

DD2356 : Quantum SPH

Generated by Doxygen 1.8.17

| | |
|--|-----------|
| 1 DD2356_QuantumSPH | 1 |
| 2 Namespace Index | 3 |
| 2.1 Namespace List | 3 |
| 3 File Index | 5 |
| 3.1 File List | 5 |
| 4 Namespace Documentation | 7 |
| 4.1 CSVReader Namespace Reference | 7 |
| 4.1.1 Function Documentation | 7 |
| 4.1.1.1 read_and_plot() | 7 |
| 4.2 quantumsph Namespace Reference | 7 |
| 4.2.1 Function Documentation | 8 |
| 4.2.1.1 acceleration() | 8 |
| 4.2.1.2 density() | 8 |
| 4.2.1.3 kernel() | 9 |
| 4.2.1.4 main() | 9 |
| 4.2.1.5 pressure() | 9 |
| 4.2.1.6 probeDensity() | 10 |
| 5 File Documentation | 11 |
| 5.1 arithmeticCost.c File Reference | 11 |
| 5.1.1 Macro Definition Documentation | 11 |
| 5.1.1.1 N | 12 |
| 5.1.2 Function Documentation | 12 |
| 5.1.2.1 main() | 12 |
| 5.2 CSVReader.py File Reference | 12 |
| 5.3 quantumsph.py File Reference | 12 |
| 5.4 quantumsph_Improved.c File Reference | 13 |
| 5.4.1 Macro Definition Documentation | 14 |
| 5.4.1.1 INV_SQRT_PI | 14 |
| 5.4.2 Function Documentation | 14 |
| 5.4.2.1 acceleration() | 14 |
| 5.4.2.2 kernel_0() | 14 |
| 5.4.2.3 kernel_1() | 15 |
| 5.4.2.4 kernel_2() | 15 |
| 5.4.2.5 main() | 16 |
| 5.4.2.6 pressure() | 16 |
| 5.4.2.7 probeDensity() | 16 |
| 5.4.3 Variable Documentation | 16 |
| 5.4.3.1 h_sq | 17 |
| 5.4.3.2 inv_h | 17 |
| 5.4.3.3 inv_h_5 | 17 |

| | |
|--|----|
| 5.4.3.4 inv_h_7 | 17 |
| 5.4.3.5 inv_h_cb | 17 |
| 5.4.3.6 inv_h_sq | 17 |
| 5.5 quantumsph_Initial.c File Reference | 17 |
| 5.5.1 Macro Definition Documentation | 18 |
| 5.5.1.1 PI | 18 |
| 5.5.2 Function Documentation | 18 |
| 5.5.2.1 acceleration() | 18 |
| 5.5.2.2 density() | 19 |
| 5.5.2.3 kernel() | 19 |
| 5.5.2.4 main() | 20 |
| 5.5.2.5 pressure() | 20 |
| 5.5.2.6 probeDensity() | 20 |
| 5.6 quantumsph_Memorize.c File Reference | 21 |
| 5.6.1 Macro Definition Documentation | 22 |
| 5.6.1.1 INV_SQRT_PI | 22 |
| 5.6.2 Function Documentation | 22 |
| 5.6.2.1 acceleration() | 22 |
| 5.6.2.2 density() | 23 |
| 5.6.2.3 FreeMatrix() | 23 |
| 5.6.2.4 InitializeMatrix() | 23 |
| 5.6.2.5 kernel_0() | 24 |
| 5.6.2.6 kernel_0_mat() | 24 |
| 5.6.2.7 kernel_1_mat() | 25 |
| 5.6.2.8 kernel_2_mat() | 25 |
| 5.6.2.9 main() | 25 |
| 5.6.2.10 pressure() | 26 |
| 5.6.2.11 probeDensity() | 26 |
| 5.6.3 Variable Documentation | 26 |
| 5.6.3.1 h_sq | 27 |
| 5.6.3.2 inv_h | 27 |
| 5.6.3.3 inv_h_5 | 27 |
| 5.6.3.4 inv_h_7 | 27 |
| 5.6.3.5 inv_h_cb | 27 |
| 5.6.3.6 inv_h_sq | 27 |
| 5.7 quantumsph_MPI.c File Reference | 27 |
| 5.7.1 Macro Definition Documentation | 28 |
| 5.7.1.1 INV_SQRT_PI | 28 |
| 5.7.2 Function Documentation | 29 |
| 5.7.2.1 acceleration() | 29 |
| 5.7.2.2 FreeMatrix() | 29 |
| 5.7.2.3 InitializeMatrix() | 29 |

