number and fluid vis-
cosity >>> b, eta, Np = 1.0, 1.0/6.0, 1 >>> >>> # initialize >>> r, p = np.array([0.0, 0.0, 3.4]),
np.array([0.0, 1.0, 0]) >>> V2a = pystokes.utils.irreducibleTensors(1, p) >>> >>> # space dimension ,
extent , discretization >>> dim, L, Ng = 3, 10, 64; >>> >>> # instantiate the Flow class >>> flow =
pystokes.wallBounded.Flow(radius=b, particles=Np, viscosity=eta, gridpoints=Ng*Ng) >>> >>> # create
grid, evaluate flow and plot >>> rr, vv = pystokes.utils.gridXY(dim, L, Ng) >>> flow.flowField2a(vv, rr, r,
V2a) >>> pystokes.utils.plotStreamlinesXY(vv, rr, r, offset=6-1, density=1.4, title='2s')
```

### flowField2s()

Compute flow field at field points due to 2s mode of the slip ...

#### Parameters

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **S** (*np.array*) – An array of 2s mode of the slip An array of size 5\*Np,

### Examples

An example of the Flow field due to 3t mode of active slip

```
>>> import pystokes, numpy as np, matplotlib.pyplot as plt
>>>
>>> # particle radius, self-propulsion speed, number and fluid viscosity
>>> b, eta, Np = 1.0, 1.0/6.0, 1
>>>
>>> # initialize
>>> r, p = np.array([0.0, 0.0, 3.4]), np.array([0.0, 1.0, 0])
>>> V3t = pystokes.utils.irreducibleTensors(1, p)
>>>
>>> # space dimension , extent , discretization
>>> dim, L, Ng = 3, 10, 64;
>>>
>>> # instantiate the Flow class
>>> flow = pystokes.wallBounded.Flow(radius=b, particles=Np, viscosity=eta,
↳ gridpoints=Ng*Ng)
>>>
>>> # create grid, evaluate flow and plot
>>> rr, vv = pystokes.utils.gridXY(dim, L, Ng)
>>> flow.flowField3t(vv, rr, r, V3t)
>>> pystokes.utils.plotStreamlinesXY(vv, rr, r, offset=6-1, density=1.4,
↳ title='1s')
```

### flowField3a()

Compute flow field at field points due to 3a mode of the slip ...

#### Parameters

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **V** (*np.array*) – An array of 3a mode of the slip An array of size 5\*Np,

**flowField3s()**

Compute flow field at field points due to 3s mode of the slip ...

**Parameters**

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **G** (*np.array*) – An array of 3s mode of the slip An array of size 7\*Np,

**flowField3t()**

Compute flow field at field points due to 3t mode of the slip ...

**Parameters**

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **D** (*np.array*) – An array of 3t mode of the slip An array of size 3\*Np,

**Examples**

An example of the Flow field due to 3t mode of active slip

```
>>> import pystokes, numpy as np, matplotlib.pyplot as plt
>>>
>>> # particle radius, self-propulsion speed, number and fluid viscosity
>>> b, eta, Np = 1.0, 1.0/6.0, 1
>>>
>>> # initialize
>>> r, p = np.array([0.0, 0.0, 3.4]), np.array([0.0, 1.0, 0])
>>> V3t = pystokes.utils.irreducibleTensors(1, p)
>>>
>>> # space dimension , extent , discretization
>>> dim, L, Ng = 3, 10, 64;
>>>
>>> # instantiate the Flow class
>>> flow = pystokes.wallBounded.Flow(radius=b, particles=Np, viscosity=eta,
↳ gridpoints=Ng*Ng)
>>>
>>> # create grid, evaluate flow and plot
>>> rr, vv = pystokes.utils.gridXY(dim, L, Ng)
>>> flow.flowField3t(vv, rr, r, V3t)
>>> pystokes.utils.plotStreamlinesXY(vv, rr, r, offset=6-1, density=1.4,
↳ title='2s')
```

**flowField4a()**

Compute flow field at field points due to 4a mode of the slip ...

**Parameters**

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,

- **M** (*np.array*) – An array of 4a mode of the slip An array of size 7\*Np,

## 1.2 Stokes flow near interfaces

Hydrodynamic interactions of active particles in the half-space bounded by a plane no-shear surface

### 1.2.1 RBM (Rigid Body Motion: Velocity and Angular Velocity)

```
class pystokes.interface.Rbm
```

```
    Rigid body motion (RBM)
```

```
    ...
```

#### Parameters

- **radius** (*float*) – Radius of the particles (a)
- **particles** (*int*) – Number of particles (Np)
- **viscosity** (*float*) – Viscosity of the fluid (eta)

```
mobilityTR()
```

Compute velocity due to body forces using  $v = \mu^{TR} \cdot T \dots$

#### Parameters

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **T** (*np.array*) – An array of forces An array of size 3\*Np,
- **ll** (*float*) – viscosity ratio of the two fluids Default is zero

```
mobilityTT()
```

Compute velocity due to body forces using  $v = \mu^{TT} \cdot F \dots$

#### Parameters

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **F** (*np.array*) – An array of forces An array of size 3\*Np,
- **ll** (*float*) – viscosity ratio of the two fluids Default is zero

### Examples

An example of the RBM

```
>>> import pystokes, numpy as np, matplotlib.pyplot as plt
>>> # particle radius, self-propulsion speed, number and fluid viscosity
>>> b, vs, Np, eta = 1.0, 1.0, 128, 0.1
>>> #initialise
>>> r = pystokes.utils.initialCondition(Np) # initial random distribution of
↪positions
>>> p = np.zeros(3*Np); p[2*Np:3*Np] = -1 # initial orientation of the
↪colloids
```

(continues on next page)

(continued from previous page)

