DD2356 : Quantum SPH

Generated by Doxygen 1.8.17

Chapter 1

DD2356_QuantumSPH

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 CSVReader Namespace Reference

Functions

- def read_and_plot (file_name)

4.1.1 Function Documentation

4.1.1.1 read_and_plot()

```
def CSVReader.read_and_plot (
              file_name )
```

```
Read a *.csv file and plot it's contents
Input: file name file_name
```

4.2 quantumsph Namespace Reference

Functions

- def main ()
- def kernel (r, h, deriv)
- def density (x, m, h)
- def pressure (x, rho, m, h)
- def acceleration (x, u, m, rho, P, b, h)
- def probeDensity (x, m, h, xx)

4.2.1 Function Documentation

4.2.1.1 acceleration()

```
def quantumsph.acceleration (
              x,
              u,
              m,
              rho,
              P,
              b,
              h )
```

```
Calculates acceleration of each particle due to quantum pressure, harmonic potential, velocity damping

  Parameters
  ----------
  x : np.array
      Position vector.
  u : np.array
      Velocity vector.
  m : float
      SPH particle mass.
  rho : np.array
      Density vector.
  P : np.array
      Pressure vector.
  b : float
      Damping coefficient.
  h : float
      Scaling length.

  Returns
  -------
  a : np.array
      Acceleration vector.
```

4.2.1.2 density()

```
def quantumsph.density (
              x,
              m,
              h )
```

```
Compute density at each of the particle locations using smoothing kernel

  Parameters
  ----------
  x : np.array
      Position vector.
  m : float
      SPH particle mass.
  h : float
      Scaling length.

  Returns
  -------
  rho : np.array
      Density vector.
```

4.2.1.3 kernel()

```
def quantumsph.kernel (
              r,
              h,
              deriv )
```

```
SPH Gaussian smoothing kernel (1D).

  Parameters
  ----------
  r : np.array
      Distance vector.
  h : float
      Scaling length.
  deriv : int
      Derivative order.

  Returns
  -------
  ker : np.array
      Weight vector.
```

4.2.1.4 main()

```
def quantumsph.main ( )
```

```
Main Loop.
Evolve the time-dependent SE and plot solutions (also saves it as results_py.csv file)
```

4.2.1.5 pressure()

```
def quantumsph.pressure (
              x,
              rho,
              m,
              h )
```

```
Compute ``pressure'' at each of the particles using smoothing kernel
    P = -(1/4)*(d^2 rho /dx^2 - (d rho / dx)^2/rho)

  Parameters
  ----------
  x : np.array
      Position vector.
  rho : np.array
      Density vector.
  m : float
      SPH particle mass.
  h : float
      Scaling length.

  Returns
  -------
  P : np.array
      Pressure vector.
```

4.2.1.6 probeDensity()

```
def quantumsph.probeDensity (
              x,
              m,
              h,
              xx )
```

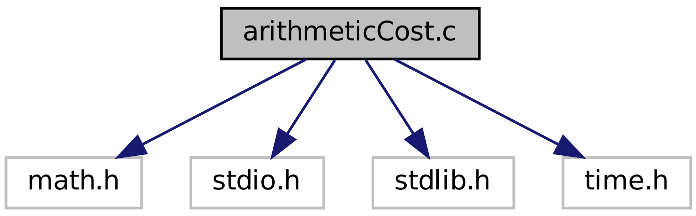
```
Probe the density at specified locations

  Parameters
  ----------
  x : np.array
      Position vector.
  m : float
      SPH particle mass.
  h : float
      Scaling length.
  xx : np.array
      Probe locations.

  Returns
  -------
  rr : np.array
      Density at probing locations.
```

Chapter 5

File Documentation

5.1 arithmeticCost.c File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```
Include dependency graph for arithmeticCost.c:



Macros

- #define N 1000000

  The number of iterations.

Functions

- int main (int argc, char ∗argv[ ])

  Timing kernel.

5.1.1 Macro Definition Documentation

5.1.1.1 N

```
#define N 1000000
```

The number of iterations.

5.1.2 Function Documentation

5.1.2.1 main()

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

Timing kernel.

Compute the execution time of N integer addition, N double precision multiply add, and N calls to the math 'exp' function

5.2 CSVReader.py File Reference

Namespaces

- CSVReader

Functions

- def CSVReader.read_and_plot (file_name)

5.3 quantumsph.py File Reference

Namespaces

- quantumsph

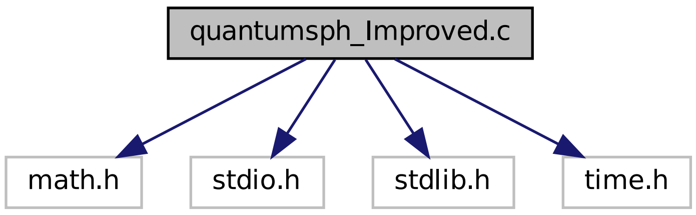
Functions

- def quantumsph.main ()
- def quantumsph.kernel (r, h, deriv)
- def quantumsph.density (x, m, h)
- def quantumsph.pressure (x, rho, m, h)
- def quantumsph.acceleration (x, u, m, rho, P, b, h)
- def quantumsph.probeDensity (x, m, h, xx)

5.4 quantumsph_Improved.c File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

Include dependency graph for quantumsph_Improved.c:



Macros

- #define INV_SQRT_PI 0.564189583547756279280

  Constant equal to 1.0/sqrt(pi)

Functions

- double kernel_0 (double r)

  SPH Gaussian smoothing kernel (1D).
- double kernel_1 (double r)

  SPH Gaussian smoothing kernel (1D), first derivative.
- double kernel_2 (double r)

  SPH Gaussian smoothing kernel (1D), second derivative.
- void pressure (double ∗rP, double ∗x, int n)

  Compute "relative pressure" at each of the particle locations using smoothing kernel.
- void acceleration (double ∗a, double ∗x, double ∗u, double ∗rP, int n, double b)

  Calculates acceleration of each particle.
- void probeDensity (double ∗rr, double ∗x, int n, double ∗xx, int nxx)

  Probe the density at specified locations.
- int main (int argc, char ∗argv[ ])

  Main loop.

Variables

- double h_sq
- double inv_h
- double inv_h_sq
- double inv_h_cb
- double inv_h_5
- double inv_h_7

5.4.1 Macro Definition Documentation

5.4.1.1 INV_SQRT_PI

```
#define INV_SQRT_PI 0.56418958354775627928
```

Constant equal to 1.0/sqrt(pi)

5.4.2 Function Documentation

5.4.2.1 acceleration()

```
void acceleration (
             double * a,
             double * x,
             double * u,
             double * rP,
             int n,
             double b )
```

Calculates acceleration of each particle.

Calculates acceleration of each particle due to quantum pressure, harmonic potential, and velocity damping.

Parameters

a	acceleration vector to be updated
x	positions of the particles
u	velocity vector
rP	relative pressure vector
n	number of particles
b	damping coefficient

5.4.2.2 kernel_0()

```
double kernel_0 (
             double r )
```

SPH Gaussian smoothing kernel (1D).

