

phasefield-accelerator-benchmarks
pre-alpha

Generated by Doxygen 1.8.13

Contents

1	Module Index	1
1.1	Modules	1
2	Class Index	2
2.1	Class List	2
3	File Index	2
3.1	File List	2
4	Module Documentation	4
4.1	Diffusion equation benchmark	4
4.1.1	Detailed Description	5
4.1.2	Typedef Documentation	5
4.1.3	Function Documentation	6
4.1.4	Variable Documentation	9
4.2	Analytic	10
4.2.1	Detailed Description	10
4.2.2	Function Documentation	10
4.3	Openmp	11
4.3.1	Detailed Description	11
4.3.2	Function Documentation	11
4.4	Serial	13
4.4.1	Detailed Description	13
4.4.2	Function Documentation	13
4.5	Tbb	15
4.5.1	Detailed Description	15
4.5.2	Function Documentation	15
4.6	Cuda	18
4.6.1	Detailed Description	19
4.6.2	Function Documentation	19
4.7	Openacc	21
4.7.1	Detailed Description	21
4.7.2	Function Documentation	21
4.8	Opencl	23
4.8.1	Detailed Description	23
4.8.2	Function Documentation	23

5	Class Documentation	26
5.1	ResidualSumOfSquares2D Class Reference	26
5.1.1	Detailed Description	26
6	File Documentation	26
6.1	boundaries.c File Reference	26
6.1.1	Detailed Description	27
6.2	boundaries.c File Reference	27
6.2.1	Detailed Description	28
6.3	boundaries.c File Reference	28
6.3.1	Detailed Description	28
6.4	boundaries.c File Reference	29
6.4.1	Detailed Description	29
6.5	boundaries.cpp File Reference	29
6.5.1	Detailed Description	30
6.6	boundaries.h File Reference	30
6.6.1	Detailed Description	31
6.7	boundaries.h File Reference	31
6.7.1	Detailed Description	32
6.8	boundaries.h File Reference	32
6.8.1	Detailed Description	33
6.9	boundaries.h File Reference	33
6.9.1	Detailed Description	34
6.10	boundaries.h File Reference	34
6.10.1	Detailed Description	35
6.11	discretization.c File Reference	35
6.11.1	Detailed Description	36
6.12	discretization.c File Reference	36
6.12.1	Detailed Description	37
6.13	discretization.c File Reference	37
6.13.1	Detailed Description	37

6.14	discretization.c File Reference	38
6.14.1	Detailed Description	38
6.15	discretization.cl File Reference	38
6.15.1	Detailed Description	39
6.16	discretization.cpp File Reference	40
6.16.1	Detailed Description	40
6.17	discretization.cu File Reference	41
6.17.1	Detailed Description	42
6.18	discretization.h File Reference	42
6.18.1	Detailed Description	43
6.19	discretization.h File Reference	43
6.19.1	Detailed Description	44
6.20	discretization.h File Reference	44
6.20.1	Detailed Description	45
6.21	discretization.h File Reference	45
6.21.1	Detailed Description	46
6.22	discretization.h File Reference	46
6.22.1	Detailed Description	47
6.23	discretization.h File Reference	47
6.23.1	Detailed Description	48
6.24	discretization.h File Reference	48
6.24.1	Detailed Description	49
6.25	main.c File Reference	49
6.25.1	Detailed Description	49
6.26	main.c File Reference	49
6.26.1	Detailed Description	50
6.27	mesh.c File Reference	50
6.27.1	Detailed Description	51
6.28	mesh.h File Reference	51
6.28.1	Detailed Description	52

6.29	numerics.c File Reference	52
6.29.1	Detailed Description	53
6.30	numerics.h File Reference	53
6.30.1	Detailed Description	54
6.31	output.c File Reference	54
6.31.1	Detailed Description	55
6.32	output.h File Reference	55
6.32.1	Detailed Description	56
6.33	timer.c File Reference	56
6.33.1	Detailed Description	57
6.34	timer.h File Reference	57
6.34.1	Detailed Description	58
6.35	type.h File Reference	58
6.35.1	Detailed Description	58
Index		59

1 Module Index

1.1 Modules

Here is a list of all modules:

