My Project

Generated by Doxygen 1.8.8

Sun Apr 17 2016 23:15:23

Contents

1 README 1

2 Class Index 3

2.1 Class List . . . 3

3 File Index 5

3.1 File List . . . 5

4 Class Documentation 7

4.1 EquationOfMotion Class Reference . . . 7

4.1.1 Constructor & Destructor Documentation . . . 7

4.1.1.1 EquationOfMotion . . . 7

4.1.2 Member Function Documentation . . . 7

4.1.2.1 EulerMethod . . . 7

4.1.2.2 RungeKutta4Method . . . 7

4.2 EquationOfState Class Reference . . . 8

4.2.1 Detailed Description . . . 8

4.2.2 Member Function Documentation . . . 8

4.2.2.1 calculateEnergyDensity . . . 8

4.2.2.2 calculateEntropyDensity . . . 8

4.2.2.3 calculatePressureDensity . . . 8

4.3 IdealEquationOfState Class Reference . . . 9

4.3.1 Detailed Description . . . 9

4.3.2 Member Function Documentation . . . 9

4.3.2.1 calculateEnergyDensity . . . 9

4.3.2.2 calculateEntropyDensity . . . 9

4.3.2.3 calculatePressureDensity . . . 10

4.4 KernelFunction Class Reference . . . 10

4.4.1 Member Function Documentation . . . 10

4.4.1.1 kernelFunction . . . 10

4.4.1.2 kernelGradient . . . 11

4.5 SPHEquation Class Reference . . . 11

4.5.1 Constructor & Destructor Documentation . . . 11
4.5.1.1 SPHEquation . . . 11
4.5.2 Member Function Documentation . . . 11
4.5.2.1 dMomentum_dTau . . . 11
4.5.2.2 entropyStar . . . 12
4.6 SPHParticle Class Reference . . . 12
4.6.1 Constructor & Destructor Documentation . . . 13
4.6.1.1 SPHParticle . . . 13
4.6.2 Member Function Documentation . . . 13
4.6.2.1 CalculateEntropyDensity . . . 13
4.6.2.2 Gamma . . . 13
4.6.2.3 hasDownParticle . . . 14
4.6.2.4 hasLeftParticle . . . 14
4.6.2.5 hasRightParticle . . . 14
4.6.2.6 hasUpParticle . . . 14
4.6.3 Member Data Documentation . . . 14
4.6.3.1 dMomentum_dTau . . . 14
4.6.3.2 downParticle . . . 14
4.6.3.3 energyDensity . . . 14
4.6.3.4 entropyDensity . . . 14
4.6.3.5 entropyDensityWeight . . . 14
4.6.3.6 equationOfMotion . . . 14
4.6.3.7 leftParticle . . . 14
4.6.3.8 momentum . . . 14
4.6.3.9 position . . . 14
4.6.3.10 pressureDensity . . . 14
4.6.3.11 rightParticle . . . 14
4.6.3.12 upParticle . . . 14
4.6.3.13 velocity . . . 14

5 File Documentation **15**

5.1 /Users/rafael/Dropbox/sphene/README.md File Reference . . . 15
5.2 /Users/rafael/Dropbox/sphene/src/equationOfMotion.cpp File Reference . . . 15
5.3 /Users/rafael/Dropbox/sphene/src/equationOfMotion.h File Reference . . . 15
5.4 /Users/rafael/Dropbox/sphene/src/equationOfState.cpp File Reference . . . 15
5.4.1 Variable Documentation . . . 15
5.4.1.1 _PI . . . 15
5.5 /Users/rafael/Dropbox/sphene/src/equationOfState.h File Reference . . . 16
5.6 /Users/rafael/Dropbox/sphene/src/idealEquationOfState.cpp File Reference . . . 16
5.6.1 Variable Documentation . . . 16

5.6.1.1 _PI . . . 16
5.7 /Users/rafael/Dropbox/sphene/src/idealEquationOfState.h File Reference . . . 16
5.8 /Users/rafael/Dropbox/sphene/src/kernelFunction.cpp File Reference . . . 16
5.8.1 Variable Documentation . . . 17
5.8.1.1 GRAD_W . . . 17
5.8.1.2 GRAD_W_2 . . . 17
5.8.1.3 H . . . 17
5.8.1.4 PI . . . 17
5.8.1.5 W . . . 17
5.8.1.6 W_2 . . . 17
5.9 /Users/rafael/Dropbox/sphene/src/kernelFunction.h File Reference . . . 17
5.10 /Users/rafael/Dropbox/sphene/src/main.cpp File Reference . . . 17
5.10.1 Function Documentation . . . 17
5.10.1.1 main . . . 17
5.11 /Users/rafael/Dropbox/sphene/src/sphEquation.cpp File Reference . . . 17
5.12 /Users/rafael/Dropbox/sphene/src/sphEquation.h File Reference . . . 18
5.13 /Users/rafael/Dropbox/sphene/src/sphParticle.cpp File Reference . . . 18
5.14 /Users/rafael/Dropbox/sphene/src/sphParticle.h File Reference . . . 18
5.15 /Users/rafael/Dropbox/sphene/src/vector.cpp File Reference . . . 18
5.15.1 Function Documentation . . . 18
5.15.1.1 normOf . . . 18
5.15.1.2 operator∗ . . . 19
5.15.1.3 operator∗ . . . 19
5.15.1.4 operator+ . . . 19
5.15.1.5 operator- . . . 19
5.16 /Users/rafael/Dropbox/sphene/src/vector.h File Reference . . . 20
5.16.1 Function Documentation . . . 20
5.16.1.1 normOf . . . 20
5.16.1.2 operator∗ . . . 20
5.16.1.3 operator∗ . . . 21
5.16.1.4 operator+ . . . 21
5.16.1.5 operator- . . . 21
5.17 /Users/rafael/Dropbox/sphene/test/unitTests.cpp File Reference . . . 21
5.17.1 Function Documentation . . . 22
5.17.1.1 main . . . 22
5.17.1.2 TEST . . . 22
5.17.1.3 TEST . . . 22
5.17.1.4 TEST . . . 23
5.17.1.5 TEST . . . 23
5.17.1.6 TEST . . . 23

5.17.1.7 TEST 23
5.17.1.8 TEST 24
5.17.1.9 velocity 25

Index **27**

Chapter 1

README

Author

Rafael

Teste

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

EquationOfMotion . . . . . ??