| | |
|--|----|
| 5.7.2.4 kernel_0() | 30 |
| 5.7.2.5 kernel_1() | 30 |
| 5.7.2.6 kernel_2() | 31 |
| 5.7.2.7 main() | 31 |
| 5.7.2.8 probeDensity() | 31 |
| 5.7.2.9 relativePressure() | 32 |
| 5.7.3 Variable Documentation | 32 |
| 5.7.3.1 h_sq | 32 |
| 5.7.3.2 inv_h | 32 |
| 5.7.3.3 inv_h_5 | 32 |
| 5.7.3.4 inv_h_7 | 33 |
| 5.7.3.5 inv_h_cb | 33 |
| 5.7.3.6 inv_h_sq | 33 |
| 5.7.3.7 n | 33 |
| 5.7.3.8 n_loc | 33 |
| 5.7.3.9 rank | 33 |
| 5.7.3.10 rem | 33 |
| 5.7.3.11 size | 33 |
| 5.8 quantumsph_OMP_Initial.c File Reference | 34 |
| 5.8.1 Macro Definition Documentation | 35 |
| 5.8.1.1 INV_SQRT_PI | 35 |
| 5.8.2 Function Documentation | 35 |
| 5.8.2.1 acceleration() | 35 |
| 5.8.2.2 FreeMatrix() | 36 |
| 5.8.2.3 InitializeMatrix() | 36 |
| 5.8.2.4 kernel_0() | 36 |
| 5.8.2.5 kernel_0_mat() | 37 |
| 5.8.2.6 kernel_1_mat() | 37 |
| 5.8.2.7 kernel_2_mat() | 37 |
| 5.8.2.8 main() | 38 |
| 5.8.2.9 probeDensity() | 38 |
| 5.8.2.10 relativePressure() | 38 |
| 5.8.3 Variable Documentation | 39 |
| 5.8.3.1 h_sq | 39 |
| 5.8.3.2 inv_h | 39 |
| 5.8.3.3 inv_h_5 | 39 |
| 5.8.3.4 inv_h_7 | 39 |
| 5.8.3.5 inv_h_cb | 40 |
| 5.8.3.6 inv_h_sq | 40 |
| 5.9 quantumsph_OMP_Memorize16.c File Reference | 40 |
| 5.9.1 Macro Definition Documentation | 41 |
| 5.9.1.1 INV_SQRT_PI | 41 |

| | |
|---|----|
| 5.9.2 Function Documentation | 41 |
| 5.9.2.1 acceleration() | 41 |
| 5.9.2.2 FreeMatrix() | 42 |
| 5.9.2.3 InitializeMatrix() | 42 |
| 5.9.2.4 kernel_0() | 43 |
| 5.9.2.5 kernel_0_mat() | 43 |
| 5.9.2.6 kernel_1_mat() | 43 |
| 5.9.2.7 kernel_2_mat() | 44 |
| 5.9.2.8 main() | 44 |
| 5.9.2.9 probeDensity() | 44 |
| 5.9.2.10 relativePressure() | 45 |
| 5.9.3 Variable Documentation | 45 |
| 5.9.3.1 h_sq | 45 |
| 5.9.3.2 inv_h | 45 |
| 5.9.3.3 inv_h_5 | 46 |
| 5.9.3.4 inv_h_7 | 46 |
| 5.9.3.5 inv_h_cb | 46 |
| 5.9.3.6 inv_h_sq | 46 |
| 5.10 quantumsph_OMP_Memorize32.c File Reference | 46 |
| 5.10.1 Macro Definition Documentation | 47 |
| 5.10.1.1 INV_SQRT_PI | 47 |
| 5.10.2 Function Documentation | 47 |
| 5.10.2.1 acceleration() | 48 |
| 5.10.2.2 FreeMatrix() | 48 |
| 5.10.2.3 InitializeMatrix() | 48 |
| 5.10.2.4 kernel_0() | 49 |
| 5.10.2.5 kernel_0_mat() | 49 |
| 5.10.2.6 kernel_1_mat() | 50 |
| 5.10.2.7 kernel_2_mat() | 50 |
| 5.10.2.8 main() | 50 |
| 5.10.2.9 probeDensity() | 51 |
| 5.10.2.10 relativePressure() | 51 |
| 5.10.3 Variable Documentation | 51 |
| 5.10.3.1 h_sq | 52 |
| 5.10.3.2 inv_h | 52 |
| 5.10.3.3 inv_h_5 | 52 |
| 5.10.3.4 inv_h_7 | 52 |
| 5.10.3.5 inv_h_cb | 52 |
| 5.10.3.6 inv_h_sq | 52 |
| 5.11 quantumsph_OMP_Memorize64.c File Reference | 52 |
| 5.11.1 Macro Definition Documentation | 53 |
| 5.11.1.1 INV_SQRT_PI | 53 |

| | |
|---|-----------|
| 5.11.2 Function Documentation | 54 |
| 5.11.2.1 acceleration() | 54 |
| 5.11.2.2 FreeMatrix() | 54 |
| 5.11.2.3 InitializeMatrix() | 54 |
| 5.11.2.4 kernel_0() | 55 |
| 5.11.2.5 kernel_0_mat() | 55 |
| 5.11.2.6 kernel_1_mat() | 56 |
| 5.11.2.7 kernel_2_mat() | 56 |
| 5.11.2.8 main() | 56 |
| 5.11.2.9 probeDensity() | 57 |
| 5.11.2.10 relativePressure() | 57 |
| 5.11.3 Variable Documentation | 57 |
| 5.11.3.1 h_sq | 58 |
| 5.11.3.2 inv_h | 58 |
| 5.11.3.3 inv_h_5 | 58 |
| 5.11.3.4 inv_h_7 | 58 |
| 5.11.3.5 inv_h_cb | 58 |
| 5.11.3.6 inv_h_sq | 58 |
| 5.12 quantumsph_OMP_NoMemory.c File Reference | 58 |
| 5.12.1 Macro Definition Documentation | 59 |
| 5.12.1.1 INV_SQRT_PI | 59 |
| 5.12.2 Function Documentation | 59 |
| 5.12.2.1 acceleration() | 60 |
| 5.12.2.2 kernel_0() | 60 |
| 5.12.2.3 kernel_1() | 60 |
| 5.12.2.4 kernel_2() | 61 |
| 5.12.2.5 main() | 61 |
| 5.12.2.6 pressure() | 61 |
| 5.12.2.7 probeDensity() | 62 |
| 5.12.3 Variable Documentation | 62 |
| 5.12.3.1 h_sq | 62 |
| 5.12.3.2 inv_h | 62 |
| 5.12.3.3 inv_h_5 | 62 |
| 5.12.3.4 inv_h_7 | 63 |
| 5.12.3.5 inv_h_cb | 63 |
| 5.12.3.6 inv_h_sq | 63 |
| 5.13 README.md File Reference | 63 |
| Index | 65 |