Diffusion equation benchmark	4
Analytic	10
Openmp	11
Serial	13
Tbb	15
Cuda	18
Openacc	21
Opencl	23

2 Class Index

2.1 Class List

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

ResidualSumOfSquares2D	
Comparison algorithm for execution on the block of threads	26

3 File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

cpu-openmp-diffusion/boundaries.c	
Implementation of boundary condition functions with OpenMP threading	26
cpu-serial-diffusion/boundaries.c	
Implementation of boundary condition functions without threading	27
gpu-cuda-diffusion/boundaries.c	
Implementation of boundary condition functions with OpenMP threading	28
gpu-openacc-diffusion/boundaries.c	
Implementation of boundary condition functions with OpenMP threading	29
gpu-opencl-diffusion/boundaries.c	??
boundaries.cpp	
Implementation of boundary condition functions with TBB threading	29
cpu-openmp-diffusion/boundaries.h	
Declaration of boundary condition function prototypes for OpenMP benchmarks	30
cpu-serial-diffusion/boundaries.h	
Declaration of boundary condition function prototypes for CPU benchmarks	31
cpu-tbb-diffusion/boundaries.h	
Declaration of boundary condition function prototypes for TBB benchmarks	32
gpu-cuda-diffusion/boundaries.h	
Declaration of boundary condition function prototypes for CUDA benchmarks	33
gpu-openacc-diffusion/boundaries.h	
Declaration of boundary condition function prototypes for OpenAcc benchmarks	34
gpu-opencl-diffusion/boundaries.h	??
cpu-analytic-diffusion/discretization.c	
Implementation of analytical solution functions	35
cpu-openmp-diffusion/discretization.c	
Implementation of boundary condition functions with OpenMP threading	36

cpu-serial-diffusion/discretization.c	
Implementation of boundary condition functions without threading	37
gpu-openacc-diffusion/discretization.c	
Implementation of boundary condition functions with OpenACC threading	38
discretization.cl	
Implementation of boundary condition functions with OpenCL acceleration	38
discretization.cpp	
Implementation of boundary condition functions with TBB threading	40
discretization.cu	
Implementation of boundary condition functions with CUDA acceleration	41
cpu-analytic-diffusion/discretization.h	
Declaration of analytical solution prototypes	42
cpu-omp-diffusion/discretization.h	
Declaration of discretized mathematical function prototypes for OpenMP benchmarks	43
cpu-serial-diffusion/discretization.h	
Declaration of discretized mathematical function prototypes for CPU benchmarks	44
cpu-tbb-diffusion/discretization.h	
Declaration of discretized mathematical function prototypes for TBB benchmarks	45
gpu-cuda-diffusion/discretization.h	
Declaration of discretized mathematical function prototypes for CUDA benchmarks	46
gpu-openacc-diffusion/discretization.h	
Declaration of discretized mathematical function prototypes for OpenAcc benchmarks	47
gpu-opencl-diffusion/discretization.h	
Declaration of discretized mathematical function prototypes for OpenCL benchmarks	48
common-diffusion/main.c	
Implementation of semi-infinite diffusion equation	49
cpu-analytic-diffusion/main.c	
Analytical solution to semi-infinite diffusion equation	49
mesh.c	
Implementation of mesh handling functions for diffusion benchmarks	50
mesh.h	
Declaration of mesh function prototypes for diffusion benchmarks	51
numerics.c	
Implementation of boundary condition functions with OpenMP threading	52
numerics.h	
Declaration of Laplacian operator and analytical solution functions	53
output.c	
Implementation of file output functions for diffusion benchmarks	54
output.h	
Declaration of output function prototypes for diffusion benchmarks	55
timer.c	
High-resolution cross-platform machine time reader	56

timer.h	Declaration of timer function prototypes for diffusion benchmarks	57
type.h	Definition of scalar data type and Doxygen diffusion group	58

4 Module Documentation

4.1 Diffusion equation benchmark

Files

- file [common-diffusion/main.c](#)
Implementation of semi-infinite diffusion equation.
- file [mesh.c](#)
Implementatation of mesh handling functions for diffusion benchmarks.
- file [mesh.h](#)
Declaration of mesh function prototypes for diffusion benchmarks.
- file [numerics.c](#)
Implementation of boundary condition functions with OpenMP threading.
- file [numerics.h](#)
Declaration of Laplacian operator and analytical solution functions.
- file [output.c](#)
Implementation of file output functions for diffusion benchmarks.
- file [output.h](#)
Declaration of output function prototypes for diffusion benchmarks.
- file [timer.c](#)
High-resolution cross-platform machine time reader.
- file [timer.h](#)
Declaration of timer function prototypes for diffusion benchmarks.
- file [type.h](#)
Definition of scalar data type and Doxygen diffusion group.