```

>>>
>>> rbm = pystokes.interface.Rbm(radius=b, particles=Np, viscosity=eta)
>>> force = pystokes.forceFields.Forces(particles=Np)
>>>
>>> def rhs(rp):
>>>     # assign fresh values at each time step
>>>     r = rp[0:3*Np]; p = rp[3*Np:6*Np]
>>>     F, v, o = np.zeros(3*Np), np.zeros(3*Np), np.zeros(3*Np)
>>>
>>>     force.lennardJonesWall(F, r, lje=0.01, ljr=5, wlje=1.2, wljr=3.4)
>>>     rbm.mobilityTT(v, r, F)
>>>     return np.concatenate( (v,o) )
>>>
>>> # simulate the resulting system
>>> Tf, Npts = 150, 200
>>> pystokes.utils.simulate(np.concatenate((r,p)),
>>>     Tf,Npts,rhs,integrator='odeint', filename='crystallization')

```

**noiseRR()**

Compute rotational Brownian motion ...

**Parameters**

- **o** (*np.array*) – An array of angular velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,

**noiseTT()**

Compute translation Brownian motion ...

**Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,

**propulsionT2s()**Compute velocity due to 2s mode of the slip  $v = \pi^{T,2s} \cdot S \dots$ **Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **S** (*np.array*) – An array of 2s mode of the slip An array of size 5\*Np,

**propulsionT3t()**Compute velocity due to 3t mode of the slip  $v = \pi^{T,3t} \cdot D \dots$ **Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **D** (*np.array*) – An array of 3t mode of the slip An array of size 3\*Np,

## 1.2.2 Flow

**class** `pystokes.interface.Flow`

Flow at given points

...

**Parameters**

- **radius** (*float*) – Radius of the particles.
- **particles** (*int*) – Number of particles
- **viscosity** (*viscosity of the fluid*) –
- **gridpoints** (*int*) – Number of grid points

**Examples**

An example of the RBM

**flowFieldIs()**

Compute flow field at field points due body forces ...

**Parameters**

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **F** (*np.array*) – An array of body force An array of size 3\*Np,

**Examples**

An example of the Flow field due to 1s mode of force per unit area

```
>>> import pystokes, numpy as np, matplotlib.pyplot as plt
>>>
>>> # particle radius, self-propulsion speed, number and fluid viscosity
>>> b, eta, Np = 1.0, 1.0/6.0, 1
>>>
>>> # initialize
>>> r, p = np.array([0.0, 0.0, 3.4]), np.array([0.0, 1.0, 0])
>>> Fls = pystokes.utils.irreducibleTensors(1, p)
>>>
>>> # space dimension , extent , discretization
>>> dim, L, Ng = 3, 10, 64;
>>>
>>> # instantiate the Flow class
>>> flow = pystokes.interface.Flow(radius=b, particles=Np, viscosity=eta,
↳ gridpoints=Ng*Ng)
>>>
>>> # create grid, evaluate flow and plot
>>> rr, vv = pystokes.utils.gridYZ(dim, L, Ng)
>>> flow.flowFieldIs(vv, rr, r, Fls)
>>> pystokes.utils.plotStreamlinesYZsurf(vv, rr, r, offset=6-1, density=1.4,
↳ title='1s')
```

## 1.3 Stokes flow bounded by a wall

Hydrodynamic interactions of active particles in the half-space bounded by a plane no-slip surface



### 1.3.1 RBM (Rigid Body Motion: Velocity and Angular Velocity)

**class** `pystokes.wallBounded.Rbm`

Rigid body motion (RBM)

...

#### Parameters

- **radius** (*float*) – Radius of the particles (*a*)
- **particles** (*int*) – Number of particles (*Np*)
- **viscosity** (*float*) – Viscosity of the fluid (*eta*)

**mobilityRR**()

Compute angular velocity due to body torques using  $\boldsymbol{o} = \boldsymbol{\mu}^{RR} \cdot \boldsymbol{T} \dots$

#### Parameters

- ***o*** (*np.array*) – An array of angular velocities An array of size  $3*Np$ ,
- ***r*** (*np.array*) – An array of positions An array of size  $3*Np$ ,
- ***T*** (*np.array*) – An array of forces An array of size  $3*Np$ ,

**mobilityRT**()

Compute angular velocity due to body forces using  $\boldsymbol{o} = \boldsymbol{\mu}^{RT} \cdot \boldsymbol{F} \dots$

#### Parameters

- ***o*** (*np.array*) – An array of angular velocities An array of size  $3*Np$ ,
- ***r*** (*np.array*) – An array of positions An array of size  $3*Np$ ,
- ***F*** (*np.array*) – An array of forces An array of size  $3*Np$ ,

**mobilityTR**()

Compute velocity due to body torque using  $\boldsymbol{v} = \boldsymbol{\mu}^{TR} \cdot \boldsymbol{T} \dots$

#### Parameters

- ***v*** (*np.array*) – An array of velocities An array of size  $3*Np$ ,
- ***r*** (*np.array*) – An array of positions An array of size  $3*Np$ ,
- ***T*** (*np.array*) – An array of torques An array of size  $3*Np$ ,

**mobilityTT**()

Compute velocity due to body forces using  $\boldsymbol{v} = \boldsymbol{\mu}^{TT} \cdot \boldsymbol{F} \dots$

#### Parameters

- ***v*** (*np.array*) – An array of velocities An array of size  $3*Np$ ,
- ***r*** (*np.array*) – An array of positions An array of size  $3*Np$ ,
- ***F*** (*np.array*) – An array of forces An array of size  $3*Np$ ,

### Examples

An example of the RBM