This function returns the weight associated with the SPH Gaussian smoothing kernel.

Parameters

r	distance between particles

Returns

weight

5.4.2.3 kernel_1()

```
double kernel_1 (
            double r )
```

SPH Gaussian smoothing kernel (1D), first derivative.

This function returns the weight associated with the first derivative of the SPH Gaussian smoothing kernel.

Parameters

r	distance between particles

Returns

weight

5.4.2.4 kernel_2()

```
double kernel_2 (
            double r )
```

SPH Gaussian smoothing kernel (1D), second derivative.

This function returns the weight associated with the second derivative of the SPH Gaussian smoothing kernel.

Parameters

r	distance between particles

Returns

weight

5.4.2.5 main()

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

Main loop.

Evolve the time-dependant Schrödinger equation, and save the solution in "results.csv"

5.4.2.6 pressure()

```
void pressure (
            double * rP,
            double * x,
            int n )
```

Compute "relative pressure" at each of the particle locations using smoothing kernel.

Parameters

rP	pressure vector to be updated, rP = -(1/4)∗(d^2 rho /dx^2 - (d rho / dx)^2/rho)/(rho^2)
x	positions of the particles
n	number of particles

5.4.2.7 probeDensity()

```
void probeDensity (
            double * rr,
            double * x,
            int n,
            double * xx,
            int nxx )
```

Probe the density at specified locations.

Parameters

rr	density at specified locations, vector to be updated
x	positions of the particles
n	number of particles
xx	probe locations
nxx	number of probe locations

5.4.3 Variable Documentation

5.4.3.1 h_sq

```
double h_sq
```

5.4.3.2 inv_h

```
double inv_h
```

5.4.3.3 inv_h_5

```
double inv_h_5
```

5.4.3.4 inv_h_7

```
double inv_h_7
```

5.4.3.5 inv_h_cb

```
double inv_h_cb
```

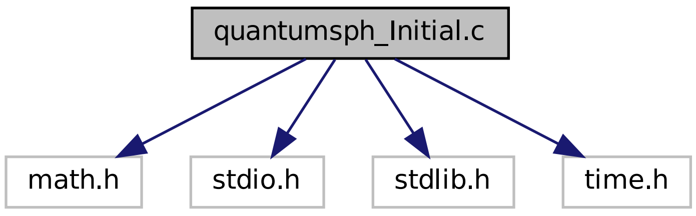
5.4.3.6 inv_h_sq

```
double inv_h_sq
```

5.5 quantumsph_Initial.c File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

Include dependency graph for quantumsph_Initial.c:



Macros

- #define PI 3.14159265358979323846

  The pi constant.

Functions

- double kernel (double r, double h, int deriv)

  SPH Gaussian smoothing kernel (1D).
- void density (double ∗rho, double ∗x, int n, double m, double h)

  Compute density at each of the particle locations using smoothing kernel.
- void pressure (double ∗P, double ∗x, double ∗rho, int n, double m, double h)

  Compute "pressure" at each of the particle locations using smoothing kernel.
- void acceleration (double ∗a, double ∗x, double ∗u, double ∗rho, double ∗P, int n, double m, double b, double h)

  Calculates acceleration of each particle.
- void probeDensity (double ∗rr, double ∗x, int n, double m, double h, double ∗xx, int nxx)

  Probe the density at specified locations.
- int main (int argc, char ∗argv[ ])

  Main loop.

5.5.1 Macro Definition Documentation

5.5.1.1 PI

```
#define PI 3.14159265358979323846
```

The pi constant.

5.5.2 Function Documentation

5.5.2.1 acceleration()

```
void acceleration (
             double ∗ a,
             double ∗ x,
             double ∗ u,
             double ∗ rho,
             double ∗ P,
             int n,
             double m,
             double b,
             double h )
```

Calculates acceleration of each particle.

Calculates acceleration of each particle due to quantum pressure, harmonic potential, and velocity damping.

Parameters

a	acceleration vector to be updated
x	positions of the particles
u	velocity vector
rho	density vector
P	pressure vector
n	number of particles
m	SPH particle mass
b	damping coefficient
h	scaling length

5.5.2.2 density()

```
void density (
            double * rho,
            double * x,
            int n,
            double m,
            double h )
```

Compute density at each of the particle locations using smoothing kernel.

Parameters

rho	density vector to be updated
x	positions of the particles
n	number of particles
m	SPH particle mass
h	scaling length

5.5.2.3 kernel()

```
double kernel (
            double r,
            double h,
            int deriv )
```

SPH Gaussian smoothing kernel (1D).

This function returns the weight associated with the SPH Gaussian smoothing kernel, for the specified derivative order.

Parameters

r	distance between particles
h	scaling length
deriv	derivative order

Returns

weight

5.5.2.4 main()

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

Main loop.

Evolve the time-dependant Schrödinger equation, and save the solution in "results.csv"

5.5.2.5 pressure()

```
void pressure (
            double * P,
            double * x,
            double * rho,
            int n,
            double m,
            double h )
```

Compute "pressure" at each of the particle locations using smoothing kernel.

Parameters

P	pressure vector to be updated, P = -(1/4)∗(d^2 rho /dx^2 - (d rho / dx)^2/rho)
x	positions of the particles
rho	density vector
n	number of particles
m	SPH particle mass
h	scaling length

5.5.2.6 probeDensity()

```
void probeDensity (
            double * rr,
            double * x,
            int n,
            double m,
            double h,
            double * xx,
            int nxx )
```

Probe the density at specified locations.

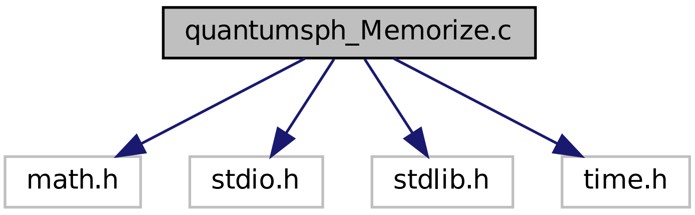
Parameters

rr	density at specified locations, vector to be updated
x	positions of the particles
n	number of particles
m	SPH particle mass
h	scaling length
xx	probe locations
nxx	number of probe locations

5.6 quantumsph_Memorize.c File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

Include dependency graph for quantumsph_Memorize.c:



Macros

- #define INV_SQRT_PI 0.564189583547756279280

  Constant equal to 1.0/sqrt(pi)

Functions

- double ∗∗ InitializeMatrix (int n_row, int n_col)

  Allocate memory space for a matrix.
- void FreeMatrix (double ∗∗A)

  Frees memory space allocated for a matrix.
- void kernel_0_mat (double ∗∗ker, double ∗x, int n)

  SPH Gaussian smoothing kernel (1D), for a set of particles.
- void kernel_1_mat (double ∗∗ker, double ∗x, int n)

  SPH Gaussian smoothing kernel (1D), first derivative, for a set of particles.
- void kernel_2_mat (double ∗∗ker, double ∗x, int n)

SPH Gaussian smoothing kernel (1D), second derivative, for a set of particles.