Macros

- `#define MAX_MASK_W 5`
Maximum width of the convolution mask (Laplacian stencil) array.
- `#define MAX_MASK_H 5`
Maximum height of the convolution mask (Laplacian stencil) array.

Typedefs

- `typedef double fp_t`

Functions

- int `main` (int argc, char *argv[])
Run simulation using input parameters specified on the command line.
- void `make_arrays` (fp_t ***conc_old, fp_t ***conc_new, fp_t ***conc_lap, fp_t ***mask_lap, int nx, int ny, int nm)
Allocate 2D arrays to store scalar composition values.
- void `free_arrays` (fp_t **conc_old, fp_t **conc_new, fp_t **conc_lap, fp_t **mask_lap)
Free dynamically allocated memory.
- void `swap_pointers` (fp_t ***conc_old, fp_t ***conc_new)
Swap pointers to data underlying two arrays.
- void `set_mask` (fp_t dx, fp_t dy, int code, fp_t **mask_lap, int nm)
Specify which stencil (mask) to use for the Laplacian (convolution)
- void `five_point_Laplacian_stencil` (fp_t dx, fp_t dy, fp_t **mask_lap, int nm)
Write 5-point Laplacian stencil into convolution mask.
- void `nine_point_Laplacian_stencil` (fp_t dx, fp_t dy, fp_t **mask_lap, int nm)
Write 9-point Laplacian stencil into convolution mask.
- void `slow_nine_point_Laplacian_stencil` (fp_t dx, fp_t dy, fp_t **mask_lap, int nm)
Write 9-point Laplacian stencil into convolution mask.
- fp_t `euclidean_distance` (fp_t ax, fp_t ay, fp_t bx, fp_t by)
Compute Euclidean distance between two points, a and b.
- fp_t `manhattan_distance` (fp_t ax, fp_t ay, fp_t bx, fp_t by)
Compute Manhattan distance between two points, a and b.
- fp_t `distance_point_to_segment` (fp_t ax, fp_t ay, fp_t bx, fp_t by, fp_t px, fp_t py)
Compute minimum distance from point p to a line segment bounded by points a and b.
- void `analytical_value` (fp_t x, fp_t t, fp_t D, fp_t bc[2][2], fp_t *c)
Analytical solution of the diffusion equation for a carburizing process.
- void `print_progress` (const int step, const int steps)
Prints timestamps and a 20-point progress bar to stdout.
- void `write_csv` (fp_t **conc, int nx, int ny, fp_t dx, fp_t dy, int step)
Writes scalar composition field to diffusion.???????.csv.
- void `write_png` (fp_t **conc, int nx, int ny, int step)
Writes scalar composition field to diffusion.???????.png.
- void `StartTimer` ()
Set CPU frequency and begin timing.
- double `GetTimer` ()
Return elapsed time in seconds.

Variables

- struct timeval `timerStart`

4.1.1 Detailed Description

4.1.2 Typedef Documentation

4.1.2.1 fp_t

```
typedef double fp_t
```

Specify the basic data type to achieve the desired accuracy in floating-point arithmetic: float for single-precision, double for double-precision. This choice propagates throughout the code, and may significantly affect runtime on GPU hardware.

Definition at line 40 of file type.h.

4.1.3 Function Documentation

4.1.3.1 analytical_value()

```
void analytical_value (
    fp_t x,
    fp_t t,
    fp_t D,
    fp_t bc[2][2],
    fp_t * c )
```

Analytical solution of the diffusion equation for a carburizing process.