Chapter 1

DD2356_QuantumSPH

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

| | | |
|----------------------------|-------|-------------------|
| CSVReader | | 7 |
| quantumsph | | 7 |

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

| | |
|---|----|
| arithmeticCost.c | 11 |
| CSVReader.py | 12 |
| quantumsph.py | 12 |
| quantumsph_Improved.c | 13 |
| quantumsph_Initial.c | 17 |
| quantumsph_Memorize.c | 21 |
| quantumsph_MPI.c | 27 |
| quantumsph_OMP_Initial.c | 34 |
| quantumsph_OMP_Memorize16.c | 40 |
| quantumsph_OMP_Memorize32.c | 46 |
| quantumsph_OMP_Memorize64.c | 52 |
| quantumsph_OMP_NoMemory.c | 58 |

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.

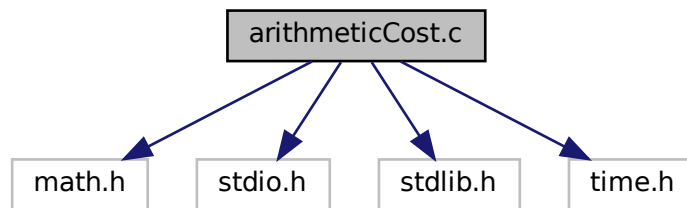
Chapter 5

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](#)

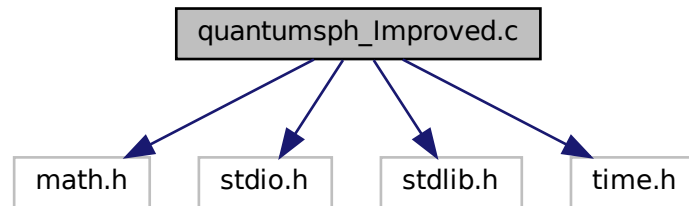
Functions

- 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` 0.56418958354775627928
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`

5.4.1 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

| | |
|-----------|-----------------------------------|
| <i>a</i> | acceleration vector to be updated |
| <i>x</i> | positions of the particles |
| <i>u</i> | velocity vector |
| <i>rP</i> | relative pressure vector |
| <i>n</i> | number of particles |
| <i>b</i> | 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

| | |
|----------|----------------------------|
| <i>r</i> | 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

| | |
|----------|----------------------------|
| <i>r</i> | 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

| | |
|----------|----------------------------|
| <i>r</i> | 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

| | |
|-----------|--|
| <i>rP</i> | pressure vector to be updated, $rP = -(1/4) * (d^2 \rho / dx^2 - (d \rho / dx)^2 / \rho) / (\rho^2)$ |
| <i>x</i> | positions of the particles |
| <i>n</i> | 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

| | |
|------------|--|
| <i>rr</i> | density at specified locations, vector to be updated |
| <i>x</i> | positions of the particles |
| <i>n</i> | number of particles |
| <i>xx</i> | probe locations |
| <i>nxx</i> | 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
```

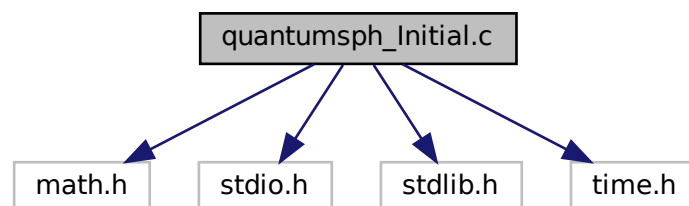
5.4.3.6 inv_h_sq

```
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

| | |
|------------|-----------------------------------|
| <i>a</i> | acceleration vector to be updated |
| <i>x</i> | positions of the particles |
| <i>u</i> | velocity vector |
| <i>rho</i> | density vector |
| <i>P</i> | pressure vector |
| <i>n</i> | number of particles |
| <i>m</i> | SPH particle mass |
| <i>b</i> | damping coefficient |
| <i>h</i> | 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

| | |
|------------|------------------------------|
| <i>rho</i> | density vector to be updated |
| <i>x</i> | positions of the particles |
| <i>n</i> | number of particles |
| <i>m</i> | SPH particle mass |
| <i>h</i> | 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

| | |
|--------------|----------------------------|
| <i>r</i> | distance between particles |
| <i>h</i> | scaling length |
| <i>deriv</i> | 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

| | |
|------------|--|
| <i>P</i> | pressure vector to be updated, $P = -(1/4)*(d^2 \rho / dx^2 - (d \rho / dx)^2 / \rho)$ |
| <i>x</i> | positions of the particles |
| <i>rho</i> | density vector |
| <i>n</i> | number of particles |
| <i>m</i> | SPH particle mass |
| <i>h</i> | 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.