- double kernel_0 (double r)

  SPH Gaussian smoothing kernel (1D).
- void density (double ∗rho, double ∗∗ker_0, int n)

  Compute density at each of the particle locations using smoothing kernel.
- void pressure (double ∗rP, double ∗rho, double ∗∗ker_0, double ∗∗ker_1, double ∗∗ker_2, int n)

  Compute "relative pressure" at each of the particle locations using smoothing kernel.
- void acceleration (double ∗a, double ∗x, double ∗u, double ∗rP, double ∗∗ker_1, int n, double b)

  Calculates acceleration of each particle.
- void probeDensity (double ∗rr, double ∗x, int n, double ∗xx, int nxx)

  Probe the density at specified locations.
- int main (int argc, char ∗argv[ ])

  Main loop.

Variables

- double h_sq
- double inv_h
- double inv_h_sq
- double inv_h_cb
- double inv_h_5
- double inv_h_7

5.6.1 Macro Definition Documentation

5.6.1.1 INV_SQRT_PI

```
#define INV_SQRT_PI 0.56418958354775627928
```

Constant equal to 1.0/sqrt(pi)

5.6.2 Function Documentation

5.6.2.1 acceleration()

```
void acceleration (
             double * a,
             double * x,
             double * u,
             double * rP,
             double ** ker_1,
             int n,
             double b )
```

Calculates acceleration of each particle.

Calculates acceleration of each particle due to quantum pressure, harmonic potential, and velocity damping.

Parameters

a	acceleration vector to be updated
x	positions of the particles
u	velocity vector
rP	relative pressure vector
ker←→_1	pre-computed first derivative of smoothing kernel - interactions between particles
n	number of particles
b	damping coefficient

5.6.2.2 density()

```
void density (
            double * rho,
            double ** ker_0,
            int n )
```

Compute density at each of the particle locations using smoothing kernel.

Parameters

rho	density vector to be updated
ker←→_0	pre-computed smoothing kernel - interactions between particles
n	number of particles

5.6.2.3 FreeMatrix()

```
void FreeMatrix (
            double ** A )
```

Frees memory space allocated for a matrix.

Parameters

A	matrix, pointer to memory space to be freed

5.6.2.4 InitializeMatrix()

```
double** InitializeMatrix (
            int n_row,
            int n_col )
```

Allocate memory space for a matrix.

This function allocates memory space for a matrix with n_row rows and n_col columns

Parameters

n_row	number of rows
n_col	number of columns

Returns

pointer to space allocated

5.6.2.5 kernel_0()

```
double kernel_0 (
            double r )
```

SPH Gaussian smoothing kernel (1D).

This function returns the weight associated with the SPH Gaussian smoothing kernel.

Parameters

r	distance between particles
h	scaling length

Returns

weight

5.6.2.6 kernel_0_mat()

```
void kernel_0_mat (
            double ** ker,
            double * x,
            int n )
```

SPH Gaussian smoothing kernel (1D), for a set of particles.

This function computes the weight associated with the SPH Gaussian smoothing kernel, for all pairwise interactions of the particles in x. It stores them in ker.

Parameters

ker	matrix to be updated with the computed weights
x	positions of the particles
n	number of particles

5.6.2.7 kernel_1_mat()

```
void kernel_1_mat (
             double ** ker,
             double * x,
             int n )
```

SPH Gaussian smoothing kernel (1D), first derivative, for a set of particles.

This function computes the weight associated with the first derivative of the SPH Gaussian smoothing kernel, for all pairwise interactions of the particles in x. It stores them in ker.

Parameters

ker	matrix to be updated with the computed weights
x	positions of the particles
n	number of particles

5.6.2.8 kernel_2_mat()

```
void kernel_2_mat (
             double ** ker,
             double * x,
             int n )
```

SPH Gaussian smoothing kernel (1D), second derivative, for a set of particles.

This function computes the weight associated with the second derivative of the SPH Gaussian smoothing kernel, for all pairwise interactions of the particles in x. It stores them in ker.

Parameters

ker	matrix to be updated with the computed weights
x	positions of the particles
n	number of particles

5.6.2.9 main()

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

Main loop.

Evolve the time-dependant Schrödinger equation, and save the solution in "results.csv"

5.6.2.10 pressure()

```
void pressure (
            double * rP,
            double * rho,
            double ** ker_0,
            double ** ker_1,
            double ** ker_2,
            int n )
```

Compute "relative pressure" at each of the particle locations using smoothing kernel.

Parameters

rP	pressure vector to be updated, rP = -(1/4)∗(d^2 rho /dx^2 - (d rho / dx)^2/rho)/(rho^2)
rho	density vector
ker←↩_0	pre-computed smoothing kernel - interactions between particles
ker←↩_1	pre-computed first derivative of smoothing kernel - interactions between particles
ker←↩_2	pre-computed second derivative of smoothing kernel - interactions between particles
n	number of particles

5.6.2.11 probeDensity()

```
void probeDensity (
            double * rr,
            double * x,
            int n,
            double * xx,
            int nxx )
```

Probe the density at specified locations.

Parameters

rr	density at specified locations, vector to be updated
x	positions of the particles
n	number of particles
xx	probe locations
nxx	number of probe locations

5.6.3 Variable Documentation

5.6.3.1 h_sq

```
double h_sq
```

5.6.3.2 inv_h

```
double inv_h
```

5.6.3.3 inv_h_5

```
double inv_h_5
```

5.6.3.4 inv_h_7

```
double inv_h_7
```

5.6.3.5 inv_h_cb

```
double inv_h_cb
```

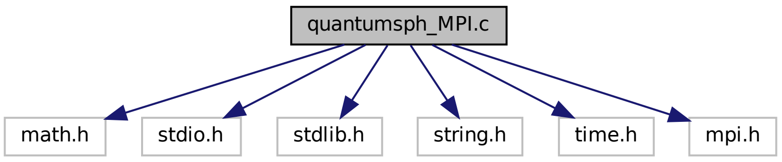
5.6.3.6 inv_h_sq

```
double inv_h_sq
```

5.7 quantumsph_MPI.c File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <mpi.h>
```

Include dependency graph for quantumsph_MPI.c:



Macros

- #define INV_SQRT_PI 0.56418958354775627928

  Constant equal to 1.0/sqrt(pi)

Functions

- double ∗∗ InitializeMatrix (int n_row, int n_col)

  Allocate memory space for a matrix.
- void FreeMatrix (double ∗∗A)

  Frees memory space allocated for a matrix.
- double kernel_0 (double r)

  SPH Gaussian smoothing kernel (1D).
- double kernel_1 (double r)

  SPH Gaussian smoothing kernel (1D), first derivative.
- double kernel_2 (double r)

  SPH Gaussian smoothing kernel (1D), second derivative.
- void relativePressure (double ∗rP, double ∗∗ker_0, double ∗∗ker_1, double ∗x, double ∗x_ext)

  Compute "relative pressure" at each of the particle locations using smoothing kernel.
- void acceleration (double ∗a, double ∗x, double ∗u, double ∗rP, double ∗rP_ext, double ∗∗ker_1, double b)

  Calculates acceleration of each particle.
- void probeDensity (double ∗rr, double ∗x, double ∗x_ext, double ∗xx, int nxx_loc)

  Probe the density at specified locations.
- int main (int argc, char ∗argv[ ])

  Main loop.

Variables

- int n
- int rank
- int size
- int n_loc
- int rem
- double h_sq
- double inv_h
- double inv_h_sq
- double inv_h_cb
- double inv_h_5
- double inv_h_7

5.7.1 Macro Definition Documentation

5.7.1.1 INV_SQRT_PI