For 1D diffusion through a semi-infinite domain with initial and far-field composition c_∞ and boundary value $c(x = 0, t) = c_0$ with constant diffusivity D , the solution to Fick's second law is

$$c(x, t) = c_0 - (c_0 - c_\infty) \operatorname{erf} \left(\frac{x}{\sqrt{4Dt}} \right)$$

which reduces, when $c_\infty = 0$, to

$$c(x, t) = c_0 \left[1 - \operatorname{erf} \left(\frac{x}{\sqrt{4Dt}} \right) \right].$$

Definition at line 183 of file numerics.c.

4.1.3.2 distance_point_to_segment()

```
fp_t distance_point_to_segment (
    fp_t ax,
    fp_t ay,
    fp_t bx,
    fp_t by,
    fp_t px,
    fp_t py )
```

Compute minimum distance from point p to a line segment bounded by points a and b .

This function computes the projection of p onto ab , limiting the projected range to $[0, 1]$ to handle projections that fall outside of ab . Implemented after Grumdrig on Stackoverflow, <https://stackoverflow.com/a/1501725>.

Definition at line 160 of file numerics.c.

4.1.3.3 five_point_Laplacian_stencil()

```
void five_point_Laplacian_stencil (
    fp_t dx,
    fp_t dy,
    fp_t ** mask_lap,
    int nm )
```

Write 5-point Laplacian stencil into convolution mask.

3×3 mask, 5 values, truncation error $\mathcal{O}(\Delta x^2)$

Definition at line 75 of file numerics.c.

4.1.3.4 main()

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

Run simulation using input parameters specified on the command line.

Program will write a series of PNG image files to visualize scalar composition field, plus a final CSV raw data file and CSV runtime log tabulating the iteration counter (iter), elapsed simulation time (sim_time), system free energy (energy), error relative to analytical solution (wrss), time spent performing convolution (conv_time), time spent updating fields (step_time), time spent writing to disk (IO_time), time spent generating analytical values (soln_time), and total elapsed (run_time).

Definition at line 52 of file common-diffusion/main.c.

4.1.3.5 make_arrays()

```
void make_arrays (
    fp_t *** conc_old,
    fp_t *** conc_new,
    fp_t *** conc_lap,
    fp_t *** mask_lap,
    int nx,
    int ny,
    int nm )
```

Allocate 2D arrays to store scalar composition values.

Arrays are allocated as 1D arrays, then 2D pointer arrays are mapped over the top. This facilitates use of either 1D or 2D data access, depending on whether the task is spatially dependent or not.

Definition at line 41 of file mesh.c.

4.1.3.6 `nine_point_Laplacian_stencil()`

```
void nine_point_Laplacian_stencil (
    fp_t dx,
    fp_t dy,
    fp_t ** mask_lap,
    int nm )
```

Write 9-point Laplacian stencil into convolution mask.

3×3 mask, 9 values, truncation error $\mathcal{O}(\Delta x^4)$

Definition at line 91 of file numerics.c.

4.1.3.7 `print_progress()`

```
void print_progress (
    const int step,
    const int steps )
```

Prints timestamps and a 20-point progress bar to stdout.

Call inside the timestepping loop, near the top, e.g.

```
for (int step=0; step<steps; step++) {
    print_progress(step, steps);
    take_a_step();
    elapsed += dt;
}
```

Definition at line 49 of file output.c.

4.1.3.8 `set_mask()`

```
void set_mask (
    fp_t dx,
    fp_t dy,
    int code,
    fp_t ** mask_lap,
    int nm )
```

Specify which stencil (mask) to use for the Laplacian (convolution)

The mask corresponding to the numerical code will be applied. The suggested encoding is mask width as the ones digit and value count as the tens digit, e.g. five-point Laplacian is 53, nine-point is 93.

To add your own mask (stencil), define its prototype in `numerics.h`, implement it in `numerics.c`, add a case to this function with your chosen numerical encoding, then specify that code in `params.txt`. Note that, for a Laplacian stencil, the sum of the coefficients must equal zero.

If your stencil is larger than 5×5 , you must increase the values defined by `MAX_MASK_W` and `MAX_MASK_H` in `numerics.h`.

Definition at line 50 of file numerics.c.

4.1.3.9 slow_nine_point_Laplacian_stencil()

```
void slow_nine_point_Laplacian_stencil (
    fp_t dx,
    fp_t dy,
    fp_t ** mask_lap,
    int nm )
```

Write 9-point Laplacian stencil into convolution mask.