```

>>> import pystokes, numpy as np, matplotlib.pyplot as plt
>>> # particle radius, self-propulsion speed, number and fluid viscosity
>>> b, vs, Np, eta = 1.0, 1.0, 128, 0.1
>>> #initialise
>>> r = pystokes.utils.initialCondition(Np) # initial random distribution of
↪positions
>>> p = np.zeros(3*Np); p[2*Np:3*Np] = -1 # initial orientation of the
↪colloids
>>>
>>> rbm = pystokes.wallBounded.Rbm(radius=b, particles=Np, viscosity=eta)
>>> force = pystokes.forceFields.Forces(particles=Np)
>>>
>>> def rhs(rp):
>>>     # assign fresh values at each time step
>>>     r = rp[0:3*Np]; p = rp[3*Np:6*Np]
>>>     F, v, o = np.zeros(3*Np), np.zeros(3*Np), np.zeros(3*Np)
>>>
>>>     force.lennardJonesWall(F, r, lje=0.01, ljr=5, wlje=1.2, wljr=3.4)
>>>     rbm.mobilityTT(v, r, F)
>>>     return np.concatenate( (v,o) )
>>>
>>> # simulate the resulting system
>>> Tf, Npts = 150, 200
>>> pystokes.utils.simulate(np.concatenate((r,p)),
>>>     Tf,Npts,rhs,integrator='odeint', filename='crystallization')

```

**noiseRR()**

Compute rotational Brownian motion ...

**Parameters**

- **o** (*np.array*) – An array of angular velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,

**noiseTT()**

Compute translation Brownian motion ...

**Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,

**propulsionT2s()**

Compute velocity due to 2s mode of the slip  $v = \pi^{T,2s} \cdot S$  ...

**Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **S** (*np.array*) – An array of 2s mode of the slip An array of size 5\*Np,

**propulsionT3t()**

Compute velocity due to 3t mode of the slip  $v = \pi^{T,3t} \cdot D$  ...

**Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,

- **D** (*np.array*) – An array of 3t mode of the slip An array of size 3\*Np,

### 1.3.2 Flow

**class** pystokes.wallBounded.**Flow**

Flow at given points

...

#### Parameters

- **radius** (*float*) – Radius of the particles.
- **particles** (*int*) – Number of particles
- **viscosity** (*viscosity of the fluid*) –
- **gridpoints** (*int*) – Number of grid points

#### Examples

An example of the RBM

**flowField1s** ()

Compute flow field at field points due body forces ...

#### Parameters

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **F** (*np.array*) – An array of body force An array of size 3\*Np,

#### Examples

An example of the Flow field due to 1s mode of force per unit area

```
>>> import pystokes, numpy as np, matplotlib.pyplot as plt
>>>
>>> # particle radius, self-propulsion speed, number and fluid viscosity
>>> b, eta, Np = 1.0, 1.0/6.0, 1
>>>
>>> # initialize
>>> r, p = np.array([0.0, 0.0, 3.4]), np.array([0.0, 1.0, 0])
>>> Fls = pystokes.utils.irreducibleTensors(1, p)
>>>
>>> # space dimension , extent , discretization
>>> dim, L, Ng = 3, 10, 64;
>>>
>>> # instantiate the Flow class
>>> flow = pystokes.wallBounded.Flow(radius=b, particles=Np, viscosity=eta,
↳ gridpoints=Ng*Ng)
>>>
>>> # create grid, evaluate flow and plot
>>> rr, vv = pystokes.utils.gridYZ(dim, L, Ng)
```

(continues on next page)

(continued from previous page)

```
>>> flow.flowField1s(vv, rr, r, Fls)
>>> pystokes.utils.plotStreamlinesYZsurf(vv, rr, r, offset=6-1, density=1.4,
↳title='1s')
```

**flowField2a()**

Compute flow field at field points due body Torque ...

**Parameters**

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **T** (*np.array*) – An array of body torque An array of size 3\*Np,

**Examples**

An example of the RBM

```
# Example 1: Flow field due to $2a$ mode of force per unit area >>> import pystokes, numpy as
np, matplotlib.pyplot as plt >>> >>> # particle radius, self-propulsion speed, number and fluid vis-
cosity >>> b, eta, Np = 1.0, 1.0/6.0, 1 >>> >>> # initialize >>> r, p = np.array([0.0, 0.0, 3.4]),
np.array([0.0, 1.0, 0]) >>> V2a = pystokes.utils.irreducibleTensors(1, p) >>> >>> # space dimension ,
extent , discretization >>> dim, L, Ng = 3, 10, 64; >>> >>> # instantiate the Flow class >>> flow =
pystokes.wallBounded.Flow(radius=b, particles=Np, viscosity=eta, gridpoints=Ng*Ng) >>> >>> # create
grid, evaluate flow and plot >>> rr, vv = pystokes.utils.gridYZ(dim, L, Ng) >>> flow.flowField2a(vv, rr, r,
V2a) >>> pystokes.utils.plotStreamlinesYZsurf(vv, rr, r, offset=6-1, density=1.4, title='2s')
```

**flowField2s()**

Compute flow field at field points due to 2s mode of the slip ...

**Parameters**

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **S** (*np.array*) – An array of 2s mode of the slip An array of size 5\*Np,

**Examples**

An example of the Flow field due to \$3t\$ mode of active slip

```
>>> import pystokes, numpy as np, matplotlib.pyplot as plt
>>>
>>> # particle radius, self-propulsion speed, number and fluid viscosity
>>> b, eta, Np = 1.0, 1.0/6.0, 1
>>>
>>> # initialize
>>> r, p = np.array([0.0, 0.0, 3.4]), np.array([0.0, 1.0, 0])
>>> V3t = pystokes.utils.irreducibleTensors(1, p)
>>>
>>> # space dimension , extent , discretization
>>> dim, L, Ng = 3, 10, 64;
```

(continues on next page)

(continued from previous page)

```

>>>
>>> # instantiate the Flow class
>>> flow = pystokes.wallBounded.Flow(radius=b, particles=Np, viscosity=eta,
↳gridpoints=Ng*Ng)
>>>
>>> # create grid, evaluate flow and plot
>>> rr, vv = pystokes.utils.gridYZ(dim, L, Ng)
>>> flow.flowField3t(vv, rr, r, V3t)
>>> pystokes.utils.plotStreamlinesYZsurf(vv, rr, r, offset=6-1, density=1.4,
↳title='1s')

```

**flowField3t()**

Compute flow field at field points due to 3t mode of the slip ...

**Parameters**

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **D** (*np.array*) – An array of 3t mode of the slip An array of size 3\*Np,

**Examples**

An example of the Flow field due to 3t mode of active slip