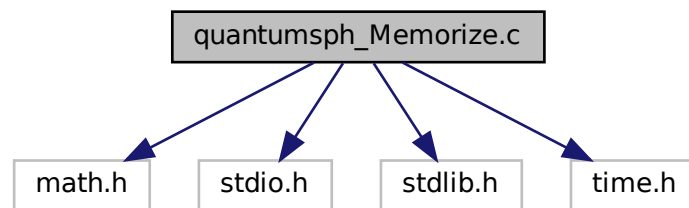
Parameters

| | |
|------------|--|
| <i>rr</i> | density at specified locations, vector to be updated |
| <i>x</i> | positions of the particles |
| <i>n</i> | number of particles |
| <i>m</i> | SPH particle mass |
| <i>h</i> | scaling length |
| <i>xx</i> | probe locations |
| <i>nxx</i> | 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 0.56418958354775627928`
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 `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

| | |
|--------------------------|--|
| <i>a</i> | acceleration vector to be updated |
| <i>x</i> | positions of the particles |
| <i>u</i> | velocity vector |
| <i>rP</i> | relative pressure vector |
| <i>ker_↔_1</i> | pre-computed first derivative of smoothing kernel - interactions between particles |
| <i>n</i> | number of particles |
| <i>b</i> | 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

| | |
|--------------------------|--|
| <i>rho</i> | density vector to be updated |
| <i>ker_↔_0</i> | pre-computed smoothing kernel - interactions between particles |
| <i>n</i> | number of particles |

5.6.2.3 FreeMatrix()

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

Frees memory space allocated for a matrix.

Parameters

| | |
|----------|---|
| <i>A</i> | 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

| | |
|--------------------|-------------------|
| <code>n_row</code> | number of rows |
| <code>n_col</code> | 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

| | |
|----------------|----------------------------|
| <code>r</code> | distance between particles |
| <code>h</code> | 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

| | |
|------------------|--|
| <code>ker</code> | matrix to be updated with the computed weights |
| <code>x</code> | positions of the particles |
| <code>n</code> | 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

| | |
|------------|--|
| <i>ker</i> | matrix to be updated with the computed weights |
| <i>x</i> | positions of the particles |
| <i>n</i> | 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

| | |
|------------|--|
| <i>ker</i> | matrix to be updated with the computed weights |
| <i>x</i> | positions of the particles |
| <i>n</i> | 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

| | |
|--------------------------|--|
| <i>rP</i> | pressure vector to be updated, $rP = -(1/4) * (d^2 \rho / dx^2 - (d \rho / dx)^2 / \rho) / (\rho^2)$ |
| <i>rho</i> | density vector |
| <i>ker</i> _{↔0} | pre-computed smoothing kernel - interactions between particles |
| <i>ker</i> _{↔1} | pre-computed first derivative of smoothing kernel - interactions between particles |
| <i>ker</i> _{↔2} | pre-computed second derivative of smoothing kernel - interactions between particles |
| <i>n</i> | 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

| | |
|------------|--|
| <i>rr</i> | density at specified locations, vector to be updated |
| <i>x</i> | positions of the particles |
| <i>n</i> | number of particles |
| <i>xx</i> | probe locations |
| <i>nxx</i> | 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
```

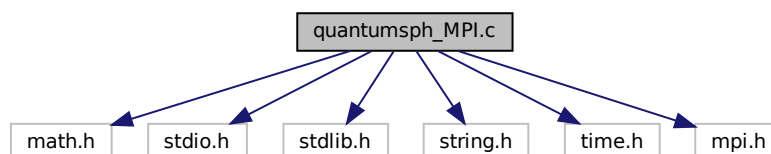
5.6.3.6 inv_h_sq

```
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

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

5.7.2.2 FreeMatrix()

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

Frees memory space allocated for a matrix.

Parameters

| | |
|----------|---|
| <i>A</i> | 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

| | |
|--------------------|-------------------|
| <code>n_row</code> | number of rows |
| <code>n_col</code> | 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

| | |
|----------------|----------------------------|
| <code>r</code> | distance between particles |
| <code>h</code> | 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

| | |
|----------------|----------------------------|
| <code>r</code> | 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

| | |
|----------|----------------------------|
| <i>r</i> | 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

| | |
|--------------|---|
| <i>rr</i> | density at specified locations, vector to be updated |
| <i>x</i> | positions of the particles |
| <i>x_ext</i> | allocated memory space for receiving positions of other processes's particles |
| <i>xx</i> | Generated by Probe locations |
| <i>nxx</i> | 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

| | |
|--------------------------|--|
| <i>rP</i> | pressure vector to be updated, $rP = -(1/4) * (d^2 \rho / dx^2 - (d \rho / dx)^2 / \rho) / (\rho^2)$ |
| <i>ker_←_0</i> | smoothing kernel to be computed - interactions between particles |
| <i>ker_←_1</i> | first derivative of smoothing kernel to be computed - interactions between particles |
| <i>x</i> | positions of the particles |
| <i>x_ext</i> | 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
```

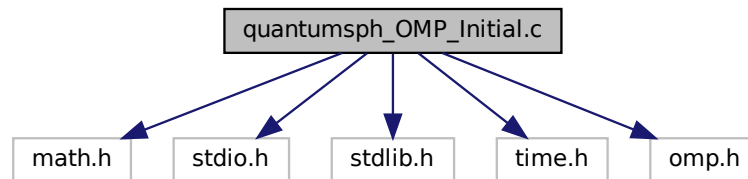
5.7.3.11 size

```
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

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

5.8.2.2 FreeMatrix()

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

Frees memory space allocated for a matrix.