5×5 mask, 9 values, truncation error $\mathcal{O}(\Delta x^4)$

Provided for testing and demonstration of scalability, only: as the name indicates, this 9-point stencil is computationally more expensive than the 3×3 version. If your code requires $\mathcal{O}(\Delta x^4)$ accuracy, please use [nine_point_Laplacian_stencil\(\)](#).

Definition at line 118 of file numerics.c.

4.1.3.10 swap_pointers()

```
void swap_pointers (
    fp_t *** conc_old,
    fp_t *** conc_new )
```

Swap pointers to data underlying two arrays.

Rather than copy data from conc_old into conc_new, an expensive operation, simply trade the top-most pointers. New becomes old with no data lost and in almost no time.

Definition at line 95 of file mesh.c.

4.1.4 Variable Documentation

4.1.4.1 timerStart

```
struct timeval timerStart
```

Platform-dependent data type of hardware time value

Definition at line 49 of file timer.c.

4.2 Analytic

Files

- file [cpu-analytic-diffusion/discretization.c](#)
Implementation of analytical solution functions.
- file [cpu-analytic-diffusion/discretization.h](#)
Declaration of analytical solution prototypes.
- file [cpu-analytic-diffusion/main.c](#)
Analytical solution to semi-infinite diffusion equation.

Functions

- void [solve_diffusion_equation](#) ([fp_t](#) **conc_old, [fp_t](#) **conc_new, [fp_t](#) **conc_lap, int nx, int ny, [fp_t](#) dx, [fp_t](#) dy, int nm, [fp_t](#) D, [fp_t](#) dt, [fp_t](#) elapsed)
Update the scalar composition field using analytical solution.
- int [main](#) (int argc, char *argv[])
Run simulation using input parameters specified on the command line.

4.2.1 Detailed Description

4.2.2 Function Documentation

4.2.2.1 main()

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

Run simulation using input parameters specified on the command line.

Program will write a series of PNG image files to visualize the scalar composition field, plus `delta.png` showing the difference between the analytical result and the image stored in `../common-diffusion/diffusion.10000.png`.

Definition at line 48 of file `cpu-analytic-diffusion/main.c`.

4.3 Openmp

Files

- file [cpu-openmp-diffusion/boundaries.c](#)
Implementation of boundary condition functions with OpenMP threading.
- file [cpu-openmp-diffusion/boundaries.h](#)
Declaration of boundary condition function prototypes for OpenMP benchmarks.
- file [cpu-openmp-diffusion/discretization.c](#)
Implementation of boundary condition functions with OpenMP threading.
- file [cpu-openmp-diffusion/discretization.h](#)
Declaration of discretized mathematical function prototypes for OpenMP benchmarks.

Functions

- void [set_boundaries](#) ([fp_t](#) bc[2][2])
Set values to be used along the simulation domain boundaries.
- void [apply_initial_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Initialize flat composition field with fixed boundary conditions.
- void [apply_boundary_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Set fixed value (c_{hi}) along left and bottom, zero-flux elsewhere.
- void [set_threads](#) (int n)
Set number of OpenMP threads to use in parallel code sections.
- void [compute_convolution](#) ([fp_t](#) **conc_old, [fp_t](#) **conc_lap, [fp_t](#) **mask_lap, int nx, int ny, int nm)
Perform the convolution of the mask matrix with the composition matrix.
- void [solve_diffusion_equation](#) ([fp_t](#) **conc_old, [fp_t](#) **conc_new, [fp_t](#) **conc_lap, int nx, int ny, int nm, [fp_t](#) D, [fp_t](#) dt, [fp_t](#) *elapsed)
Update the scalar composition field using old and Laplacian values.
- void [check_solution](#) ([fp_t](#) **conc_new, int nx, int ny, [fp_t](#) dx, [fp_t](#) dy, int nm, [fp_t](#) elapsed, [fp_t](#) D, [fp_t](#) bc[2][2], [fp_t](#) *rss)
Compare numerical and analytical solutions of the diffusion equation.

4.3.1 Detailed Description

4.3.2 Function Documentation

4.3.2.1 [apply_initial_conditions\(\)](#)

```
void apply_initial_conditions (
    fp\_t ** conc,
    int nx,
    int ny,
    int nm,
    fp\_t bc[2][2] )
```

Initialize flat composition field with fixed boundary conditions.