```
#define INV_SQRT_PI 0.56418958354775627928
```

Constant equal to 1.0/sqrt(pi)

5.7.2 Function Documentation

5.7.2.1 acceleration()

```
void acceleration (
            double * a,
            double * x,
            double * u,
            double * rP,
            double * rP_ext,
            double ** ker_1,
            double b )
```

Calculates acceleration of each particle.

Calculates acceleration of each particle due to quantum pressure, harmonic potential, and velocity damping.

Parameters

a	acceleration vector to be updated
x	positions of the particles
u	velocity vector
rP	relative pressure vector
rP_ext	allocated memory space for receiving other process's relative pressure vectors
ker↩ _1	pre-computed first derivative of smoothing kernel - interactions between particles
b	damping coefficient

5.7.2.2 FreeMatrix()

```
void FreeMatrix (
            double ** A )
```

Frees memory space allocated for a matrix.

Parameters

A	matrix, pointer to memory space to be freed

5.7.2.3 InitializeMatrix()

```
double** InitializeMatrix (
            int n_row,
            int n_col )
```

Allocate memory space for a matrix.

This function allocates memory space for a matrix with n_row rows and n_col columns

Parameters

n_row	number of rows
n_col	number of columns

Returns

pointer to space allocated

5.7.2.4 kernel_0()

```
double kernel_0 (
            double r )
```

SPH Gaussian smoothing kernel (1D).

This function returns the weight associated with the SPH Gaussian smoothing kernel.

Parameters

r	distance between particles
h	scaling length

Returns

weight

5.7.2.5 kernel_1()

```
double kernel_1 (
            double r )
```

SPH Gaussian smoothing kernel (1D), first derivative.

This function returns the weight associated with the first derivative of the SPH Gaussian smoothing kernel.

Parameters

r	distance between particles

Returns

weight

5.7.2.6 kernel_2()

```
double kernel_2 (
             double r )
```

SPH Gaussian smoothing kernel (1D), second derivative.

This function returns the weight associated with the second derivative of the SPH Gaussian smoothing kernel.

Parameters

r	distance between particles

Returns

weight

5.7.2.7 main()

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

Main loop.

Evolve the time-dependant Schrödinger equation, and save the solution in "results.csv"

5.7.2.8 probeDensity()

```
void probeDensity (
             double * rr,
             double * x,
             double * x_ext,
             double * xx,
             int nxx_loc )
```

Probe the density at specified locations.

Parameters

rr	density at specified locations, vector to be updated
x	positions of the particles
x_ext	allocated memory space for receiving positions of other processes's particles
xx	probe locations
nxx	number of probe locations

5.7.2.9 relativePressure()

```
void relativePressure (
            double * rP,
            double ** ker_0,
            double ** ker_1,
            double * x,
            double * x_ext )
```

Compute "relative pressure" at each of the particle locations using smoothing kernel.

Computes "relative pressure" at each of the particle locations using smoothing kernel. Also stores interactions relative to the smoothing kernel and it's first derivative in ker_0 and ker_1.

Parameters

rP	pressure vector to be updated, rP = -(1/4)∗(d^2 rho /dx^2 - (d rho / dx)^2/rho)/(rho^2)
ker↩ _0	smoothing kernel to be computed - interactions between particles
ker↩ _1	first derivative of smoothing kernel to be computed - interactions between particles
x	positions of the particles
x_ext	allocated memory space for receiving positions of other processes's particles

5.7.3 Variable Documentation

5.7.3.1 h_sq

```
double h_sq
```

5.7.3.2 inv_h

```
double inv_h
```

5.7.3.3 inv_h_5

```
double inv_h_5
```

5.7.3.4 inv_h_7

```
double inv_h_7
```

5.7.3.5 inv_h_cb

```
double inv_h_cb
```

5.7.3.6 inv_h_sq

```
double inv_h_sq
```

5.7.3.7 n

```
int n
```

5.7.3.8 n_loc

```
int n_loc
```

5.7.3.9 rank

```
int rank
```

5.7.3.10 rem

```
int rem
```

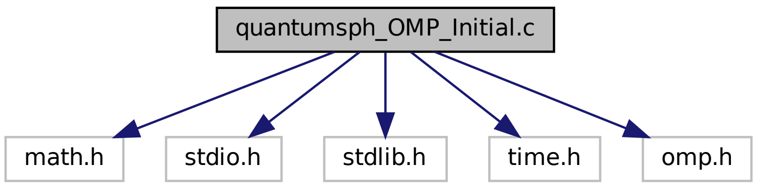
5.7.3.11 size

```
int size
```

5.8 quantumsph_OMP_Initial.c File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>
```

Include dependency graph for quantumsph_OMP_Initial.c:



Macros

- #define INV_SQRT_PI 0.56418958354775627928

  Constant equal to 1.0/sqrt(pi)

Functions

- double ∗∗ InitializeMatrix (int n_row, int n_col)

  Allocate memory space for a matrix.
- void FreeMatrix (double ∗∗A)

  Frees memory space allocated for a matrix.
- void kernel_0_mat (double ∗∗ker, double ∗x, int n)

  SPH Gaussian smoothing kernel (1D), for a set of particles.
- void kernel_1_mat (double ∗∗ker, double ∗x, int n)

  SPH Gaussian smoothing kernel (1D), first derivative, for a set of particles.
- void kernel_2_mat (double ∗∗ker, double ∗x, int n)

  SPH Gaussian smoothing kernel (1D), second derivative, for a set of particles.
- double kernel_0 (double r)

  SPH Gaussian smoothing kernel (1D).
- void relativePressure (double ∗rP, double ∗x, double ∗∗ker_0, double ∗∗ker_1, double ∗∗ker_2, int n)

  Compute "relative pressure" at each of the particle locations using smoothing kernel.
- void acceleration (double ∗a, double ∗x, double ∗u, double ∗rP, double ∗∗ker_1, int n, double b)

  Calculates acceleration of each particle.
- void probeDensity (double ∗rr, double ∗x, int n, double ∗xx, int nxx)

  Probe the density at specified locations.
- int main (int argc, char ∗argv[ ])

  Main loop.

Variables

- double h_sq
- double inv_h
- double inv_h_sq
- double inv_h_cb
- double inv_h_5
- double inv_h_7

5.8.1 Macro Definition Documentation

5.8.1.1 INV_SQRT_PI

```
#define INV_SQRT_PI 0.56418958354775627928
```

Constant equal to 1.0/sqrt(pi)

5.8.2 Function Documentation

5.8.2.1 acceleration()

```
void acceleration (
             double * a,
             double * x,
             double * u,
             double * rP,
             double ** ker_1,
             int n,
             double b )
```

Calculates acceleration of each particle.