EquationOfState

Abstract class for equations of state . . . . . ??

IdealEquationOfState

The class of the “ideal" equation of state: $p = \frac{\epsilon}{3}$ . . . . . ??

KernelFunction . . . . . ??

SPHEquation . . . . . ??

SPHParticle . . . . . ??

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

/Users/rafael/Dropbox/sphene/src/equationOfMotion.cpp . . . ??
/Users/rafael/Dropbox/sphene/src/equationOfMotion.h . . . ??
/Users/rafael/Dropbox/sphene/src/equationOfState.cpp . . . ??
/Users/rafael/Dropbox/sphene/src/equationOfState.h . . . ??
/Users/rafael/Dropbox/sphene/src/idealEquationOfState.cpp . . . ??
/Users/rafael/Dropbox/sphene/src/idealEquationOfState.h . . . ??
/Users/rafael/Dropbox/sphene/src/kernelFunction.cpp . . . ??
/Users/rafael/Dropbox/sphene/src/kernelFunction.h . . . ??
/Users/rafael/Dropbox/sphene/src/main.cpp . . . ??
/Users/rafael/Dropbox/sphene/src/sphEquation.cpp . . . ??
/Users/rafael/Dropbox/sphene/src/sphEquation.h . . . ??
/Users/rafael/Dropbox/sphene/src/sphParticle.cpp . . . ??
/Users/rafael/Dropbox/sphene/src/sphParticle.h . . . ??
/Users/rafael/Dropbox/sphene/src/vector.cpp . . . ??
/Users/rafael/Dropbox/sphene/src/vector.h . . . ??
/Users/rafael/Dropbox/sphene/test/unitTests.cpp . . . ??

Chapter 4

Class Documentation

4.1 EquationOfMotion Class Reference

`#include <equationOfMotion.h>`

Collaboration diagram for EquationOfMotion:

Public Member Functions

- EquationOfMotion ()
- vector< double > EulerMethod (vector< double >, vector< double >(*)(double), double, double)
- vector< double > RungeKutta4Method (vector< double >, vector< double >(*)(double), double, double)

4.1.1 Constructor & Destructor Documentation

4.1.1.1 EquationOfMotion::EquationOfMotion ( )

```
6 {
7
8 }
```

4.1.2 Member Function Documentation

4.1.2.1 vector< double > EquationOfMotion::EulerMethod (vector< double > , vector< double > *)(double, double , double)

```
11 {
12   return y + dt * f(t);
13 }
```

4.1.2.2 vector<double> EquationOfMotion::RungeKutta4Method (vector< double > , vector< double > *)(double, double , double)

The documentation for this class was generated from the following files:

- /Users/rafael/Dropbox/sphene/src/equationOfMotion.h
- /Users/rafael/Dropbox/sphene/src/equationOfMotion.cpp

4.2 EquationOfState Class Reference

Abstract class for equations of state.

`#include <equationOfState.h>`

Collaboration diagram for EquationOfState:

Public Member Functions

- double calculatePressureDensity (double)

  Function that returns pressure density in function of entropy density : $p(s)$.
- double calculateEnergyDensity (double)

  Function that returns energy density in function of entropy density : $\epsilon(s)$.
- double calculateEntropyDensity (double)

  Function that returns entropy density in function of energy density : $s(\epsilon)$.

4.2.1 Detailed Description

Abstract class for equations of state.

4.2.2 Member Function Documentation

4.2.2.1 double EquationOfState::calculateEnergyDensity (double *EntropyDensity*)

Function that returns energy density in function of entropy density : $\epsilon(s)$.

Parameters

EntropyDensity	

Referenced by TEST().

```
9 {return 0}
```

4.2.2.2 double EquationOfState::calculateEntropyDensity (double *EnergyDensity*)

Function that returns entropy density in function of energy density : $s(\epsilon)$.

Parameters

EnergyDensity	

Referenced by TEST().

```
11 {return 0}
```

4.2.2.3 double EquationOfState::calculatePressureDensity (double *EntropyDensity*)

Function that returns pressure density in function of entropy density : $p(s)$.

Parameters

EntropyDensity	

Referenced by TEST().

```
7 {return 0}
```

The documentation for this class was generated from the following files:

- /Users/rafael/Dropbox/sphene/src/equationOfState.h
- /Users/rafael/Dropbox/sphene/src/equationOfState.cpp

4.3 IdealEquationOfState Class Reference

The class of the “ideal" equation of state: $p = \frac{\epsilon}{3}$.

`#include <idealEquationOfState.h>`

Collaboration diagram for IdealEquationOfState:

Public Member Functions

- double calculatePressureDensity (double)

  Function that returns pressure density in function of entropy density : $p(s)$.
- double calculateEnergyDensity (double)

  Function that returns energy density in function of entropy density : $\epsilon(s)$.
- double calculateEntropyDensity (double)

  Function that returns entropy density in function of energy density : $s(\epsilon)$.

4.3.1 Detailed Description

The class of the “ideal" equation of state: $p = \frac{\epsilon}{3}$.