The boundary conditions are fixed values of c_{hi} along the lower-left half and upper-right half walls, no flux everywhere else, with an initial values of c_{lo} everywhere. These conditions represent a carburizing process, with partial exposure (rather than the entire left and right walls) to produce an inhomogeneous workload and highlight numerical errors at the boundaries.

Definition at line 56 of file [cpu-openmp-diffusion/boundaries.c](#).

4.3.2.2 check_solution()

```
void check_solution (
    fp_t ** conc_new,
    int nx,
    int ny,
    fp_t dx,
    fp_t dy,
    int nm,
    fp_t elapsed,
    fp_t D,
    fp_t bc[2][2],
    fp_t * rss )
```

Compare numerical and analytical solutions of the diffusion equation.

Returns the residual sum of squares (RSS), normalized to the domain size.

Definition at line 98 of file `cpu-openmp-diffusion/discretization.c`.

4.3.2.3 compute_convolution()

```
void compute_convolution (
    fp_t ** conc_old,
    fp_t ** conc_lap,
    fp_t ** mask_lap,
    int nx,
    int ny,
    int nm )
```

Perform the convolution of the mask matrix with the composition matrix.

If the convolution mask is the Laplacian stencil, the convolution evaluates the discrete Laplacian of the composition field. Other masks are possible, for example the Sobel filters for edge detection. This function is general purpose: as long as the dimensions `nx`, `ny`, and `nm` are properly specified, the convolution will be correctly computed.

Definition at line 51 of file `cpu-openmp-diffusion/discretization.c`.

4.3.2.4 set_boundaries()

```
void set_boundaries (
    fp_t bc[2][2] )
```

Set values to be used along the simulation domain boundaries.

Indexing is row-major, i.e. $A[y][x]$, so $bc = [[y_{lo}, y_{hi}], [x_{lo}, x_{hi}]]$.

Definition at line 38 of file `cpu-openmp-diffusion/boundaries.c`.

4.4 Serial

Files

- file [cpu-serial-diffusion/boundaries.c](#)
Implementation of boundary condition functions without threading.
- file [cpu-serial-diffusion/boundaries.h](#)
Declaration of boundary condition function prototypes for CPU benchmarks.
- file [cpu-serial-diffusion/discretization.c](#)
Implementation of boundary condition functions without threading.
- file [cpu-serial-diffusion/discretization.h](#)
Declaration of discretized mathematical function prototypes for CPU benchmarks.

Functions

- void [set_boundaries](#) ([fp_t](#) bc[2][2])
Set values to be used along the simulation domain boundaries.
- void [apply_initial_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Initialize flat composition field with fixed boundary conditions.
- void [apply_boundary_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Set fixed value (c_{hi}) along left and bottom, zero-flux elsewhere.
- void [set_threads](#) (int n)
Set number of OpenMP threads to use in parallel code sections.
- void [compute_convolution](#) ([fp_t](#) **conc_old, [fp_t](#) **conc_lap, [fp_t](#) **mask_lap, int nx, int ny, int nm)
Perform the convolution of the mask matrix with the composition matrix.
- void [solve_diffusion_equation](#) ([fp_t](#) **conc_old, [fp_t](#) **conc_new, [fp_t](#) **conc_lap, int nx, int ny, int nm, [fp_t](#) D, [fp_t](#) dt, [fp_t](#) *elapsed)
Update the scalar composition field using old and Laplacian values.
- void [check_solution](#) ([fp_t](#) **conc_new, int nx, int ny, [fp_t](#) dx, [fp_t](#) dy, int nm, [fp_t](#) elapsed, [fp_t](#) D, [fp_t](#) bc[2][2], [fp_t](#) *rss)
Compare numerical and analytical solutions of the diffusion equation.

4.4.1 Detailed Description

4.4.2 Function Documentation

4.4.2.1 [apply_initial_conditions\(\)](#)

```
void apply_initial_conditions (
    fp\_t ** conc,
    int nx,
    int ny,
    int nm,
    fp\_t bc[2][2] )
```

Initialize flat composition field with fixed boundary conditions.