Parameters

| | |
|----------|---|
| <i>A</i> | 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

| | |
|--------------|-------------------|
| <i>n_row</i> | number of rows |
| <i>n_col</i> | 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

| | |
|----------|----------------------------|
| <i>r</i> | 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

| | |
|------------|--|
| <i>ker</i> | matrix to be updated with the computed weights |
| <i>x</i> | positions of the particles |
| <i>n</i> | 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

| | |
|------------|--|
| <i>ker</i> | matrix to be updated with the computed weights |
| <i>x</i> | positions of the particles |
| <i>n</i> | 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

| | |
|------------|--|
| <i>ker</i> | matrix to be updated with the computed weights |
| <i>x</i> | positions of the particles |
| <i>n</i> | 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

| | |
|------------|--|
| <i>rr</i> | density at specified locations, vector to be updated |
| <i>x</i> | positions of the particles |
| <i>n</i> | number of particles |
| <i>xx</i> | probe locations |
| <i>nxx</i> | 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 |
| $ker_{\leftrightarrow 0}$ | pre-computed smoothing kernel - interactions between particles |
| $ker_{\leftrightarrow 1}$ | pre-computed first derivative of smoothing kernel - interactions between particles |
| $ker_{\leftrightarrow 2}$ | 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
```

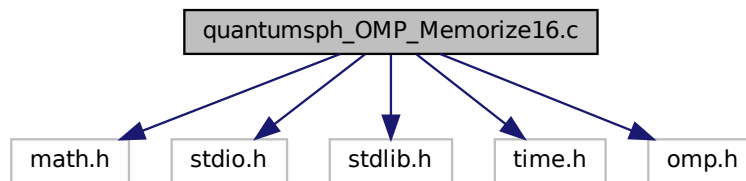
5.8.3.6 inv_h_sq

```
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` 0.56418958354775627928
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`

5.9.1 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_{\leftrightarrow_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

| | |
|----------|----------------------------|
| <i>r</i> | 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

| | |
|------------|--|
| <i>ker</i> | matrix to be updated with the computed weights |
| <i>x</i> | positions of the particles |
| <i>n</i> | 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

| | |
|------------|--|
| <i>ker</i> | matrix to be updated with the computed weights |
| <i>x</i> | positions of the particles |
| <i>n</i> | 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

| | |
|------------|--|
| <i>ker</i> | matrix to be updated with the computed weights |
| <i>x</i> | positions of the particles |
| <i>n</i> | 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

| | |
|------------|--|
| <i>rr</i> | density at specified locations, vector to be updated |
| <i>x</i> | positions of the particles |
| <i>n</i> | number of particles |
| <i>xx</i> | probe locations |
| <i>nxx</i> | 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

| | |
|--------------------------|--|
| <i>rP</i> | pressure vector to be updated, $rP = -(1/4) * (d^2 \rho / dx^2 - (d \rho / dx)^2 / \rho) / (\rho^2)$ |
| <i>x</i> | positions of the particles |
| <i>ker_↔_0</i> | pre-computed smoothing kernel - interactions between particles |
| <i>ker_↔_1</i> | pre-computed first derivative of smoothing kernel - interactions between particles |
| <i>ker_↔_2</i> | pre-computed second derivative of smoothing kernel - interactions between particles |
| <i>n</i> | 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
```

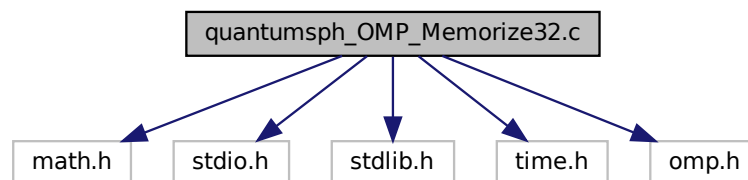
5.9.3.6 inv_h_sq

```
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

| | |
|-------------------------------------|--|
| <i>a</i> | acceleration vector to be updated |
| <i>x</i> | positions of the particles |
| <i>u</i> | velocity vector |
| <i>rP</i> | relative pressure vector |
| <i>ker_↔</i> <i>_1</i> | pre-computed first derivative of smoothing kernel - interactions between particles |
| <i>n</i> | number of particles |
| <i>b</i> | damping coefficient |

5.10.2.2 FreeMatrix()

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

Frees memory space allocated for a matrix.

Parameters

| | |
|----------|---|
| <i>A</i> | 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

| | |
|--------------|-------------------|
| <i>n_row</i> | number of rows |
| <i>n_col</i> | 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

| | |
|----------|----------------------------|
| <i>r</i> | 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

| | |
|------------|--|
| <i>ker</i> | matrix to be updated with the computed weights |
| <i>x</i> | positions of the particles |
| <i>n</i> | 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

| | |
|------------|--|
| <i>ker</i> | matrix to be updated with the computed weights |
| <i>x</i> | positions of the particles |
| <i>n</i> | 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

| | |
|------------|--|
| <i>ker</i> | matrix to be updated with the computed weights |
| <i>x</i> | positions of the particles |
| <i>n</i> | 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

| | |
|------------|--|
| <i>rr</i> | density at specified locations, vector to be updated |
| <i>x</i> | positions of the particles |
| <i>n</i> | number of particles |
| <i>xx</i> | probe locations |
| <i>nxx</i> | 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

| | |
|------------------------------------|--|
| <i>rP</i> | pressure vector to be updated, $rP = -(1/4) * (d^2 \rho / dx^2 - (d \rho / dx)^2 / \rho) / (\rho^2)$ |
| <i>x</i> | positions of the particles |
| <i>ker_↔₀</i> | pre-computed smoothing kernel - interactions between particles |
| <i>ker_↔₁</i> | pre-computed first derivative of smoothing kernel - interactions between particles |
| <i>ker_↔₂</i> | pre-computed second derivative of smoothing kernel - interactions between particles |
| <i>n</i> | 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
```

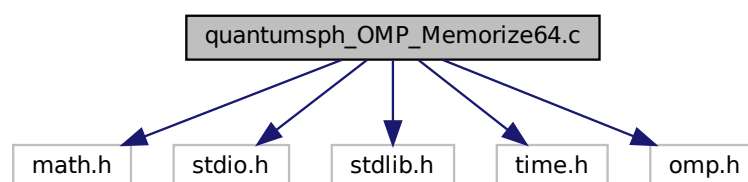
5.10.3.6 inv_h_sq

```
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

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

5.11.2.2 FreeMatrix()

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

Frees memory space allocated for a matrix.