Calculates acceleration of each particle due to quantum pressure, harmonic potential, and velocity damping.

Parameters

a	acceleration vector to be updated
x	positions of the particles
u	velocity vector
rP	relative pressure vector
ker_1	pre-computed first derivative of smoothing kernel - interactions between particles
n	number of particles
b	damping coefficient

5.8.2.2 FreeMatrix()

```
void FreeMatrix (
            double ** A )
```

Frees memory space allocated for a matrix.

Parameters

A	matrix, pointer to memory space to be freed

5.8.2.3 InitializeMatrix()

```
double** InitializeMatrix (
            int n_row,
            int n_col )
```

Allocate memory space for a matrix.

This function allocates memory space for a matrix with n_row rows and n_col columns

Parameters

n_row	number of rows
n_col	number of columns

Returns

pointer to space allocated

5.8.2.4 kernel_0()

```
double kernel_0 (
            double r )
```

SPH Gaussian smoothing kernel (1D).

This function returns the weight associated with the SPH Gaussian smoothing kernel.

Parameters

r	distance between particles

Returns

weight

5.8.2.5 kernel_0_mat()

```
void kernel_0_mat (
            double ** ker,
            double * x,
            int n )
```

SPH Gaussian smoothing kernel (1D), for a set of particles.

This function computes the weight associated with the SPH Gaussian smoothing kernel, for all pairwise interactions of the particles in x. It stores them in ker.

Parameters

ker	matrix to be updated with the computed weights
x	positions of the particles
n	number of particles

5.8.2.6 kernel_1_mat()

```
void kernel_1_mat (
            double ** ker,
            double * x,
            int n )
```

SPH Gaussian smoothing kernel (1D), first derivative, for a set of particles.

This function computes the weight associated with the first derivative of the SPH Gaussian smoothing kernel, for all pairwise interactions of the particles in x. It stores them in ker.

Parameters

ker	matrix to be updated with the computed weights
x	positions of the particles
n	number of particles

5.8.2.7 kernel_2_mat()

```
void kernel_2_mat (
            double ** ker,
            double * x,
            int n )
```

SPH Gaussian smoothing kernel (1D), second derivative, for a set of particles.

This function computes the weight associated with the second derivative of the SPH Gaussian smoothing kernel, for all pairwise interactions of the particles in x. It stores them in ker.

Parameters

ker	matrix to be updated with the computed weights
x	positions of the particles
n	number of particles

5.8.2.8 main()

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

Main loop.

Evolve the time-dependant Schrödinger equation, and save the solution in "results.csv"

5.8.2.9 probeDensity()

```
void probeDensity (
            double * rr,
            double * x,
            int n,
            double * xx,
            int nxx )
```

Probe the density at specified locations.

Parameters

rr	density at specified locations, vector to be updated
x	positions of the particles
n	number of particles
xx	probe locations
nxx	number of probe locations

5.8.2.10 relativePressure()

```
void relativePressure (
            double * rP,
            double * x,
            double ** ker_0,
            double ** ker_1,
            double ** ker_2,
            int n )
```

Compute "relative pressure" at each of the particle locations using smoothing kernel.

Parameters

rP	pressure vector to be updated, rP = -(1/4)*(d^2 rho /dx^2 - (d rho / dx)^2/rho)/(rho^2)
x	positions of the particles
ker←→_0	pre-computed smoothing kernel - interactions between particles
ker←→_1	pre-computed first derivative of smoothing kernel - interactions between particles
ker←→_2	pre-computed second derivative of smoothing kernel - interactions between particles
n	number of particles

5.8.3 Variable Documentation

5.8.3.1 h_sq

```
double h_sq
```

5.8.3.2 inv_h

```
double inv_h
```

5.8.3.3 inv_h_5

```
double inv_h_5
```

5.8.3.4 inv_h_7

```
double inv_h_7
```

5.8.3.5 inv_h_cb

```
double inv_h_cb
```

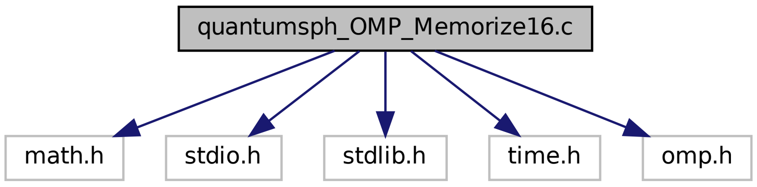
5.8.3.6 inv_h_sq

```
double inv_h_sq
```

5.9 quantumsph_OMP_Memorize16.c File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>
```

Include dependency graph for quantumsph_OMP_Memorize16.c:



Macros

- #define INV_SQRT_PI 0.564189583547756279280

    Constant equal to 1.0/sqrt(pi)

Functions

- double ∗∗ InitializeMatrix (int n_row, int n_col)

    Allocate memory space for a matrix.
- void FreeMatrix (double ∗∗A)

    Frees memory space allocated for a matrix.
- void kernel_0_mat (double ∗∗ker, double ∗x, int n)

    SPH Gaussian smoothing kernel (1D), for a set of particles.
- void kernel_1_mat (double ∗∗ker, double ∗x, int n)

    SPH Gaussian smoothing kernel (1D), first derivative, for a set of particles.
- void kernel_2_mat (double ∗∗ker, double ∗x, int n)

    SPH Gaussian smoothing kernel (1D), second derivative, for a set of particles.

- double kernel_0 (double r)

  SPH Gaussian smoothing kernel (1D).
- void relativePressure (double ∗rP, double ∗x, double ∗∗ker_0, double ∗∗ker_1, double ∗∗ker_2, int n)

  Compute "relative pressure" at each of the particle locations using smoothing kernel.
- void acceleration (double ∗a, double ∗x, double ∗u, double ∗rP, double ∗∗ker_1, int n, double b)

  Calculates acceleration of each particle.
- void probeDensity (double ∗rr, double ∗x, int n, double ∗xx, int nxx)

  Probe the density at specified locations.
- int main (int argc, char ∗argv[ ])

  Main loop.

Variables

- double h_sq
- double inv_h
- double inv_h_sq
- double inv_h_cb
- double inv_h_5
- double inv_h_7

5.9.1 Macro Definition Documentation

5.9.1.1 INV_SQRT_PI

```
#define INV_SQRT_PI 0.56418958354775627928
```

Constant equal to 1.0/sqrt(pi)

5.9.2 Function Documentation

5.9.2.1 acceleration()

```
void acceleration (
            double * a,
            double * x,
            double * u,
            double * rP,
            double ** ker_1,
            int n,
            double b )
```

Calculates acceleration of each particle.

Calculates acceleration of each particle due to quantum pressure, harmonic potential, and velocity damping.

Parameters

a	acceleration vector to be updated
x	positions of the particles
u	velocity vector
rP	relative pressure vector
ker←↩ _1	pre-computed first derivative of smoothing kernel - interactions between particles
n	number of particles
b	damping coefficient

5.9.2.2 FreeMatrix()