```

>>> import pystokes, numpy as np, matplotlib.pyplot as plt
>>>
>>> # particle radius, self-propulsion speed, number and fluid viscosity
>>> b, eta, Np = 1.0, 1.0/6.0, 1
>>>
>>> # initialize
>>> r, p = np.array([0.0, 0.0, 3.4]), np.array([0.0, 1.0, 0])
>>> V3t = pystokes.utils.irreducibleTensors(1, p)
>>>
>>> # space dimension , extent , discretization
>>> dim, L, Ng = 3, 10, 64;
>>>
>>> # instantiate the Flow class
>>> flow = pystokes.wallBounded.Flow(radius=b, particles=Np, viscosity=eta,
↳gridpoints=Ng*Ng)
>>>
>>> # create grid, evaluate flow and plot
>>> rr, vv = pystokes.utils.gridYZ(dim, L, Ng)
>>> flow.flowField3t(vv, rr, r, V3t)
>>> pystokes.utils.plotStreamlinesYZsurf(vv, rr, r, offset=6-1, density=1.4,
↳title='1s')

```

## 1.4 Stokes flow in a Hele-Shaw cell

Stokesian hydrodynamic in a Hele-Shaw cell

### 1.4.1 RBM (Rigid Body Motion: Velocity and Angular Velocity)

```
class pystokes.twoWalls.Rbm
```

```
    Rigid body motion (RBM)
```

```
    ...
```

```
    radius: float Radius of the particles (a).
```

```
    particles: int Number of particles (Np)
```

```
    viscosity: float Viscosity of the fluid (eta)
```

```
    mobilityTT()
```

```
        Compute velocity due to body forces using  $v = \mu^{TT} \cdot F \dots$ 
```

**Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **F** (*np.array*) – An array of forces An array of size 3\*Np,
- **H** (*float*) – Height of the Hele-Shaw cell

```
    propulsionT2s()
```

```
        Compute velocity due to 2s mode of the slip ...
```

**Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **S** (*np.array*) – An array of forces An array of size 5\*Np,
- **H** (*float*) – Height of the Hele-Shaw cell

```
    propulsionT3t()
```

```
        Compute velocity due to 3t mode of the slip ...
```

**Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **D** (*np.array*) – An array of forces An array of size 3\*Np,
- **H** (*float*) – Height of the Hele-Shaw cell

### 1.4.2 Flow