Parameters

| | |
|----------|---|
| <i>A</i> | 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

| | |
|--------------------|-------------------|
| <code>n_row</code> | number of rows |
| <code>n_col</code> | 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

| | |
|----------------|----------------------------|
| <code>r</code> | 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

| | |
|------------------|--|
| <code>ker</code> | matrix to be updated with the computed weights |
| <code>x</code> | positions of the particles |
| <code>n</code> | 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

| | |
|------------|--|
| <i>ker</i> | matrix to be updated with the computed weights |
| <i>x</i> | positions of the particles |
| <i>n</i> | 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

| | |
|------------|--|
| <i>ker</i> | matrix to be updated with the computed weights |
| <i>x</i> | positions of the particles |
| <i>n</i> | 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

| | |
|------------|--|
| <i>rr</i> | density at specified locations, vector to be updated |
| <i>x</i> | positions of the particles |
| <i>n</i> | number of particles |
| <i>xx</i> | probe locations |
| <i>nxx</i> | 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

| | |
|------------------------------------|--|
| <i>rP</i> | pressure vector to be updated, $rP = -(1/4) * (d^2 \rho / dx^2 - (d \rho / dx)^2 / \rho) / (\rho^2)$ |
| <i>x</i> | positions of the particles |
| <i>ker_↔₀</i> | pre-computed smoothing kernel - interactions between particles |
| <i>ker_↔₁</i> | pre-computed first derivative of smoothing kernel - interactions between particles |
| <i>ker_↔₂</i> | pre-computed second derivative of smoothing kernel - interactions between particles |
| <i>n</i> | 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
```

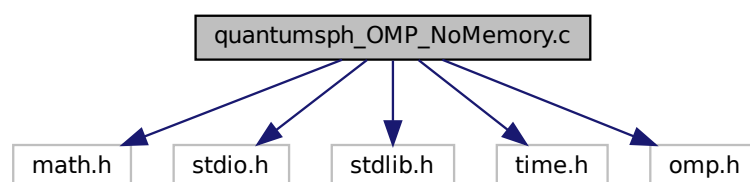
5.11.3.6 inv_h_sq

```
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

| | |
|-----------|-----------------------------------|
| <i>a</i> | acceleration vector to be updated |
| <i>x</i> | positions of the particles |
| <i>u</i> | velocity vector |
| <i>rP</i> | relative pressure vector |
| <i>n</i> | number of particles |
| <i>b</i> | 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

| | |
|----------|----------------------------|
| <i>r</i> | distance between particles |
| <i>h</i> | 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

| | |
|----------|----------------------------|
| <i>r</i> | 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

| | |
|----------|----------------------------|
| <i>r</i> | 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

acceleration

- quantumsph, [8](#)
- quantumsph_Improved.c, [14](#)
- quantumsph_Initial.c, [18](#)
- quantumsph_Memorize.c, [22](#)
- quantumsph_MPI.c, [29](#)
- quantumsph_OMP_Initial.c, [35](#)
- quantumsph_OMP_Memorize16.c, [41](#)
- quantumsph_OMP_Memorize32.c, [47](#)
- quantumsph_OMP_Memorize64.c, [54](#)
- quantumsph_OMP_NoMemory.c, [59](#)

arithmeticCost.c, [11](#)

- main, [12](#)
- N, [11](#)

CSVReader, [7](#)

- read_and_plot, [7](#)

CSVReader.py, [12](#)

density

- quantumsph, [8](#)
- quantumsph_Initial.c, [19](#)
- quantumsph_Memorize.c, [23](#)

FreeMatrix

- quantumsph_Memorize.c, [23](#)
- quantumsph_MPI.c, [29](#)
- quantumsph_OMP_Initial.c, [36](#)
- quantumsph_OMP_Memorize16.c, [42](#)
- quantumsph_OMP_Memorize32.c, [48](#)
- quantumsph_OMP_Memorize64.c, [54](#)

h_sq

- quantumsph_Improved.c, [16](#)
- quantumsph_Memorize.c, [26](#)
- quantumsph_MPI.c, [32](#)
- quantumsph_OMP_Initial.c, [39](#)
- quantumsph_OMP_Memorize16.c, [45](#)
- quantumsph_OMP_Memorize32.c, [51](#)
- quantumsph_OMP_Memorize64.c, [57](#)
- quantumsph_OMP_NoMemory.c, [62](#)

InitializeMatrix

- quantumsph_Memorize.c, [23](#)
- quantumsph_MPI.c, [29](#)
- quantumsph_OMP_Initial.c, [36](#)
- quantumsph_OMP_Memorize16.c, [42](#)
- quantumsph_OMP_Memorize32.c, [48](#)
- quantumsph_OMP_Memorize64.c, [54](#)

inv_h

- quantumsph_Improved.c, [17](#)

- quantumsph_Memorize.c, [27](#)

- quantumsph_MPI.c, [32](#)