4.3.2 Member Function Documentation

4.3.2.1 double IdealEquationOfState::calculateEnergyDensity (double *EntropyDensity*)

Function that returns energy density in function of entropy density : $\epsilon(s)$.

Parameters

EntropyDensity	

```
15 {
16   return pow(EntropyDensity/2.43, 1.333333333);/*_PI * _PI * 47.5 * pow((15.0 * EntropyDensity)/(_PI * _PI
      * 47.5), 1.3333333333) / 30.0;*/
17 }
```

4.3.2.2 double IdealEquationOfState::calculateEntropyDensity (double *EnergyDensity*)

Function that returns entropy density in function of energy density : $s(\epsilon)$.

Parameters

EnergyDensity	

```
20 {
21   return 2.43 * pow(EnergyDensity, 0.75);
22   /*_PI * _PI * 47.5 * pow((30.0 * EnergyDensity)/(_PI * _PI * 47.5), 0.75)/15.0;*/
23 }
```

4.3.2.3 double IdealEquationOfState::calculatePressureDensity (double *EntropyDensity*)

Function that returns pressure density in function of entropy density : $p(s)$.

Parameters

EntropyDensity	

```
9 {
10
11   return calculateEnergyDensity(EntropyDensity)/3.0;/*_PI * _PI * 47.5 * pow((15.0 *
       EntropyDensity)/(_PI * _PI * 47.5), 1.333333333) / 30.0;*/
12 }
```

The documentation for this class was generated from the following files:

- /Users/rafael/Dropbox/sphene/src/idealEquationOfState.h
- /Users/rafael/Dropbox/sphene/src/idealEquationOfState.cpp

4.4 KernelFunction Class Reference

`#include <kernelFunction.h>`

Collaboration diagram for KernelFunction:

Public Member Functions

- double kernelFunction (vector< double >)
- vector< double > kernelGradient (vector< double >)

4.4.1 Member Function Documentation

4.4.1.1 double KernelFunction::kernelFunction (vector< double > *r*)

References H, normOf(), and W.

Referenced by SPHEquation::entropyStar(), and TEST().

```
13                                                  {
14
15     double a;
16     double q;
17
18     a = normOf(r);
19     q = a/H;
20
21     if (q > 2)
22         return 0;
23     if (q >= 1)
24         return W * 0.25 * (pow(2 - q, 3));
25
26     return W * (1 - (1.5 * q * q) + (0.75 * q * q * q));
27 }
```

4.4.1.2 vector< double > KernelFunction::kernelGradient (vector< double > *r*)

References GRAD_W, H, and normOf().

Referenced by SPHEquation::dMomentum_dTau().

```
30                                                                  {
31
32     double a = normOf(r);
33     double q = a/H;
34     vector<double> zero (2, 0.0);
35     vector<double> oneOverR(2, 0.0);
36
37     for(int i = 0; i < 2; i++)
38       oneOverR[i] = 1/(r[i]);
39
40     if (q > 2)
41         return zero;
42     if (q > 1)
43         return ((GRAD_W / a) * (2 - q) * (2 - q)) * r;
44
45     return (GRAD_W_2 * (-3 + 9 * q/4.)) * r;
46 }
```

The documentation for this class was generated from the following files:

- /Users/rafael/Dropbox/sphene/src/kernelFunction.h
- /Users/rafael/Dropbox/sphene/src/kernelFunction.cpp

4.5 SPHEquation Class Reference

`#include <sphEquation.h>`

Collaboration diagram for SPHEquation:

Public Member Functions

- SPHEquation ()
- double entropyStar (SPHParticle, vector< SPHParticle >, KernelFunction)
- vector< double > dMomentum_dTau (SPHParticle, vector< SPHParticle >, KernelFunction, double)

4.5.1 Constructor & Destructor Documentation

4.5.1.1 SPHEquation::SPHEquation ( )

```
6 {
7
8 }
```

4.5.2 Member Function Documentation

4.5.2.1 vector< double > SPHEquation::dMomentum_dTau (SPHParticle *sphParticle,* vector< SPHParticle > *listOfParticles,* KernelFunction *w,* double *tau*)

References SPHParticle::entropyDensity, SPHParticle::entropyDensityWeight, SPHParticle::Gamma(), KernelFunction::kernelGradient(), SPHParticle::position, and SPHParticle::pressureDensity.

Referenced by TEST().

```
25 {
26   vector<double> result (2, 0.0);
27   vector<double> r (2, 0.0);
```

```
28
29   for(int i = 0; i < listOfParticles.size(); i++)
30   {
31     r = sphParticle.position - listOfParticles[i].position;
32     result[0] += (sphParticle.entropyDensityWeight * listOfParticles[i].
      entropyDensityWeight) * (sphParticle.pressureDensity/pow((sphParticle.Gamma() * sphParticle.
      entropyDensity), 2) + (listOfParticles[i].pressureDensity/pow((listOfParticles[i].Gamma() *
      listOfParticles[i].entropyDensity), 2))) * w.kernelGradient(r)[0];
33     result[1] += (sphParticle.entropyDensityWeight * listOfParticles[i].
      entropyDensityWeight) * (sphParticle.pressureDensity/pow((sphParticle.Gamma() * sphParticle.
      entropyDensity), 2) + (listOfParticles[i].pressureDensity/pow((listOfParticles[i].Gamma() *
      listOfParticles[i].entropyDensity), 2))) * w.kernelGradient(r)[1];
34   }
35
36   return -(1/tau) * result;
37 }
```

4.5.2.2 double SPHEquation::entropyStar (SPHParticle *sphParticle*, vector< SPHParticle > *listOfParticles*, KernelFunction *w*)

: list of SPHParticles, KernelFunction (x-x_j)

References KernelFunction::kernelFunction(), and SPHParticle::position.