```
class pystokes.twoWalls.Flow
```

```
    Flow at given points
```

```
    ...
```

**Parameters**

- **radius** (*float*) – Radius of the particles.
- **particles** (*int*) – Number of particles
- **viscosity** (*viscosity of the fluid*) –

- **gridpoints** (*int*) – Number of grid points

**flowField1s** ()

Compute flow field at field points due body forces ...

**Parameters**

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **F** (*np.array*) – An array of body force An array of size 3\*Np,
- **H** (*float*) – Height of the Hele-Shaw cell

**flowField2s** ()

Compute flow field at field points due to 2s mode of th slip ...

**Parameters**

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **S** (*np.array*) – An array of 2s mode of the slip An array of size 5\*Np,
- **H** (*float*) – Height of the Hele-Shaw cell

**flowField3t** ()

Compute flow field at field points due to 3t mode of th slip ...

**Parameters**

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **D** (*np.array*) – An array of 2s mode of the slip An array of size r\*Np,
- **H** (*float*) – Height of the Hele-Shaw cell

## 1.5 Stokes flow in a periodic domain

Hydrodynamic interactions of active particles in a periodic 3D space

### 1.5.1 RBM (Rigid Body Motion: Velocity and Angular Velocity)

**class** `pystokes.periodic.Rbm`

Rigid body motion (RBM) - velocity and angular velocity

Methods in this class update velocities or angular velocities using the inputs of - arrays of positions, velocity or angular velocity, along with an array of forces or torques or a slip mode

The array of velocity or angular velocities is then update by each method.

...

**radius:** *float* Radius of the particles (a).

**particles:** `int` Number of particles (Np)

**viscosity:** `float` Viscosity of the fluid (eta)

**boxSize:** `float` Length of the box which is repeated periodically in 3D

**mobilityRR()**

Compute angular velocity due to body torques using  $\omega = \mu^{RR} \cdot T \dots$

**Parameters**

- $\omega$  (`np.array`) – An array of angular velocities An array of size 3\*Np,
- $\mathbf{r}$  (`np.array`) – An array of positions An array of size 3\*Np,
- $\mathbf{T}$  (`np.array`) – An array of forces An array of size 3\*Np,
- **Nb** (`int`) – Number of periodic boxed summed Default is 6
- **Nm** (`int`) – Number of Fourier modes summed Default is 6

**mobilityRT()**

Compute angular velocity due to body forces using  $\omega = \mu^{RT} \cdot F \dots$

**Parameters**

- $\omega$  (`np.array`) – An array of angular velocities An array of size 3\*Np,
- $\mathbf{r}$  (`np.array`) – An array of positions An array of size 3\*Np,
- $\mathbf{F}$  (`np.array`) – An array of forces An array of size 3\*Np,
- **Nb** (`int`) – Number of periodic boxed summed Default is 6
- **Nm** (`int`) – Number of Fourier modes summed Default is 6

**mobilityTR()**

Compute velocity due to body torque using  $v = \mu^{TR} \cdot T \dots$

**Parameters**

- $\mathbf{v}$  (`np.array`) – An array of velocities An array of size 3\*Np,
- $\mathbf{r}$  (`np.array`) – An array of positions An array of size 3\*Np,
- $\mathbf{T}$  (`np.array`) – An array of torques An array of size 3\*Np,
- **Nb** (`int`) – Number of periodic boxed summed Default is 6
- **Nm** (`int`) – Number of Fourier modes summed Default is 6

**mobilityTT()**

Compute velocity due to body forces using  $v = \mu^{TT} \cdot F \dots$

**Parameters**

- $\mathbf{v}$  (`np.array`) – An array of velocities An array of size 3\*Np,
- $\mathbf{r}$  (`np.array`) – An array of positions An array of size 3\*Np,
- $\mathbf{F}$  (`np.array`) – An array of forces An array of size 3\*Np,
- **Nb** (`int`) – Number of periodic boxed summed Default is 10
- **Nm** (`int`) – Number of Fourier modes summed Default is 1

**propulsionR2s()**

Compute angular velocity due to 2s mode of the slip  $\omega = \pi^{R,2s} \cdot S \dots$

**Parameters**



- $\mathbf{o}$  (*np.array*) – An array of angular velocities An array of size  $3*Np$ ,
- $\mathbf{r}$  (*np.array*) – An array of positions An array of size  $3*Np$ ,
- $\mathbf{S}$  (*np.array*) – An array of 2s mode of the slip An array of size  $5*Np$ ,
- $\mathbf{Nb}$  (*int*) – Number of periodic boxed summed Default is 6
- $\mathbf{Nm}$  (*int*) – Number of Fourier modes summed Default is 6

**propulsionR3a()**

Compute angular velocity due to 3a mode of the slip  $v = \pi^{R,3a} \cdot V \dots$

**Parameters**

- $\mathbf{o}$  (*np.array*) – An array of angular velocities An array of size  $3*Np$ ,
- $\mathbf{r}$  (*np.array*) – An array of positions An array of size  $3*Np$ ,
- $\mathbf{V}$  (*np.array*) – An array of 3a mode of the slip An array of size  $5*Np$ ,
- $\mathbf{Nb}$  (*int*) – Number of periodic boxed summed Default is 6
- $\mathbf{Nm}$  (*int*) – Number of Fourier modes summed Default is 6

**propulsionR3s()**

Compute angular velocity due to 3s mode of the slip  $v = \pi^{R,3s} \cdot G \dots$

**Parameters**

- $\mathbf{o}$  (*np.array*) – An array of angular velocities An array of size  $3*Np$ ,
- $\mathbf{r}$  (*np.array*) – An array of positions An array of size  $3*Np$ ,
- $\mathbf{G}$  (*np.array*) – An array of 3s mode of the slip An array of size  $7*Np$ ,
- $\mathbf{Nb}$  (*int*) – Number of periodic boxed summed Default is 6
- $\mathbf{Nm}$  (*int*) – Number of Fourier modes summed Default is 6

**propulsionR4a()**

Compute angular velocity due to 4a mode of the slip  $v = \pi^{R,4a} \cdot M \dots$

**Parameters**

- $\mathbf{o}$  (*np.array*) – An array of angular velocities An array of size  $3*Np$ ,
- $\mathbf{r}$  (*np.array*) – An array of positions An array of size  $3*Np$ ,
- $\mathbf{M}$  (*np.array*) – An array of 4a mode of the slip An array of size  $7*Np$ ,
- $\mathbf{Nb}$  (*int*) – Number of periodic boxed summed Default is 6
- $\mathbf{Nm}$  (*int*) – Number of Fourier modes summed Default is 6

**propulsionT2s()**

Compute velocity due to 2s mode of the slip  $v = \pi^{T,2s} \cdot S \dots$

**Parameters**

- $\mathbf{v}$  (*np.array*) – An array of velocities An array of size  $3*Np$ ,
- $\mathbf{r}$  (*np.array*) – An array of positions An array of size  $3*Np$ ,
- $\mathbf{S}$  (*np.array*) – An array of 2s mode of the slip An array of size  $5*Np$ ,
- $\mathbf{Nb}$  (*int*) – Number of periodic boxed summed Default is 6
- $\mathbf{Nm}$  (*int*) – Number of Fourier modes summed Default is 6

**propulsionT3a()**

Compute velocity due to 3a mode of the slip  $v = \pi^{T,3a} \cdot V \dots$

**Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **V** (*np.array*) – An array of 3a mode of the slip An array of size 5\*Np,
- **Nb** (*int*) – Number of periodic boxed summed Default is 6
- **Nm** (*int*) – Number of Fourier modes summed Default is 6

**propulsionT3s()**

Compute velocity due to 3s mode of the slip  $v = \pi^{T,3s} \cdot G \dots$

**Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **G** (*np.array*) – An array of 3s mode of the slip An array of size 7\*Np,
- **Nb** (*int*) – Number of periodic boxed summed Default is 6
- **Nm** (*int*) – Number of Fourier modes summed Default is 6

**propulsionT3t()**

Compute velocity due to 3t mode of the slip  $v = \pi^{T,3t} \cdot D \dots$

**Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **D** (*np.array*) – An array of 3t mode of the slip An array of size 3\*Np,
- **Nb** (*int*) – Number of periodic boxed summed Default is 6
- **Nm** (*int*) – Number of Fourier modes summed Default is 6

**propulsionT4a()**

Compute velocity due to 4a mode of the slip  $v = \pi^{T,4a} \cdot M \dots$

**Parameters**

- **v** (*np.array*) – An array of velocities An array of size 3\*Np,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **M** (*np.array*) – An array of 4a mode of the slip An array of size 7\*Np,
- **Nb** (*int*) – Number of periodic boxed summed Default is 6
- **Nm** (*int*) – Number of Fourier modes summed Default is 6

## 1.5.2 Flow

**class** `pystokes.periodic.Flow`

Flow at given points

...

**Parameters**

- **radius** (*float*) – Radius of the particles.
- **particles** (*int*) – Number of particles
- **viscosity** (*viscosity of the fluid*) –
- **gridpoints** (*int*) – Number of grid points
- **boxsize** (*int*) – Box size

**flowField1s()**

Compute flow field at field points due body forces ...

**Parameters**

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **F** (*np.array*) – An array of body force An array of size 3\*Np,
- **Nb** (*int*) – Number of periodic boxed summed Default is 6
- **Nm** (*int*) – Number of Fourier modes summed Default is 6

**Examples**

An example of the Flow field due to 1s mode of force per unit area