- quantumsph_OMP_Initial.c, [39](#)

- quantumsph_OMP_Memorize16.c, [45](#)

- quantumsph_OMP_Memorize32.c, [52](#)

- quantumsph_OMP_Memorize64.c, [58](#)

- quantumsph_OMP_NoMemory.c, [62](#)

inv_h_5

- quantumsph_Improved.c, [17](#)

- quantumsph_Memorize.c, [27](#)

- quantumsph_MPI.c, [32](#)

- quantumsph_OMP_Initial.c, [39](#)

- quantumsph_OMP_Memorize16.c, [45](#)

- quantumsph_OMP_Memorize32.c, [52](#)

- quantumsph_OMP_Memorize64.c, [58](#)

- quantumsph_OMP_NoMemory.c, [62](#)

inv_h_7

- quantumsph_Improved.c, [17](#)

- quantumsph_Memorize.c, [27](#)

- quantumsph_MPI.c, [32](#)

- quantumsph_OMP_Initial.c, [39](#)

- quantumsph_OMP_Memorize16.c, [46](#)

- quantumsph_OMP_Memorize32.c, [52](#)

- quantumsph_OMP_Memorize64.c, [58](#)

- quantumsph_OMP_NoMemory.c, [62](#)

inv_h_cb

- quantumsph_Improved.c, [17](#)

- quantumsph_Memorize.c, [27](#)

- quantumsph_MPI.c, [33](#)

- quantumsph_OMP_Initial.c, [39](#)

- quantumsph_OMP_Memorize16.c, [46](#)

- quantumsph_OMP_Memorize32.c, [52](#)

- quantumsph_OMP_Memorize64.c, [58](#)

- quantumsph_OMP_NoMemory.c, [63](#)

inv_h_sq

- quantumsph_Improved.c, [17](#)

- quantumsph_Memorize.c, [27](#)

- quantumsph_MPI.c, [33](#)

- quantumsph_OMP_Initial.c, [40](#)

- quantumsph_OMP_Memorize16.c, [46](#)

- quantumsph_OMP_Memorize32.c, [52](#)

- quantumsph_OMP_Memorize64.c, [58](#)

- quantumsph_OMP_NoMemory.c, [63](#)

INV_SQRT_PI

- quantumsph_Improved.c, [14](#)

- quantumsph_Memorize.c, [22](#)

- quantumsph_MPI.c, [28](#)

- quantumsph_OMP_Initial.c, [35](#)

- quantumsph_OMP_Memorize16.c, [41](#)

- quantumsph_OMP_Memorize32.c, 47
 - quantumsph_OMP_Memorize64.c, 53
 - quantumsph_OMP_NoMemory.c, 59
- kernel
 - quantumsph, 8
 - quantumsph_Initial.c, 19
- kernel_0
 - quantumsph_Improved.c, 14
 - quantumsph_Memorize.c, 24
 - quantumsph_MPI.c, 30
 - quantumsph_OMP_Initial.c, 36
 - quantumsph_OMP_Memorize16.c, 42
 - quantumsph_OMP_Memorize32.c, 49
 - quantumsph_OMP_Memorize64.c, 55
 - quantumsph_OMP_NoMemory.c, 60
- kernel_0_mat
 - quantumsph_Memorize.c, 24
 - quantumsph_OMP_Initial.c, 37
 - quantumsph_OMP_Memorize16.c, 43
 - quantumsph_OMP_Memorize32.c, 49
 - quantumsph_OMP_Memorize64.c, 55
- kernel_1
 - quantumsph_Improved.c, 15
 - quantumsph_MPI.c, 30
 - quantumsph_OMP_NoMemory.c, 60
- kernel_1_mat
 - quantumsph_Memorize.c, 25
 - quantumsph_OMP_Initial.c, 37
 - quantumsph_OMP_Memorize16.c, 43
 - quantumsph_OMP_Memorize32.c, 50
 - quantumsph_OMP_Memorize64.c, 56
- kernel_2
 - quantumsph_Improved.c, 15
 - quantumsph_MPI.c, 31
 - quantumsph_OMP_NoMemory.c, 61
- kernel_2_mat
 - quantumsph_Memorize.c, 25
 - quantumsph_OMP_Initial.c, 37
 - quantumsph_OMP_Memorize16.c, 44
 - quantumsph_OMP_Memorize32.c, 50
 - quantumsph_OMP_Memorize64.c, 56
- main
 - arithmeticCost.c, 12
 - quantumsph, 9
 - quantumsph_Improved.c, 15
 - quantumsph_Initial.c, 20
 - quantumsph_Memorize.c, 25
 - quantumsph_MPI.c, 31
 - quantumsph_OMP_Initial.c, 38
 - quantumsph_OMP_Memorize16.c, 44
 - quantumsph_OMP_Memorize32.c, 50
 - quantumsph_OMP_Memorize64.c, 56
 - quantumsph_OMP_NoMemory.c, 61
- N
 - arithmeticCost.c, 11
- n
 - quantumsph_MPI.c, 33
- n_loc
 - quantumsph_MPI.c, 33
- PI
 - quantumsph_Initial.c, 18
- pressure
 - quantumsph, 9
 - quantumsph_Improved.c, 16
 - quantumsph_Initial.c, 20
 - quantumsph_Memorize.c, 25
 - quantumsph_OMP_NoMemory.c, 61
- probeDensity
 - quantumsph, 9
 - quantumsph_Improved.c, 16
 - quantumsph_Initial.c, 20
 - quantumsph_Memorize.c, 26
 - quantumsph_MPI.c, 31
 - quantumsph_OMP_Initial.c, 38
 - quantumsph_OMP_Memorize16.c, 44
 - quantumsph_OMP_Memorize32.c, 50
 - quantumsph_OMP_Memorize64.c, 56
 - quantumsph_OMP_NoMemory.c, 62
- quantumsph, 7
 - acceleration, 8
 - density, 8
 - kernel, 8
 - main, 9
 - pressure, 9
 - probeDensity, 9
- quantumsph.py, 12
- quantumsph_Improved.c, 13
 - acceleration, 14
 - h_sq, 16
 - inv_h, 17
 - inv_h_5, 17
 - inv_h_7, 17
 - inv_h_cb, 17
 - inv_h_sq, 17
 - INV_SQRT_PI, 14
 - kernel_0, 14
 - kernel_1, 15
 - kernel_2, 15
 - main, 15
 - pressure, 16
 - probeDensity, 16
- quantumsph_Initial.c, 17
 - acceleration, 18
 - density, 19
 - kernel, 19
 - main, 20
 - PI, 18
 - pressure, 20
 - probeDensity, 20
- quantumsph_Memorize.c, 21
 - acceleration, 22
 - density, 23
 - FreeMatrix, 23