The boundary conditions are fixed values of c_{hi} along the lower-left half and upper-right half walls, no flux everywhere else, with an initial values of c_{lo} everywhere. These conditions represent a carburizing process, with partial exposure (rather than the entire left and right walls) to produce an inhomogeneous workload and highlight numerical errors at the boundaries.

Definition at line 55 of file [cpu-serial-diffusion/boundaries.c](#).

4.4.2.2 check_solution()

```
void check_solution (
    fp_t ** conc_new,
    int nx,
    int ny,
    fp_t dx,
    fp_t dy,
    int nm,
    fp_t elapsed,
    fp_t D,
    fp_t bc[2][2],
    fp_t * rss )
```

Compare numerical and analytical solutions of the diffusion equation.

Returns

Residual sum of squares (RSS), normalized to the domain size.

Definition at line 90 of file `cpu-serial-diffusion/discretization.c`.

4.4.2.3 compute_convolution()

```
void compute_convolution (
    fp_t ** conc_old,
    fp_t ** conc_lap,
    fp_t ** mask_lap,
    int nx,
    int ny,
    int nm )
```

Perform the convolution of the mask matrix with the composition matrix.

If the convolution mask is the Laplacian stencil, the convolution evaluates the discrete Laplacian of the composition field. Other masks are possible, for example the Sobel filters for edge detection. This function is general purpose: as long as the dimensions `nx`, `ny`, and `nm` are properly specified, the convolution will be correctly computed.

Definition at line 52 of file `cpu-serial-diffusion/discretization.c`.

4.4.2.4 set_boundaries()

```
void set_boundaries (
    fp_t bc[2][2] )
```

Set values to be used along the simulation domain boundaries.

Indexing is row-major, i.e. $A[y][x]$, so `bc` = $[[y_{lo}, y_{hi}], [x_{lo}, x_{hi}]]$.

Definition at line 37 of file `cpu-serial-diffusion/boundaries.c`.

4.4.2.5 set_threads()

```
void set_threads (
    int n )
```

Set number of OpenMP threads to use in parallel code sections.

Warning

Serial code contains no parallel sections: this setting has no effect.

Definition at line 38 of file `cpu-serial-diffusion/discretization.c`.

4.5 Tbb

Files

- file [boundaries.cpp](#)
Implementation of boundary condition functions with TBB threading.
- file [cpu-tbb-diffusion/boundaries.h](#)
Declaration of boundary condition function prototypes for TBB benchmarks.
- file [discretization.cpp](#)
Implementation of boundary condition functions with TBB threading.
- file [cpu-tbb-diffusion/discretization.h](#)
Declaration of discretized mathematical function prototypes for TBB benchmarks.

Classes

- class [ResidualSumOfSquares2D](#)
Comparison algorithm for execution on the block of threads.

Functions

- void [set_boundaries](#) ([fp_t](#) bc[2][2])
Set values to be used along the simulation domain boundaries.
- void [apply_initial_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Initialize flat composition field with fixed boundary conditions.
- void [apply_boundary_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Set fixed value (c_{hi}) along left and bottom, zero-flux elsewhere.
- void [set_threads](#) (int n)
Requested number of TBB threads to use in parallel code sections.
- void [compute_convolution](#) ([fp_t](#) **conc_old, [fp_t](#) **conc_lap, [fp_t](#) **mask_lap, int nx, int ny, int nm)
Perform the convolution of the mask matrix with the composition matrix.
- void [solve_diffusion_equation](#) ([fp_t](#) **conc_old, [fp_t](#) **B, [fp_t](#) **conc_lap, int nx, int ny, int nm, [fp_t](#) D, [fp_t](#) dt, [fp_t](#) *elapsed)
Update the scalar composition field using old and Laplacian values.
- void [check_solution](#) ([fp_t](#) **conc_new, int nx, int ny, [fp_t](#) dx, [fp_t](#) dy, int nm, [fp_t](#) elapsed, [fp_t](#) D, [fp_t](#) bc[2][2], [fp_t](#) *rss)
Compare numerical and analytical solutions of the diffusion equation.

4.5.1 Detailed Description

4.5.2 Function Documentation

4.5.2.1 apply_initial_conditions()

```
void apply_initial_conditions (
    fp_t ** conc,
    int nx,
    int ny,
    int nm,
    fp_t bc[2][2] )
```

Initialize flat composition field with fixed boundary conditions.