```
>>> import pystokes, numpy as np, matplotlib.pyplot as plt
>>>
>>> # particle radius, self-propulsion speed, number and fluid viscosity
>>> b, eta, Np = 1.0, 1.0/6.0, 1
>>>
>>> # initialize
>>> r, p = np.array([0.0, 0.0, 3.4]), np.array([0.0, 1.0, 0])
>>> Fls = pystokes.utils.irreducibleTensors(1, p)
>>>
>>> # space dimension , extent , discretization
>>> dim, L, Ng = 3, 10, 64;
>>>
>>> # instantiate the Flow class
>>> flow = pystokes.periodic.Flow(radius=b, particles=Np, viscosity=eta,
↳ gridpoints=Ng*Ng,
>>>     boxSize=L)
>>>
>>> # create grid, evaluate flow and plot
>>> rr, vv = pystokes.utils.gridXY(dim, L, Ng)
>>> flow.flowField1s(vv, rr, r, Fls)
>>> pystokes.utils.plotStreamlinesXY(vv, rr, r, offset=6-1, density=1.4,
↳ title='1s')
```

**flowField2s()**

Compute flow field at field points due to 2s mode of the slip ...

**Parameters**

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,

- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **S** (*np.array*) – An array of 2s mode of the slip An array of size 5\*Np,
- **Nb** (*int*) – Number of periodic boxed summed Default is 6
- **Nm** (*int*) – Number of Fourier modes summed Default is 6

**flowField3t** ()

Compute flow field at field points due to 3t mode of the slip ...

**Parameters**

- **vv** (*np.array*) – An array of flow at field points An array of size 3\*Nt,
- **rt** (*np.array*) – An array of field points An array of size 3\*Nt,
- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **D** (*np.array*) – An array of 3t mode of the slip An array of size 3\*Np,
- **Nb** (*int*) – Number of periodic boxed summed Default is 6
- **Nm** (*int*) – Number of Fourier modes summed Default is 6

## 1.6 Phoresis in an unbounded domain

Phoretic motion in an infinite space

### 1.6.1 Phoresis

**class** `pystokes.phoreticUnbounded.Phoresis`

...

**Parameters**

- **radius** (*float*) – Radius of the particles.
- **particles** (*int*) – Number of particles
- **phoreticConstant** (*float*) – PhoreticConstant

#### Examples

An example of Phoresis

### 1.6.2 Field

**class** `pystokes.phoreticUnbounded.Field`

Phoretic field at given points

...

**Parameters**

- **radius** (*float*) – Radius of the particles.
- **particles** (*int*) – Number of particles
- **phoreticConstant** (*float*) – PhoreticConstant of the fluid

- **gridpoints** (*int*) – Number of grid points

## 1.7 Phoresis in a space bounded by a wall

Phoresis in the half-space bounded by a plane no-slip wall

### 1.7.1 Phoresis

```
class pystokes.phoreticWallBounded.Phoresis
    ...
```

#### Parameters

- **radius** (*float*) – Radius of the particles.
- **particles** (*int*) – Number of particles
- **phoreticConstant** (*float*) – PhoreticConstant

#### Examples

An example of Phoresis

### 1.7.2 Field

```
class pystokes.phoreticWallBounded.Field
    Phoretic field at given points
    ...
```

#### Parameters

- **radius** (*float*) – Radius of the particles.
- **particles** (*int*) – Number of particles
- **phoreticConstant** (*float*) – PhoreticConstant of the fluid
- **gridpoints** (*int*) – Number of grid points

## 1.8 Force fields in colloidal systems

Force fields in colloidal systems

### 1.8.1 Forces