Referenced by TEST().

```
11 {
12   double entropyStar = 0.0;
13   vector<double> r (2, 0.0);
14
15   for(int i = 0; i < listOfParticles.size(); i++)
16   {
17     r = sphParticle.position - listOfParticles[i].position;
18     entropyStar += listOfParticles[i].entropyDensityWeight * w.kernelFunction(r);
19   }
20
21   return entropyStar;
22 }
```

The documentation for this class was generated from the following files:

- /Users/rafael/Dropbox/sphene/src/sphEquation.h
- /Users/rafael/Dropbox/sphene/src/sphEquation.cpp

4.6 SPHParticle Class Reference

`#include <sphParticle.h>`

Collaboration diagram for SPHParticle:

Public Member Functions

- SPHParticle ()
- bool hasLeftParticle ()
- bool hasRightParticle ()
- bool hasUpParticle ()
- bool hasDownParticle ()
- double Gamma ()
- void CalculateEntropyDensity (double)

Public Attributes

- vector< double > velocity
- vector< double > position
- vector< double > momentum
- vector< double > dMomentum_dTau
- double energyDensity
- double entropyDensity
- double entropyDensityWeight
- double pressureDensity
- EquationOfMotion equationOfMotion
- SPHParticle * leftParticle
- SPHParticle * rightParticle
- SPHParticle * upParticle
- SPHParticle * downParticle

4.6.1 Constructor & Destructor Documentation

4.6.1.1 SPHParticle::SPHParticle ( )

References velocity().

```
6 {
7   velocity.resize(2);
8
9   velocity[0] = 0.0;
10   velocity[1] = 0.0;
11
12   position.resize(2);
13   momentum.resize(2);
14   dMomentum_dTau.resize(2);
15 }
```

4.6.2 Member Function Documentation

4.6.2.1 void SPHParticle::CalculateEntropyDensity (double *entropyStar*)

s = {s^{*}}{{-g} }

```
26 {
27   entropyDensity = entropyStar/Gamma();
28 }
```

4.6.2.2 double SPHParticle::Gamma ( )

References normOf(), and velocity().

Referenced by SPHEquation::dMomentum_dTau().

```
18 {
19   double vAux = normOf(velocity);
20   return (1/sqrt(1.0 - (vAux*vAux)));
21 }
```

4.6.2.3 bool SPHParticle::hasDownParticle ( )

4.6.2.4 bool SPHParticle::hasLeftParticle ( )

4.6.2.5 bool SPHParticle::hasRightParticle ( )

4.6.2.6 bool SPHParticle::hasUpParticle ( )

4.6.3 Member Data Documentation

4.6.3.1 vector<double> SPHParticle::dMomentum_dTau

4.6.3.2 SPHParticle* SPHParticle::downParticle

4.6.3.3 double SPHParticle::energyDensity

Thermodynamic variables

Referenced by TEST().

4.6.3.4 double SPHParticle::entropyDensity

Referenced by SPHEquation::dMomentum_dTau().

4.6.3.5 double SPHParticle::entropyDensityWeight

Referenced by SPHEquation::dMomentum_dTau().

4.6.3.6 EquationOfMotion SPHParticle::equationOfMotion

4.6.3.7 SPHParticle* SPHParticle::leftParticle

4.6.3.8 vector<double> SPHParticle::momentum

4.6.3.9 vector<double> SPHParticle::position

Referenced by SPHEquation::dMomentum_dTau(), SPHEquation::entropyStar(), and TEST().

4.6.3.10 double SPHParticle::pressureDensity

Referenced by SPHEquation::dMomentum_dTau().

4.6.3.11 SPHParticle* SPHParticle::rightParticle

4.6.3.12 SPHParticle* SPHParticle::upParticle

4.6.3.13 vector<double> SPHParticle::velocity

Dynamic variables

The documentation for this class was generated from the following files:

- /Users/rafael/Dropbox/sphene/src/sphParticle.h
- /Users/rafael/Dropbox/sphene/src/sphParticle.cpp

Chapter 5

File Documentation

5.1 /Users/rafael/Dropbox/sphene/README.md File Reference

5.2 /Users/rafael/Dropbox/sphene/src/equationOfMotion.cpp File Reference

```
#include "equationOfMotion.h"
```
Include dependency graph for equationOfMotion.cpp:

5.3 /Users/rafael/Dropbox/sphene/src/equationOfMotion.h File Reference

```
#include <vector>
#include "vector.h"
```
Include dependency graph for equationOfMotion.h: This graph shows which files directly or indirectly include this file:

Classes

- class EquationOfMotion

5.4 /Users/rafael/Dropbox/sphene/src/equationOfState.cpp File Reference

```
#include "equationOfState.h"
```
Include dependency graph for equationOfState.cpp:

Variables

- double _PI = 3.14159265358979

5.4.1 Variable Documentation

5.4.1.1 double _PI = 3.14159265358979

5.5 /Users/rafael/Dropbox/sphene/src/equationOfState.h File Reference

`#include <cmath>`
Include dependency graph for equationOfState.h: This graph shows which files directly or indirectly include this file:

Classes

- class EquationOfState

  Abstract class for equations of state.

5.6 /Users/rafael/Dropbox/sphene/src/idealEquationOfState.cpp File Reference

```
#include "equationOfState.h"
#include "idealEquationOfState.h"
```
Include dependency graph for idealEquationOfState.cpp:

Variables

- double _PI = 3.14159265358979

5.6.1 Variable Documentation

5.6.1.1 double _PI = 3.14159265358979

5.7 /Users/rafael/Dropbox/sphene/src/idealEquationOfState.h File Reference

`#include <cmath>`
Include dependency graph for idealEquationOfState.h: This graph shows which files directly or indirectly include this file:

Classes

- class IdealEquationOfState

  The class of the "ideal" equation of state: $p = \frac{\epsilon}{3}$.