- h_sq, 26
- InitializeMatrix, 23
- inv_h, 27
- inv_h_5, 27
- inv_h_7, 27
- inv_h_cb, 27
- inv_h_sq, 27
- INV_SQRT_PI, 22
- kernel_0, 24
- kernel_0_mat, 24
- kernel_1_mat, 25
- kernel_2_mat, 25
- main, 25
- pressure, 25
- probeDensity, 26
- quantumsph_MPI.c, 27
 - acceleration, 29
 - FreeMatrix, 29
 - h_sq, 32
 - InitializeMatrix, 29
 - inv_h, 32
 - inv_h_5, 32
 - inv_h_7, 32
 - inv_h_cb, 33
 - inv_h_sq, 33
 - INV_SQRT_PI, 28
 - kernel_0, 30
 - kernel_1, 30
 - kernel_2, 31
 - main, 31
 - n, 33
 - n_loc, 33
 - probeDensity, 31
 - rank, 33
 - relativePressure, 32
 - rem, 33
 - size, 33
- quantumsph_OMP_Initial.c, 34
 - acceleration, 35
 - FreeMatrix, 36
 - h_sq, 39
 - InitializeMatrix, 36
 - inv_h, 39
 - inv_h_5, 39
 - inv_h_7, 39
 - inv_h_cb, 39
 - inv_h_sq, 40
 - INV_SQRT_PI, 35
 - kernel_0, 36
 - kernel_0_mat, 37
 - kernel_1_mat, 37
 - kernel_2_mat, 37
 - main, 38
 - probeDensity, 38
 - relativePressure, 38
- quantumsph_OMP_Memorize16.c, 40
 - acceleration, 41
 - FreeMatrix, 42
 - h_sq, 45
 - InitializeMatrix, 42
 - inv_h, 45
 - inv_h_5, 45
 - inv_h_7, 46
 - inv_h_cb, 46
 - inv_h_sq, 46
 - INV_SQRT_PI, 41
 - kernel_0, 42
 - kernel_0_mat, 43
 - kernel_1_mat, 43
 - kernel_2_mat, 44
 - main, 44
 - probeDensity, 44
 - relativePressure, 45
- quantumsph_OMP_Memorize32.c, 46
 - acceleration, 47
 - FreeMatrix, 48
 - h_sq, 51
 - InitializeMatrix, 48
 - inv_h, 52
 - inv_h_5, 52
 - inv_h_7, 52
 - inv_h_cb, 52
 - inv_h_sq, 52
 - INV_SQRT_PI, 47
 - kernel_0, 49
 - kernel_0_mat, 49
 - kernel_1_mat, 50
 - kernel_2_mat, 50
 - main, 50
 - probeDensity, 50
 - relativePressure, 51
- quantumsph_OMP_Memorize64.c, 52
 - acceleration, 54
 - FreeMatrix, 54
 - h_sq, 57
 - InitializeMatrix, 54
 - inv_h, 58
 - inv_h_5, 58
 - inv_h_7, 58
 - inv_h_cb, 58
 - inv_h_sq, 58
 - INV_SQRT_PI, 53
 - kernel_0, 55
 - kernel_0_mat, 55
 - kernel_1_mat, 56
 - kernel_2_mat, 56
 - main, 56
 - probeDensity, 56
 - relativePressure, 57
- quantumsph_OMP_NoMemory.c, 58
 - acceleration, 59
 - h_sq, 62
 - inv_h, 62
 - inv_h_5, 62
 - inv_h_7, 62
 - inv_h_cb, 63

- inv_h_sq, [63](#)
- INV_SQRT_PI, [59](#)
- kernel_0, [60](#)
- kernel_1, [60](#)
- kernel_2, [61](#)
- main, [61](#)
- pressure, [61](#)
- probeDensity, [62](#)

rank

- quantumsph_MPI.c, [33](#)

read_and_plot

- CSVReader, [7](#)

README.md, [63](#)

relativePressure

- quantumsph_MPI.c, [32](#)
- quantumsph_OMP_Initial.c, [38](#)
- quantumsph_OMP_Memorize16.c, [45](#)
- quantumsph_OMP_Memorize32.c, [51](#)
- quantumsph_OMP_Memorize64.c, [57](#)

rem

- quantumsph_MPI.c, [33](#)

size

- quantumsph_MPI.c, [33](#)