```
class pystokes.forceFields.Forces
    Computes forces in a system of colloidal particles
```

Methods in the Forces class take input,

- arrays of positions, forces
- parameters for a given potential

The array of forces is then update by each method.

...

**Parameters** `particles` (*int*) – Number of particles (Np)

**harmonicConfinement** ()

Forces on colloids in a harmonic trap

**lennardJones** ()

The standard Lennard-Jones potential truncated at the minimum (aslo called WCA potential)

We choose  $\phi(r) = lje/12 (r^{12} - 2*r^6) + lje/12$ , as the standard WCA potential. `ljr`: minimum of the LJ potential and  $r=ljr/r$ .

...

**Parameters**

- `r` (*np.array*) – An array of positions An array of size 3\*Np,
- `F` (*np.array*) – An array of forces An array of size 3\*Np,
- `lje` (*float*) – Strength of the LJ
- `ljr` (*float*) – Range of the LJ

**lennardJonesWall** ()

The standard Lennard-Jones potential truncated at the minimum (aslo called WCA potential)

We choose  $\phi(r) = lje/12 (r^{12} - 2*r^6) + lje/12$ , as the standard WCA potential. `ljr`: minimum of the LJ potential and  $r=ljr/r$ .

...

**Parameters**

- `r` (*np.array*) – An array of positions An array of size 3\*Np,
- `F` (*np.array*) – An array of forces An array of size 3\*Np,
- `lje` (*float*) – Strength of the LJ
- `ljr` (*float*) – Range of the LJ

**lennardJonesXWall** ()

The standard Lennard-Jones potential truncated at the minimum (aslo called WCA potential)

We choose  $\phi(r) = lje/12 (r^{12} - 2*r^6) + lje/12$ , as the standard WCA potential. `ljr`: minimum of the LJ potential and  $r=ljr/r$ . This force is only in z-direction due to wall at  $x=0$

...

**Parameters**

- `r` (*np.array*) – An array of positions An array of size 3\*Np,
- `F` (*np.array*) – An array of forces An array of size 3\*Np,
- `lje` (*float*) – Strength of the LJ
- `ljr` (*float*) – Range of the LJ

**membraneBound** ()

Force on colloids in membraneSurface

...

**Parameters**

- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **r0** (*np.array*) – An array of trap centers An array of size 3\*Np,
- **F** (*np.array*) – An array of forces An array of size 3\*Np,
- **cn** (*float*) – Stiffness of the trap

**membraneConfinement** ()

Force on colloids in membraneSurface

...

#### Parameters

- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **r0** (*np.array*) – An array of trap centers An array of size 3\*Np,
- **F** (*np.array*) – An array of forces An array of size 3\*Np,
- **cn** (*float*) – Stiffness of the trap

**membraneSurface** ()

Force on colloids connected as a membrane

...

#### Parameters

- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **F** (*np.array*) – An array of forces An array of size 3\*Np,
- **bondLength** (*float*) – The size of natural spring
- **springModulus** (*float*) – Stiffness of the trap
- **bendModulus** (*float*) – Bending cost

**multiRingpolymers** ()

Force on colloids connected by a spring in a ring polymer

...

#### Parameters

- **r** (*np.array*) – An array of positions
- **F** (*np.array*) – An array of forces An array of size 3\*Np, An array of size 3\*Np,
- **bondLength** (*float*) – The size of natural spring
- **springModulus** (*float*) – Stiffness of the trap

**multipolymers** ()

Force on colloids in many polymers connected by a spring

...

#### Parameters

- **r** (*np.array*) – An array of positions
- **F** (*np.array*) – An array of forces An array of size 3\*Np, An array of size 3\*Np,
- **bondLength** (*float*) – The size of natural spring
- **springModulus** (*float*) – Stiffness of the trap

**opticalConfinement()**

Force on colloids in optical traps of varying stiffnesses

...

**Parameters**

- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **r0** (*np.array*) – An array of trap centers An array of size 3\*Np,
- **F** (*np.array*) – An array of forces An array of size 3\*Np,
- **k** (*float*) – Stiffness of the trap

**sedimentation()**

Force on colloids in sedimentation

...

**Parameters**

- **r** (*np.array*) – An array of positions An array of size 3\*Np,
- **F** (*np.array*) – An array of forces An array of size 3\*Np,
- **g** (*float*) – Gravity

**softSpringWall()** $F = -k(r-r_{min})$ **spring()**

Force on colloids connected by a spring in a single polymer

...

**Parameters**

- **r** (*np.array*) – An array of positions
- **F** (*np.array*) – An array of forces An array of size 3\*Np, An array of size 3\*Np,
- **bondLength** (*float*) – The size of natural spring
- **springModulus** (*float*) – Stiffness of the trap

## 1.8.2 Torques

**class** `pystokes.forceFields.Torques`

Computes torques in a system of colloidal particles

**Methods in the Torques class take input,**

- arrays of positions, Torques
- parameters for a given potential

The array of torques is then update by each method.

...

**Parameters** **particles** (*int*) – Number of particles (Np)**bottomHeaviness()**

Torque due to bottom-heaviness

...



**Parameters**

- **p** (*np.array*) – An array of Orientations An array of size 3\*Np,
- **F** (*np.array*) – An array of Torques An array of size 3\*Np,
- **bh** (*float*) – bottomHeaviness

## 1.9 Utils: Miscellaneous function

`utils.irreducibleTensors()`

Uniaxial parameterization of the tensorial harmonics (Yl) of order l ...

**Parameters**

- **l** (*int*) – Tensorial Harmonics of order l
- **p** (*np.ndarray*) – An array of size 3 Axis along which the mode is parameterized
- **Y0** (*float*) – Strength of the mode
- **returns** (*Yl - tensorialHarmonics of rank l*) –

`utils.gridXY()`

Returns the grid in XY direction centered around zero ...

**Parameters**

- **L** (*float*) – Length of the grid
- **Ng** (*int*) – Number of grid points

### Examples

An example of creating grid

```
>>> import numpy as np, pystokes
>>> dim, L, Ng = 3, 10, 32
>>> rr, vv = pystokes.utils.gridXY(dim, L, Ng)
```

`utils.gridYZ()`

Returns the grid in YZ direction centered around zero ...

**Parameters**

- **L** (*float*) – Length of the grid
- **Ng** (*int*) – Number of grid points

### Examples

An example of creating grid