5.8 /Users/rafael/Dropbox/sphene/src/kernelFunction.cpp File Reference

`#include "kernelFunction.h"`
Include dependency graph for kernelFunction.cpp:

Variables

- const double PI = 3.14159265358979323846264338 3
- const double H = 0.3
- const double W = 10./ (PI ∗ 7 ∗ H ∗ H)
- const double W_2 = -10./ (7 ∗ PI ∗ H ∗ H ∗ H)
- const double GRAD_W = -(3./4) ∗ 10./ (7 ∗ PI ∗ H ∗ H ∗ H)
- const double GRAD_W_2 = 10./ (7 ∗ PI ∗ H ∗ H ∗ H ∗ H)

5.8.1 Variable Documentation

5.8.1.1 const double GRAD_W = -(3./4) * 10./ (7 * PI * H * H * H)

Referenced by KernelFunction::kernelGradient().

5.8.1.2 const double GRAD_W_2 = 10./ (7 * PI * H * H * H * H)

5.8.1.3 const double H = 0.3

Referenced by KernelFunction::kernelFunction(), and KernelFunction::kernelGradient().

5.8.1.4 const double PI = 3.14159265358979323846264338 3

5.8.1.5 const double W = 10./ (PI * 7 * H * H)

Referenced by KernelFunction::kernelFunction().

5.8.1.6 const double W_2 = -10./ (7 * PI * H * H * H)

5.9 /Users/rafael/Dropbox/sphene/src/kernelFunction.h File Reference

```
#include <iostream>
#include <vector>
#include <cmath>
#include "vector.h"
```

Include dependency graph for kernelFunction.h: This graph shows which files directly or indirectly include this file:

Classes

- class KernelFunction

5.10 /Users/rafael/Dropbox/sphene/src/main.cpp File Reference

Functions

- int main ()

5.10.1 Function Documentation

5.10.1.1 int main ( )

```
2 {
3     // TODO: implementation
4 }
```

5.11 /Users/rafael/Dropbox/sphene/src/sphEquation.cpp File Reference

```
#include "sphEquation.h"
```

Include dependency graph for sphEquation.cpp:

5.12 /Users/rafael/Dropbox/sphene/src/sphEquation.h File Reference

```
#include <vector>
#include "vector.h"
#include "sphParticle.h"
#include "kernelFunction.h"
```

Include dependency graph for sphEquation.h: This graph shows which files directly or indirectly include this file:

Classes

- class SPHEquation

5.13 /Users/rafael/Dropbox/sphene/src/sphParticle.cpp File Reference

```
#include "sphParticle.h"
```

Include dependency graph for sphParticle.cpp:

5.14 /Users/rafael/Dropbox/sphene/src/sphParticle.h File Reference

```
#include <iostream>
#include <vector>
#include <cmath>
#include "vector.h"
#include "equationOfMotion.h"
```

Include dependency graph for sphParticle.h: This graph shows which files directly or indirectly include this file:

Classes

- class SPHParticle

5.15 /Users/rafael/Dropbox/sphene/src/vector.cpp File Reference

```
#include "vector.h"
```

Include dependency graph for vector.cpp:

Functions

- vector< double > operator- (const vector< double > &r1, const vector< double > &r2)
- vector< double > operator+ (const vector< double > &r1, const vector< double > &r2)
- vector< double > operator* (const double &K, const vector< double > &x)
- double operator* (const vector< double > &a, const vector< double > &b)
- double normOf (const vector< double > &r)

5.15.1 Function Documentation

5.15.1.1 double normOf (const vector< double > &)

Function to calculates the norm of a vector

Referenced by SPHParticle::Gamma(), KernelFunction::kernelFunction(), and KernelFunction::kernelGradient().

```
64 {
65   double size = r.size();
66   double sum = 0.0;
67
68   for(int i = 0; i < size; i++)
69     sum += r[i] * r[i];
70
71   return sqrt(sum);
72 }
```

5.15.1.2 vector<double> operator*(const double &, const vector< double > &)

Overloading the times (*) operator to define the multiplication of a escalar (double) and a vector.

```
40 {
41   vector<double> result(x.size());
42   for (int i = 0; i < x.size(); i++)
43     result[i] = K * x[i];
44
45   return result;
46 }
```

5.15.1.3 double operator*(const vector< double > &, const vector< double > &)

Overloading the times(*) operator to define the escalar product between two vectors.

```
49 {
50   double result = 0;
51
52   if (a.size() != b.size()){
53     cerr << "Vector innerProduct Error: Vectors with different dimensions!";
54     exit(-1);
55   }
56
57   for (int i = 0; i < a.size(); i++)
58     result += a[i] * b[i];
59
60   return result;
61 }
```

5.15.1.4 vector<double> operator+(const vector< double > &, const vector< double > &)

Overloading the plus (+) operator to define the summation of two vectors.

```
22 {
23   double size;
24   size = r1.size();
25
26   if(r1.size() != r2.size()) {
27     cerr << "Vector Sumation Error: Vectors with different dimensions!";
28     exit(-1);
29   }
30
31   vector<double> r3 (size,0);
32
33   for(int i = 0; i < int(r3.size()); i++)
34     r3[i] = r1[i] + r2[i];
35
36   return r3;
37 }
```

5.15.1.5 vector<double> operator-(const vector< double > &, const vector< double > &)

Overloading the minus (-) operator to define the subtraction of two vectors.

```
4 {
5   double size;
6   size = r1.size();
7
8   if(r1.size() != r2.size()) {
9     cerr << "Vector Subtraction Error: Vectors with different dimensions!";
10     exit(-1);
11   }
12
13   vector<double> r3 (size,0);
14
15   for(int i = 0; i < int(r3.size()); i++)
16     r3[i] = r1[i] - r2[i];
17
18   return r3;
19 }
```