```
void FreeMatrix (
             double ** A )
```

Frees memory space allocated for a matrix.

Parameters

A	matrix, pointer to memory space to be freed

5.9.2.3 InitializeMatrix()

```
double** InitializeMatrix (
             int n_row,
             int n_col )
```

Allocate memory space for a matrix.

This function allocates memory space for a matrix with n_row rows and n_col columns

Parameters

n_row	number of rows
n_col	number of columns

Returns

pointer to space allocated

5.9.2.4 kernel_0()

```
double kernel_0 (
            double r )
```

SPH Gaussian smoothing kernel (1D).

This function returns the weight associated with the SPH Gaussian smoothing kernel.

Parameters

r	distance between particles

Returns

weight

5.9.2.5 kernel_0_mat()

```
void kernel_0_mat (
            double ** ker,
            double * x,
            int n )
```

SPH Gaussian smoothing kernel (1D), for a set of particles.

This function computes the weight associated with the SPH Gaussian smoothing kernel, for all pairwise interactions of the particles in x. It stores them in ker.

Parameters

ker	matrix to be updated with the computed weights
x	positions of the particles
n	number of particles

5.9.2.6 kernel_1_mat()

```
void kernel_1_mat (
            double ** ker,
            double * x,
            int n )
```

SPH Gaussian smoothing kernel (1D), first derivative, for a set of particles.

This function computes the weight associated with the first derivative of the SPH Gaussian smoothing kernel, for all pairwise interactions of the particles in x. It stores them in ker.

Parameters

ker	matrix to be updated with the computed weights
x	positions of the particles
n	number of particles

5.9.2.7 kernel_2_mat()

```
void kernel_2_mat (
            double ** ker,
            double * x,
            int n )
```

SPH Gaussian smoothing kernel (1D), second derivative, for a set of particles.

This function computes the weight associated with the second derivative of the SPH Gaussian smoothing kernel, for all pairwise interactions of the particles in x. It stores them in ker.

Parameters

ker	matrix to be updated with the computed weights
x	positions of the particles
n	number of particles

5.9.2.8 main()

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

Main loop.

Evolve the time-dependant Schrödinger equation, and save the solution in "results.csv"

5.9.2.9 probeDensity()

```
void probeDensity (
            double * rr,
            double * x,
            int n,
            double * xx,
            int nxx )
```

Probe the density at specified locations.

Parameters

rr	density at specified locations, vector to be updated
x	positions of the particles
n	number of particles
xx	probe locations
nxx	number of probe locations

5.9.2.10 relativePressure()

```
void relativePressure (
             double * rP,
             double * x,
             double ** ker_0,
             double ** ker_1,
             double ** ker_2,
             int n )
```

Compute "relative pressure" at each of the particle locations using smoothing kernel.

Parameters

rP	pressure vector to be updated, rP = -(1/4)∗(d^2 rho /dx^2 - (d rho / dx)^2/rho)/(rho^2)
x	positions of the particles
ker←↩ _0	pre-computed smoothing kernel - interactions between particles
ker←↩ _1	pre-computed first derivative of smoothing kernel - interactions between particles
ker←↩ _2	pre-computed second derivative of smoothing kernel - interactions between particles
n	number of particles

5.9.3 Variable Documentation

5.9.3.1 h_sq

```
double h_sq
```

5.9.3.2 inv_h

```
double inv_h
```

5.9.3.3 inv_h_5

```
double inv_h_5
```

5.9.3.4 inv_h_7

```
double inv_h_7
```

5.9.3.5 inv_h_cb

```
double inv_h_cb
```

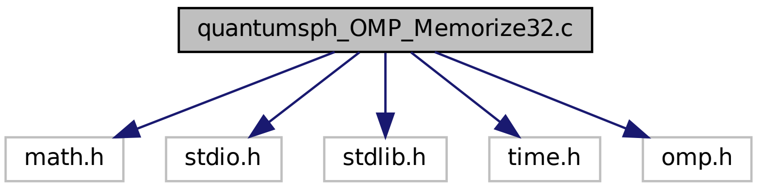
5.9.3.6 inv_h_sq

```
double inv_h_sq
```

5.10 quantumsph_OMP_Memorize32.c File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>
```

Include dependency graph for quantumsph_OMP_Memorize32.c:



Macros

- #define INV_SQRT_PI 0.56418958354775627928

  Constant equal to 1.0/sqrt(pi)

Functions

- double ∗∗ InitializeMatrix (int n_row, int n_col)

  Allocate memory space for a matrix.
- void FreeMatrix (double ∗∗A)

  Frees memory space allocated for a matrix.
- void kernel_0_mat (double ∗∗ker, double ∗x, int n)

  SPH Gaussian smoothing kernel (1D), for a set of particles.
- void kernel_1_mat (double ∗∗ker, double ∗x, int n)

  SPH Gaussian smoothing kernel (1D), first derivative, for a set of particles.
- void kernel_2_mat (double ∗∗ker, double ∗x, int n)

  SPH Gaussian smoothing kernel (1D), second derivative, for a set of particles.
- double kernel_0 (double r)

  SPH Gaussian smoothing kernel (1D).
- void relativePressure (double ∗rP, double ∗x, double ∗∗ker_0, double ∗∗ker_1, double ∗∗ker_2, int n)

  Compute "relative pressure" at each of the particle locations using smoothing kernel.
- void acceleration (double ∗a, double ∗x, double ∗u, double ∗rP, double ∗∗ker_1, int n, double b)

  Calculates acceleration of each particle.
- void probeDensity (double ∗rr, double ∗x, int n, double ∗xx, int nxx)

  Probe the density at specified locations.
- int main (int argc, char ∗argv[ ])

  Main loop.

Variables

- double h_sq
- double inv_h
- double inv_h_sq
- double inv_h_cb
- double inv_h_5
- double inv_h_7

5.10.1 Macro Definition Documentation

5.10.1.1 INV_SQRT_PI

```
#define INV_SQRT_PI 0.56418958354775627928
```

Constant equal to 1.0/sqrt(pi)

5.10.2 Function Documentation

5.10.2.1 acceleration()

```
void acceleration (
             double * a,
             double * x,
             double * u,
             double * rP,
             double ** ker_1,
             int n,
             double b )
```

Calculates acceleration of each particle.

Calculates acceleration of each particle due to quantum pressure, harmonic potential, and velocity damping.

Parameters

a	acceleration vector to be updated
x	positions of the particles
u	velocity vector
rP	relative pressure vector
ker←→_1	pre-computed first derivative of smoothing kernel - interactions between particles
n	number of particles
b	damping coefficient

5.10.2.2 FreeMatrix()

```
void FreeMatrix (
             double ** A )
```

Frees memory space allocated for a matrix.

Parameters

A	matrix, pointer to memory space to be freed

5.10.2.3 InitializeMatrix()

```
double** InitializeMatrix (
             int n_row,
             int n_col )
```

Allocate memory space for a matrix.