```
>>> import numpy as np, pystokes
>>> dim, L, Ng = 3, 10, 32
>>> rr, vv = pystokes.utils.gridYZ(dim, L, Ng)
```

`utils.simulate()`

Simulates using choice of integrator

...

**Parameters**

- **rp0** (*np.array*) – Initial condition
- **Tf** (*int*) – Final time
- **Npts** (*int*) – Number of points to return data
- **rhs** (*Python Function*) – Right hand side to integrate
- **integrator** (*string*) – Default is ‘odeint’ of scipy
- **filename** (*string*) – filename to write the data. Deafult is ‘this.mat’

## B

`bottomHeaviness()` (*pystokes.forceFields.Torques method*), 28

## F

`Field` (*class in pystokes.phoreticUnbounded*), 24  
`Field` (*class in pystokes.phoreticWallBounded*), 25  
`Flow` (*class in pystokes.interface*), 11  
`Flow` (*class in pystokes.periodic*), 22  
`Flow` (*class in pystokes.twoWalls*), 18  
`Flow` (*class in pystokes.unbounded*), 6  
`Flow` (*class in pystokes.wallBounded*), 15  
`flowField1s()` (*pystokes.interface.Flow method*), 12  
`flowField1s()` (*pystokes.periodic.Flow method*), 23  
`flowField1s()` (*pystokes.twoWalls.Flow method*), 19  
`flowField1s()` (*pystokes.unbounded.Flow method*), 7  
`flowField1s()` (*pystokes.wallBounded.Flow method*), 15  
`flowField2a()` (*pystokes.unbounded.Flow method*), 7  
`flowField2a()` (*pystokes.wallBounded.Flow method*), 16  
`flowField2s()` (*pystokes.periodic.Flow method*), 23  
`flowField2s()` (*pystokes.twoWalls.Flow method*), 19  
`flowField2s()` (*pystokes.unbounded.Flow method*), 8  
`flowField2s()` (*pystokes.wallBounded.Flow method*), 16  
`flowField3a()` (*pystokes.unbounded.Flow method*), 8  
`flowField3s()` (*pystokes.unbounded.Flow method*), 8  
`flowField3t()` (*pystokes.periodic.Flow method*), 24  
`flowField3t()` (*pystokes.twoWalls.Flow method*), 19  
`flowField3t()` (*pystokes.unbounded.Flow method*), 9  
`flowField3t()` (*pystokes.wallBounded.Flow method*), 17

`flowField4a()` (*pystokes.unbounded.Flow method*), 9

`Forces` (*class in pystokes.forceFields*), 25

## G

`gridXY()` (*pystokes.utils method*), 29  
`gridYZ()` (*pystokes.utils method*), 29

## H

`harmonicConfinement()` (*pystokes.forceFields.Forces method*), 26

## I

`irreducibleTensors()` (*pystokes.utils method*), 29

## L

`lennardJones()` (*pystokes.forceFields.Forces method*), 26  
`lennardJonesWall()` (*pystokes.forceFields.Forces method*), 26  
`lennardJonesXWall()` (*pystokes.forceFields.Forces method*), 26

## M

`membraneBound()` (*pystokes.forceFields.Forces method*), 26  
`membraneConfinement()` (*pystokes.forceFields.Forces method*), 27  
`membraneSurface()` (*pystokes.forceFields.Forces method*), 27  
`mobilityRR()` (*pystokes.periodic.Rbm method*), 20  
`mobilityRR()` (*pystokes.unbounded.Rbm method*), 3  
`mobilityRR()` (*pystokes.wallBounded.Rbm method*), 13  
`mobilityRT()` (*pystokes.periodic.Rbm method*), 20  
`mobilityRT()` (*pystokes.unbounded.Rbm method*), 3  
`mobilityRT()` (*pystokes.wallBounded.Rbm method*), 13  
`mobilityTR()` (*pystokes.interface.Rbm method*), 10

`mobilityTR()` (*pystokes.periodic.Rbm method*), 20  
`mobilityTR()` (*pystokes.unbounded.Rbm method*), 4  
`mobilityTR()` (*pystokes.wallBounded.Rbm method*), 13  
`mobilityTT()` (*pystokes.interface.Rbm method*), 10  
`mobilityTT()` (*pystokes.periodic.Rbm method*), 20  
`mobilityTT()` (*pystokes.twoWalls.Rbm method*), 18  
`mobilityTT()` (*pystokes.unbounded.Rbm method*), 4  
`mobilityTT()` (*pystokes.wallBounded.Rbm method*), 13  
`multipolymers()` (*pystokes.forceFields.Forces method*), 27  
`multiRingpolymers()` (*pystokes.forceFields.Forces method*), 27

## N

`noiseRR()` (*pystokes.interface.Rbm method*), 11  
`noiseRR()` (*pystokes.unbounded.Rbm method*), 4  
`noiseRR()` (*pystokes.wallBounded.Rbm method*), 14  
`noiseTT()` (*pystokes.interface.Rbm method*), 11  
`noiseTT()` (*pystokes.unbounded.Rbm method*), 5  
`noiseTT()` (*pystokes.wallBounded.Rbm method*), 14

## O

`opticalConfinement()` (*pystokes.forceFields.Forces method*), 27

## P

`Phoresis` (*class in pystokes.phoreticUnbounded*), 24  
`Phoresis` (*class in pystokes.phoreticWallBounded*), 25  
`propulsionR2s()` (*pystokes.periodic.Rbm method*), 20  
`propulsionR2s()` (*pystokes.unbounded.Rbm method*), 5  
`propulsionR3a()` (*pystokes.periodic.Rbm method*), 21  
`propulsionR3a()` (*pystokes.unbounded.Rbm method*), 5  
`propulsionR3s()` (*pystokes.periodic.Rbm method*), 21  
`propulsionR3s()` (*pystokes.unbounded.Rbm method*), 5  
`propulsionR4a()` (*pystokes.periodic.Rbm method*), 21  
`propulsionR4a()` (*pystokes.unbounded.Rbm method*), 5  
`propulsionT2s()` (*pystokes.interface.Rbm method*), 11  
`propulsionT2s()` (*pystokes.periodic.Rbm method*), 21  
`propulsionT2s()` (*pystokes.twoWalls.Rbm method*), 18  
`propulsionT2s()` (*pystokes.unbounded.Rbm method*), 5

`propulsionT2s()` (*pystokes.wallBounded.Rbm method*), 14  
`propulsionT3a()` (*pystokes.periodic.Rbm method*), 21  
`propulsionT3a()` (*pystokes.unbounded.Rbm method*), 5  
`propulsionT3s()` (*pystokes.periodic.Rbm method*), 22  
`propulsionT3s()` (*pystokes.unbounded.Rbm method*), 6  
`propulsionT3t()` (*pystokes.interface.Rbm method*), 11  
`propulsionT3t()` (*pystokes.periodic.Rbm method*), 22  
`propulsionT3t()` (*pystokes.twoWalls.Rbm method*), 18  
`propulsionT3t()` (*pystokes.unbounded.Rbm method*), 6  
`propulsionT3t()` (*pystokes.wallBounded.Rbm method*), 14  
`propulsionT4a()` (*pystokes.periodic.Rbm method*), 22  
`propulsionT4a()` (*pystokes.unbounded.Rbm method*), 6

## R

`Rbm` (*class in pystokes.interface*), 10  
`Rbm` (*class in pystokes.periodic*), 19  
`Rbm` (*class in pystokes.twoWalls*), 18  
`Rbm` (*class in pystokes.unbounded*), 3  
`Rbm` (*class in pystokes.wallBounded*), 13

## S

`sedimentation()` (*pystokes.forceFields.Forces method*), 28  
`simulate()` (*pystokes.utils method*), 29  
`softSpringWall()` (*pystokes.forceFields.Forces method*), 28  
`spring()` (*pystokes.forceFields.Forces method*), 28

## T

`Torques` (*class in pystokes.forceFields*), 28