5.16 /Users/rafael/Dropbox/sphene/src/vector.h File Reference

```
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <vector>
#include <cmath>
```

Include dependency graph for vector.h: This graph shows which files directly or indirectly include this file:

Functions

- vector< double > operator+ (const vector< double > &, const vector< double > &)
- vector< double > operator- (const vector< double > &, const vector< double > &)
- vector< double > operator* (const double &, const vector< double > &)
- double operator* (const vector< double > &, const vector< double > &)
- double normOf (const vector< double > &)

5.16.1 Function Documentation

5.16.1.1 double normOf (const vector< double > &)

Function to calculates the norm of a vector

Referenced by SPHParticle::Gamma(), KernelFunction::kernelFunction(), and KernelFunction::kernelGradient().

```
64 {
65   double size = r.size();
66   double sum = 0.0;
67
68   for(int i = 0; i < size; i++)
69     sum += r[i] * r[i];
70
71   return sqrt(sum);
72 }
```

5.16.1.2 vector<double> operator* (const double & , const vector< double > &)

Overloading the times (*) operator to define the multiplication of a escalar (double) and a vector.

```
40 {
41   vector<double> result(x.size());
42   for (int i = 0; i < x.size(); i++)
43     result[i] = K * x[i];
44
45   return result;
46 }
```

5.16.1.3 double operator∗ (const vector< double > & , const vector< double > &)

Overloading the times(∗) operator to define the escalar product between two vectors.

```
49 {
50   double result = 0;
51
52   if (a.size() != b.size()){
53     cerr << "Vector innerProduct Error: Vectors with different dimensions!";
54     exit(-1);
55   }
56
57   for (int i = 0; i < a.size(); i++)
58     result += a[i] * b[i];
59
60   return result;
61 }
```

5.16.1.4 vector<double> operator+ (const vector< double > & , const vector< double > &)

Overloading the plus (+) operator to define the summation of two vectors.

```
22 {
23   double size;
24   size = r1.size();
25
26   if(r1.size() != r2.size()) {
27     cerr << "Vector Sumation Error: Vectors with different dimensions!";
28     exit(-1);
29   }
30
31   vector<double> r3 (size,0);
32
33   for(int i = 0; i < int(r3.size()); i++)
34     r3[i] = r1[i] + r2[i];
35
36   return r3;
37 }
```

5.16.1.5 vector<double> operator- (const vector< double > & , const vector< double > &)

Overloading the minus (-) operator to define the subtraction of two vectors.

```
4 {
5   double size;
6   size = r1.size();
7
8   if(r1.size() != r2.size()) {
9     cerr << "Vector Subtraction Error: Vectors with different dimensions!";
10    exit(-1);
11  }
12
13  vector<double> r3 (size,0);
14
15  for(int i = 0; i < int(r3.size()); i++)
16    r3[i] = r1[i] - r2[i];
17
18  return r3;
19 }
```

5.17 /Users/rafael/Dropbox/sphene/test/unitTests.cpp File Reference

```
#include <gtest/gtest.h>
```

```
#include <vector>
#include <iostream>
#include <fstream>
#include "../src/vector.h"
#include "../src/kernelFunction.h"
#include "../src/sphParticle.h"
#include "../src/equationOfMotion.h"
#include "../src/sphEquation.h"
#include "../src/equationOfState.h"
```

Include dependency graph for unitTests.cpp:

Functions

- TEST (kernelFunction, simpleKernelTest)
- TEST (Vector, SumTest)
- TEST (Vector, SumErrorTest)
- TEST (Vector, innerProductTest)
- TEST (Vector, innerProductErrorTest)
- vector< double > velocity (double t)
- TEST (SPHEquationAndSPHParticle, entropyStarTest)
- TEST (IntegratedTest, entropyStarTest)
- int main (int argc, char *argv[])

5.17.1 Function Documentation

5.17.1.1 int main (int *argc*, char * *argv[]*)

```
239 {
240   ::testing::InitGoogleTest(&argc, argv);
241   return RUN_ALL_TESTS();
242 }
```

5.17.1.2 TEST (kernelFunction , simpleKernelTest)

KERNELFUNCTION CLASS TESTS

References KernelFunction::kernelFunction().

```
24 {
25   vector<double> r1 (2,1);
26   vector<double> r2 (2,0.4);
27   vector<double> r3 (2,0.1);
28   KernelFunction w;
29
30   ASSERT_FLOAT_EQ(w.kernelFunction(r1), 0);
31   ASSERT_FLOAT_EQ(w.kernelFunction(r2), 10./ (3.14159265 * 7 * 0.3 * 0.3) * 0.25 * 0.00149648
      4);
32   ASSERT_FLOAT_EQ(w.kernelFunction(r3), 10./ (3.14159265 * 7 * 0.3 * 0.3) * (1 - (1.5 * 0.471
      404521 * 0.471404521) + (0.75 * 0.471404521 * 0.471404521 * 0.471404521)));
33
34 }
```

5.17.1.3 TEST (Vector , SumTest)

VECTOR CLASS TESTS

```
46 {
47   vector<double> r1(2);
48   vector<double> r2(2);
49
```

```
50   r1[0] = 2; r2[0] = 3;
51   r1[1] = 3; r2[1] = 17;
52
53   vector<double> r3 (r1 + r2);
54
55   ASSERT_EQ(r3[0], 5);
56   ASSERT_EQ(r3[1], 20);
57 }
```

5.17.1.4 TEST (Vector , SumErrorTest)

```
60 {
61   vector<double> r1(2,0.0);
62   vector<double> r2(7,0.0);
63
64   ASSERT_DEATH(r1 + r2, "Vector Sumation Error: Vectors with different dimensions!");
65 }
```