This function allocates memory space for a matrix with n_row rows and n_col columns

Parameters

n_row	number of rows
n_col	number of columns

Returns

pointer to space allocated

5.10.2.4 kernel_0()

```
double kernel_0 (
            double r )
```

SPH Gaussian smoothing kernel (1D).

This function returns the weight associated with the SPH Gaussian smoothing kernel.

Parameters

r	distance between particles

Returns

weight

5.10.2.5 kernel_0_mat()

```
void kernel_0_mat (
            double ** ker,
            double * x,
            int n )
```

SPH Gaussian smoothing kernel (1D), for a set of particles.

This function computes the weight associated with the SPH Gaussian smoothing kernel, for all pairwise interactions of the particles in x. It stores them in ker.

Parameters

ker	matrix to be updated with the computed weights
x	positions of the particles
n	number of particles

5.10.2.6 kernel_1_mat()

```
void kernel_1_mat (
            double ** ker,
            double * x,
            int n )
```

SPH Gaussian smoothing kernel (1D), first derivative, for a set of particles.

This function computes the weight associated with the first derivative of the SPH Gaussian smoothing kernel, for all pairwise interactions of the particles in x. It stores them in ker.

Parameters

ker	matrix to be updated with the computed weights
x	positions of the particles
n	number of particles

5.10.2.7 kernel_2_mat()

```
void kernel_2_mat (
            double ** ker,
            double * x,
            int n )
```

SPH Gaussian smoothing kernel (1D), second derivative, for a set of particles.

This function computes the weight associated with the second derivative of the SPH Gaussian smoothing kernel, for all pairwise interactions of the particles in x. It stores them in ker.

Parameters

ker	matrix to be updated with the computed weights
x	positions of the particles
n	number of particles

5.10.2.8 main()

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

Main loop.

Evolve the time-dependant Schrödinger equation, and save the solution in "results.csv"

5.10.2.9 probeDensity()

```
void probeDensity (
            double * rr,
            double * x,
            int n,
            double * xx,
            int nxx )
```

Probe the density at specified locations.

Parameters

rr	density at specified locations, vector to be updated
x	positions of the particles
n	number of particles
xx	probe locations
nxx	number of probe locations

5.10.2.10 relativePressure()

```
void relativePressure (
            double * rP,
            double * x,
            double ** ker_0,
            double ** ker_1,
            double ** ker_2,
            int n )
```

Compute "relative pressure" at each of the particle locations using smoothing kernel.

Parameters

rP	pressure vector to be updated, rP = -(1/4)∗(d^2 rho /dx^2 - (d rho / dx)^2/rho)/(rho^2)
x	positions of the particles
ker←_0	pre-computed smoothing kernel - interactions between particles
ker←_1	pre-computed first derivative of smoothing kernel - interactions between particles
ker←_2	pre-computed second derivative of smoothing kernel - interactions between particles
n	number of particles

5.10.3 Variable Documentation

5.10.3.1 h_sq

```
double h_sq
```

5.10.3.2 inv_h

```
double inv_h
```

5.10.3.3 inv_h_5

```
double inv_h_5
```

5.10.3.4 inv_h_7

```
double inv_h_7
```

5.10.3.5 inv_h_cb

```
double inv_h_cb
```

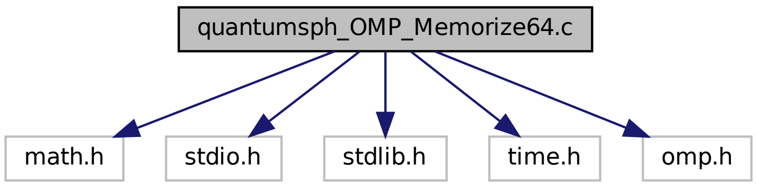
5.10.3.6 inv_h_sq

```
double inv_h_sq
```

5.11 quantumsph_OMP_Memorize64.c File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>
```

Include dependency graph for quantumsph_OMP_Memorize64.c:



Macros

- #define INV_SQRT_PI 0.56418958354775627928

    Constant equal to 1.0/sqrt(pi)

Functions

- double ∗∗ InitializeMatrix (int n_row, int n_col)

    Allocate memory space for a matrix.
- void FreeMatrix (double ∗∗A)

    Frees memory space allocated for a matrix.
- void kernel_0_mat (double ∗∗ker, double ∗x, int n)

    SPH Gaussian smoothing kernel (1D), for a set of particles.
- void kernel_1_mat (double ∗∗ker, double ∗x, int n)

    SPH Gaussian smoothing kernel (1D), first derivative, for a set of particles.
- void kernel_2_mat (double ∗∗ker, double ∗x, int n)

    SPH Gaussian smoothing kernel (1D), second derivative, for a set of particles.
- double kernel_0 (double r)

    SPH Gaussian smoothing kernel (1D).
- void relativePressure (double ∗rP, double ∗x, double ∗∗ker_0, double ∗∗ker_1, double ∗∗ker_2, int n)

    Compute "relative pressure" at each of the particle locations using smoothing kernel.
- void acceleration (double ∗a, double ∗x, double ∗u, double ∗rP, double ∗∗ker_1, int n, double b)

    Calculates acceleration of each particle.
- void probeDensity (double ∗rr, double ∗x, int n, double ∗xx, int nxx)

    Probe the density at specified locations.
- int main (int argc, char ∗argv[ ])

    Main loop.

Variables

- double h_sq
- double inv_h
- double inv_h_sq
- double inv_h_cb
- double inv_h_5
- double inv_h_7

5.11.1 Macro Definition Documentation

5.11.1.1 INV_SQRT_PI

```
#define INV_SQRT_PI 0.56418958354775627928
```

Constant equal to 1.0/sqrt(pi)

5.11.2 Function Documentation

5.11.2.1 acceleration()

```
void acceleration (
            double * a,
            double * x,
            double * u,
            double * rP,
            double ** ker_1,
            int n,
            double b )
```

Calculates acceleration of each particle.

Calculates acceleration of each particle due to quantum pressure, harmonic potential, and velocity damping.

Parameters

a	acceleration vector to be updated
x	positions of the particles
u	velocity vector
rP	relative pressure vector
ker←↩ _1	pre-computed first derivative of smoothing kernel - interactions between particles
n	number of particles
b	damping coefficient

5.11.2.2 FreeMatrix()

```
void FreeMatrix (
            double ** A )
```

Frees memory space allocated for a matrix.

Parameters

A	matrix, pointer to memory space to be freed

5.11.2.3 InitializeMatrix()

```
double** InitializeMatrix (
            int n_row,
            int n_col )
```

Allocate memory space for a matrix.

This function allocates memory space for a matrix with n_row rows and n_col columns

Parameters

n_row	number of rows
n_col	number of columns

Returns

pointer to space allocated

5.11.2.4 kernel_0()

```
double kernel_0 (
             double r )
```

SPH Gaussian smoothing kernel (1D).

This function returns the weight associated with the SPH Gaussian smoothing kernel.

Parameters

r	distance between particles

Returns

weight

5.11.2.5 kernel_0_mat()

```
void kernel_0_mat (
             double ** ker,
             double * x,
             int n )
```

SPH Gaussian smoothing kernel (1D), for a set of particles.

This function computes the weight associated with the SPH Gaussian smoothing kernel, for all pairwise interactions of the particles in x. It stores them in ker.

Parameters

ker	matrix to be updated with the computed weights
x	positions of the particles
n	number of particles

5.11.2.6 kernel_1_mat()

```
void kernel_1_mat (
            double ** ker,
            double * x,
            int n )
```

SPH Gaussian smoothing kernel (1D), first derivative, for a set of particles.

This function computes the weight associated with the first derivative of the SPH Gaussian smoothing kernel, for all pairwise interactions of the particles in x. It stores them in ker.

Parameters

ker	matrix to be updated with the computed weights
x	positions of the particles
n	number of particles

5.11.2.7 kernel_2_mat()

```
void kernel_2_mat (
            double ** ker,
            double * x,
            int n )
```

SPH Gaussian smoothing kernel (1D), second derivative, for a set of particles.

This function computes the weight associated with the second derivative of the SPH Gaussian smoothing kernel, for all pairwise interactions of the particles in x. It stores them in ker.

Parameters

ker	matrix to be updated with the computed weights
x	positions of the particles
n	number of particles

5.11.2.8 main()

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

Main loop.

Evolve the time-dependant Schrödinger equation, and save the solution in "results.csv"

5.11.2.9 probeDensity()

```
void probeDensity (
            double * rr,
            double * x,
            int n,
            double * xx,
            int nxx )
```

Probe the density at specified locations.

Parameters

rr	density at specified locations, vector to be updated
x	positions of the particles
n	number of particles
xx	probe locations
nxx	number of probe locations

5.11.2.10 relativePressure()

```
void relativePressure (
            double * rP,
            double * x,
            double ** ker_0,
            double ** ker_1,
            double ** ker_2,
            int n )
```

Compute "relative pressure" at each of the particle locations using smoothing kernel.

Parameters

rP	pressure vector to be updated, rP = -(1/4)∗(d^2 rho /dx^2 - (d rho / dx)^2/rho)/(rho^2)
x	positions of the particles
ker↩ _0	pre-computed smoothing kernel - interactions between particles
ker↩ _1	pre-computed first derivative of smoothing kernel - interactions between particles
ker↩ _2	pre-computed second derivative of smoothing kernel - interactions between particles
n	number of particles

5.11.3 Variable Documentation

5.11.3.1 h_sq

```
double h_sq
```

5.11.3.2 inv_h

```
double inv_h
```

5.11.3.3 inv_h_5

```
double inv_h_5
```

5.11.3.4 inv_h_7

```
double inv_h_7
```

5.11.3.5 inv_h_cb

```
double inv_h_cb
```

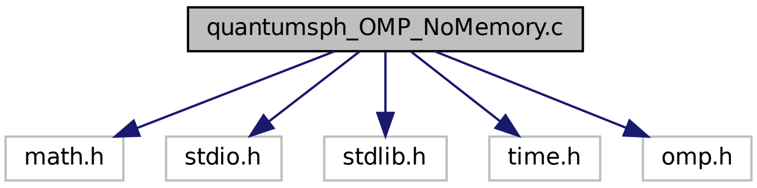
5.11.3.6 inv_h_sq

```
double inv_h_sq
```

5.12 quantumsph_OMP_NoMemory.c File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>
```

Include dependency graph for quantumsph_OMP_NoMemory.c:



Macros

- #define INV_SQRT_PI 0.56418958354775627928

  Constant equal to 1.0/sqrt(pi)

Functions

- double kernel_0 (double r)

  SPH Gaussian smoothing kernel (1D).
- double kernel_1 (double r)

  SPH Gaussian smoothing kernel (1D), first derivative.
- double kernel_2 (double r)

  SPH Gaussian smoothing kernel (1D), second derivative.
- void pressure (double ∗rP, double ∗x, int n)

  Compute "relative pressure" at each of the particle locations using smoothing kernel.
- void acceleration (double ∗a, double ∗x, double ∗u, double ∗rP, int n, double b)

  Calculates acceleration of each particle.
- void probeDensity (double ∗rr, double ∗x, int n, double ∗xx, int nxx)

  Probe the density at specified locations.
- int main (int argc, char ∗argv[ ])

  Main loop.

Variables

- double h_sq
- double inv_h
- double inv_h_sq
- double inv_h_cb
- double inv_h_5
- double inv_h_7

5.12.1 Macro Definition Documentation

5.12.1.1 INV_SQRT_PI

```
#define INV_SQRT_PI 0.56418958354775627928
```

Constant equal to 1.0/sqrt(pi)

5.12.2 Function Documentation

5.12.2.1 acceleration()

```
void acceleration (
             double * a,
             double * x,
             double * u,
             double * rP,
             int n,
             double b )
```

Calculates acceleration of each particle.

Calculates acceleration of each particle due to quantum pressure, harmonic potential, and velocity damping.

Parameters

a	acceleration vector to be updated
x	positions of the particles
u	velocity vector
rP	relative pressure vector
n	number of particles
b	damping coefficient

5.12.2.2 kernel_0()

```
double kernel_0 (
             double r )
```

SPH Gaussian smoothing kernel (1D).

This function returns the weight associated with the SPH Gaussian smoothing kernel.

Parameters

r	distance between particles
h	scaling length

Returns

weight

5.12.2.3 kernel_1()

```
double kernel_1 (
             double r )
```

SPH Gaussian smoothing kernel (1D), first derivative.

This function returns the weight associated with the first derivative of the SPH Gaussian smoothing kernel.

Parameters

r	distance between particles

Returns

weight

5.12.2.4 kernel_2()

```
double kernel_2 (
             double r )
```

SPH Gaussian smoothing kernel (1D), second derivative.

This function returns the weight associated with the second derivative of the SPH Gaussian smoothing kernel.

Parameters

r	distance between particles

Returns

weight

5.12.2.5 main()

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

Main loop.

Evolve the time-dependant Schrödinger equation, and save the solution in "results.csv"

5.12.2.6 pressure()

```
void pressure (
             double * rP,
             double * x,
             int n )
```

Compute "relative pressure" at each of the particle locations using smoothing kernel.

Parameters

rP	pressure vector to be updated, rP = -(1/4)∗(d^2 rho /dx^2 - (d rho / dx)^2/rho)/(rho^2)
x	positions of the particles
n	number of particles

5.12.2.7 probeDensity()

```
void probeDensity (
             double * rr,
             double * x,
             int n,
             double * xx,
             int nxx )
```

Probe the density at specified locations.

Parameters

rr	density at specified locations, vector to be updated
x	positions of the particles
n	number of particles
xx	probe locations
nxx	number of probe locations

5.12.3 Variable Documentation

5.12.3.1 h_sq

```
double h_sq
```

5.12.3.2 inv_h

```
double inv_h
```

5.12.3.3 inv_h_5

```
double inv_h_5
```

5.12.3.4 inv_h_7

```
double inv_h_7
```

5.12.3.5 inv_h_cb

```
double inv_h_cb
```

5.12.3.6 inv_h_sq

```
double inv_h_sq
```

5.13 README.md File Reference

Index