5.17.1.5 TEST (Vector , innerProductTest)

```
68 {
69   vector<double> r1(2);
70   vector<double> r2(2);
71
72   r1[0] = 2; r2[0] = 3;
73   r1[1] = 3; r2[1] = 17;
74
75   double r1r2 = r1 * r2;
76
77   ASSERT_EQ(r1r2, 2*3 + 3*17);
78 }
```

5.17.1.6 TEST (Vector , innerProductErrorTest)

```
81 {
82   vector<double> r1(2,0.0);
83   vector<double> r2(7,0.0);
84
85   ASSERT_DEATH(r1 * r2, "Vector innerProduct Error: Vectors with different dimensions!");
86 }
```

5.17.1.7 TEST (SPHEquationAndSPHParticle , entropyStarTest)

References SPHParticle::position.

```
123 {
124   vector<SPHParticle> listOfParticles;
125
126   // -3 to 3
127
128   for(double y = -3.0; y < 3.0; y += 0.2)
129   {
130     for(double x = -3.0; x < 3.0; x += 0.2)
131     {
132       SPHParticle sphParticle;
133       sphParticle.position[0] = x;
134       sphParticle.position[1] = y;
135
136       listOfParticles.push_back(sphParticle);
137     }
138   }
139
140   for(int i = 0; i < listOfParticles.size(); i++)
141   {
142     SPHEquation sphEquation;
143     KernelFunction w;
144     // cout << sphEquation.entropyStar(listOfParticles[i], listOfParticles, w) << endl;
145   }
146
147 }
```

5.17.1.8 TEST (IntegratedTest , entropyStarTest)

References EquationOfState::calculateEnergyDensity(), EquationOfState::calculateEntropyDensity(), EquationOfState::calculatePressureDensity(), SPHEquation::dMomentum_dTau(), SPHParticle::energyDensity, SPHEquation::entropyStar(), and SPHParticle::position.

```
150 {
151   vector<SPHParticle> listOfParticles;
152   EquationOfState equationOfState;
153   SPHEquation sphEquation;
154   KernelFunction w;
155 
156   // -3 to 3
157   for(double y = -3.0; y < 3.0; y += 0.2)
158   {
159     for(double x = -3.0; x < 3.0; x += 0.2)
160     {
161       SPHParticle sphParticle;
162       sphParticle.position[0] = x;
163       sphParticle.position[1] = y;
164       sphParticle.energyDensity = pow(2,2.666666667)/pow((1 + 2*(1 + x*x + y*y) + pow((1 - x*x
     - y*y),2)),1.333333333);
165 
166       listOfParticles.push_back(sphParticle);
167     }
168   }
169 
170   double totalEnergyDensity = 0.0;
171   for(int i = 0; i < listOfParticles.size(); i++)
172   {
173     totalEnergyDensity += listOfParticles[i].energyDensity;
174   }
175 
176   double totalEntropyDensity = equationOfState.calculateEntropyDensity(
     totalEnergyDensity);
177 
178   int contador = 0;
179   for(double tau = 1.0; tau < 2.5; tau += 0.001)
180   {
181 
182     ofstream file;
183     file.open("teste" + to_string(contador) + ".dat");
184 
185     for(int i = 0; i < listOfParticles.size(); i++)
186     {
187       listOfParticles[i].entropyDensityWeight = equationOfState.
     calculateEntropyDensity(listOfParticles[i].energyDensity)/totalEntropyDensity;
188     }
189 
190     for(int i = 0; i < listOfParticles.size(); i++)
191     {
192       listOfParticles[i].CalculateEntropyDensity(sphEquation.entropyStar(listOfParticles[i],
     listOfParticles, w));
193     }
194     for(int i = 0; i < listOfParticles.size(); i++)
195     {
196       listOfParticles[i].pressureDensity = equationOfState.
     calculatePressureDensity(listOfParticles[i].entropyDensity);
197       listOfParticles[i].pressureDensity = equationOfState.
     calculateEnergyDensity(listOfParticles[i].entropyDensity);
198     }
199 
200     for(int i = 0; i < listOfParticles.size(); i++)
201     {
202       listOfParticles[i].pressureDensity = equationOfState.
     calculatePressureDensity(listOfParticles[i].entropyDensity);
203       listOfParticles[i].pressureDensity = equationOfState.
     calculateEnergyDensity(listOfParticles[i].entropyDensity);
204     }
205 
206     for(int i = 0; i < listOfParticles.size(); i++)
207     {
208 
209       listOfParticles[i].dMomentum_dTau = sphEquation.dMomentum_dTau(listOfParticles[i],
     listOfParticles, w, tau);
210       listOfParticles[i].momentum = listOfParticles[i].momentum + 0.001 * listOfParticles[i].dMomentum_dTau
     ;
211       listOfParticles[i].velocity[0] = (listOfParticles[i].momentum[0]/listOfParticles[i].Gamma()) * (
     listOfParticles[i].entropyDensity / (listOfParticles[i].energyDensity + listOfParticles[i].pressureDensity));
212       listOfParticles[i].velocity[1] = (listOfParticles[i].momentum[1]/listOfParticles[i].Gamma()) * (
     listOfParticles[i].entropyDensity / (listOfParticles[i].energyDensity + listOfParticles[i].pressureDensity));
213       listOfParticles[i].position = listOfParticles[i].position + 0.01 * listOfParticles[i].velocity;
214 
215       if(abs(listOfParticles[i].position[0]) < 0.001)
216       {
```

```
217         file << listOfParticles[i].position[0] << "\t" << listOfParticles[i].position[1] << "\t" <<
      listOfParticles[i].energyDensity << endl;
218       }
219     }
220     cout << "i: " << contador << endl;
221     for(int i = 0; i < listOfParticles.size(); i++)
222     {
223       listOfParticles[i].energyDensity = listOfParticles[i].energyDensity - 0.001 * listOfParticles[i].
      pressureDensity/listOfParticles[i].entropyDensityWeight * listOfParticles[i].entropyDensity;
224     }
225
226
227
228     file.close();
229     contador++;
230   }
231
232
233 }
```

5.17.1.9 vector<double> velocity (double *t*)

EQUATION OF STATE AND SPHPARTICLE TESTS

Referenced by SPHParticle::Gamma(), and SPHParticle::SPHParticle().

```
93 {
94   vector<double> v (2, 10*t);
95   return v;
96 }
```

Index

/Users/rafael/Dropbox/sphene/README.md, 15
/Users/rafael/Dropbox/sphene/src/equationOfMotion.↩
cpp, 15
/Users/rafael/Dropbox/sphene/src/equationOfMotion.h,
15
/Users/rafael/Dropbox/sphene/src/equationOfState.cpp,
15
/Users/rafael/Dropbox/sphene/src/equationOfState.h,
16
/Users/rafael/Dropbox/sphene/src/idealEquationOf↩
State.cpp, 16
/Users/rafael/Dropbox/sphene/src/idealEquationOf↩
State.h, 16
/Users/rafael/Dropbox/sphene/src/kernelFunction.cpp,
16
/Users/rafael/Dropbox/sphene/src/kernelFunction.h, 17
/Users/rafael/Dropbox/sphene/src/main.cpp, 17
/Users/rafael/Dropbox/sphene/src/sphEquation.cpp, 17
/Users/rafael/Dropbox/sphene/src/sphEquation.h, 18
/Users/rafael/Dropbox/sphene/src/sphParticle.cpp, 18
/Users/rafael/Dropbox/sphene/src/sphParticle.h, 18
/Users/rafael/Dropbox/sphene/src/vector.cpp, 18
/Users/rafael/Dropbox/sphene/src/vector.h, 20
/Users/rafael/Dropbox/sphene/test/unitTests.cpp, 21
_PI
equationOfState.cpp, 15
idealEquationOfState.cpp, 16

calculateEnergyDensity
EquationOfState, 8
IdealEquationOfState, 9
CalculateEntropyDensity
SPHParticle, 13
calculateEntropyDensity
EquationOfState, 8
IdealEquationOfState, 9
calculatePressureDensity
EquationOfState, 8
IdealEquationOfState, 10

dMomentum_dTau
SPHEquation, 11
SPHParticle, 14
downParticle
SPHParticle, 14

energyDensity
SPHParticle, 14
entropyDensity
SPHParticle, 14
entropyDensityWeight
SPHParticle, 14
entropyStar
SPHEquation, 12
EquationOfMotion, 7
EquationOfMotion, 7
EulerMethod, 7
RungeKutta4Method, 7
equationOfMotion
SPHParticle, 14
EquationOfState, 8
calculateEnergyDensity, 8
calculateEntropyDensity, 8
calculatePressureDensity, 8
equationOfState.cpp
_PI, 15
EulerMethod
EquationOfMotion, 7

GRAD_W
kernelFunction.cpp, 17
GRAD_W_2
kernelFunction.cpp, 17
Gamma
SPHParticle, 13

H
kernelFunction.cpp, 17
hasDownParticle
SPHParticle, 13
hasLeftParticle
SPHParticle, 14
hasRightParticle
SPHParticle, 14
hasUpParticle
SPHParticle, 14

IdealEquationOfState, 9
calculateEnergyDensity, 9
calculateEntropyDensity, 9
calculatePressureDensity, 10
idealEquationOfState.cpp
_PI, 16

KernelFunction, 10
kernelFunction, 10
kernelGradient, 10
kernelFunction
KernelFunction, 10
kernelFunction.cpp

GRAD_W, 17
GRAD_W_2, 17
H, 17
PI, 17
W, 17
W_2, 17
kernelGradient
KernelFunction, 10

leftParticle
SPHParticle, 14

main
main.cpp, 17
unitTests.cpp, 22
main.cpp
main, 17
momentum
SPHParticle, 14

normOf
vector.cpp, 18
vector.h, 20

operator∗
vector.cpp, 19
vector.h, 20
operator+
vector.cpp, 19
vector.h, 21
operator-
vector.cpp, 19
vector.h, 21

PI
kernelFunction.cpp, 17
position
SPHParticle, 14
pressureDensity
SPHParticle, 14

rightParticle
SPHParticle, 14
RungeKutta4Method
EquationOfMotion, 7

SPHEquation, 11
dMomentum_dTau, 11
entropyStar, 12
SPHEquation, 11
SPHParticle, 12
CalculateEntropyDensity, 13
dMomentum_dTau, 14
downParticle, 14
energyDensity, 14
entropyDensity, 14
entropyDensityWeight, 14
equationOfMotion, 14
Gamma, 13
hasDownParticle, 13
hasLeftParticle, 14
hasRightParticle, 14
hasUpParticle, 14
leftParticle, 14
momentum, 14
position, 14
pressureDensity, 14
rightParticle, 14
SPHParticle, 13
upParticle, 14
velocity, 14

TEST
unitTests.cpp, 22, 23

unitTests.cpp
main, 22
TEST, 22, 23
velocity, 25
upParticle
SPHParticle, 14

vector.cpp
normOf, 18
operator∗, 19
operator+, 19
operator-, 19
vector.h
normOf, 20
operator∗, 20
operator+, 21
operator-, 21
velocity
SPHParticle, 14
unitTests.cpp, 25

W
kernelFunction.cpp, 17
W_2
kernelFunction.cpp, 17