The boundary conditions are fixed values of c_{hi} along the lower-left half and upper-right half walls, no flux everywhere else, with an initial values of c_{lo} everywhere. These conditions represent a carburizing process, with partial exposure (rather than the entire left and right walls) to produce an inhomogeneous workload and highlight numerical errors at the boundaries.

Definition at line 59 of file boundaries.cpp.

4.5.2.2 check_solution()

```
void check_solution (
    fp_t ** conc_new,
    int nx,
    int ny,
    fp_t dx,
    fp_t dy,
    int nm,
    fp_t elapsed,
    fp_t D,
    fp_t bc[2][2],
    fp_t * rss )
```

Compare numerical and analytical solutions of the diffusion equation.

Returns the residual sum of squares (RSS), normalized to the domain size.

Definition at line 181 of file discretization.cpp.

4.5.2.3 compute_convolution()

```
void compute_convolution (
    fp_t ** conc_old,
    fp_t ** conc_lap,
    fp_t ** mask_lap,
    int nx,
    int ny,
    int nm )
```

Perform the convolution of the mask matrix with the composition matrix.

If the convolution mask is the Laplacian stencil, the convolution evaluates the discrete Laplacian of the composition field. Other masks are possible, for example the Sobel filters for edge detection. This function is general purpose: as long as the dimensions nx , ny , and nm are properly specified, the convolution will be correctly computed.

Definition at line 57 of file discretization.cpp.

4.5.2.4 set_boundaries()

```
void set_boundaries (
    fp_t bc[2][2] )
```

Set values to be used along the simulation domain boundaries.

Indexing is row-major, i.e. $A[y][x]$, so $bc = [[y_{lo}, y_{hi}], [x_{lo}, x_{hi}]]$.

Definition at line 41 of file boundaries.cpp.

4.5.2.5 set_threads()

```
void set_threads (
    int n )
```

Requested number of TBB threads to use in parallel code sections.

Warning

This setting does not appear to have any effect.

Definition at line 43 of file discretization.cpp.

4.6 Cuda

Files

- file [gpu-cuda-diffusion/boundaries.c](#)
Implementation of boundary condition functions with OpenMP threading.
- file [gpu-cuda-diffusion/boundaries.h](#)
Declaration of boundary condition function prototypes for CUDA benchmarks.
- file [discretization.cu](#)
Implementation of boundary condition functions with CUDA acceleration.
- file [gpu-cuda-diffusion/discretization.h](#)
Declaration of discretized mathematical function prototypes for CUDA benchmarks.

Macros

- `#define MAX_TILE_W 32`
Maximum width of an input tile, including halo cells, for GPU memory allocation.
- `#define MAX_TILE_H 32`
Maximum height of an input tile, including halo cells, for GPU memory allocation.

Functions

- void [set_boundaries](#) ([fp_t](#) bc[2][2])
Set values to be used along the simulation domain boundaries.
- void [apply_initial_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Initialize flat composition field with fixed boundary conditions.
- void [apply_boundary_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Set fixed value (c_{hi}) along left and bottom, zero-flux elsewhere.
- void [set_threads](#) (int n)
Set number of OpenMP threads to use in CPU code sections.
- `__global__` void [convolution_kernel](#) ([fp_t](#) *conc_old, [fp_t](#) *conc_lap, int nx, int ny, int nm)
Tiled convolution algorithm for execution on the GPU
This function accesses 1D data rather than the 2D array representation of the scalar composition field, mapping into 2D tiles on the GPU with halo cells before computing the convolution.
- void [compute_convolution](#) ([fp_t](#) **conc_old, [fp_t](#) **conc_lap, [fp_t](#) **mask_lap, int nx, int ny, int nm)
Perform the convolution of the mask matrix with the composition matrix.
- `__global__` void [diffusion_kernel](#) ([fp_t](#) *conc_old, [fp_t](#) *conc_new, [fp_t](#) *conc_lap, int nx, int ny, int nm, [fp_t](#) D, [fp_t](#) dt)
Vector addition algorithm for execution on the GPU
This function accesses 1D data rather than the 2D array representation of the scalar composition field.
- void [solve_diffusion_equation](#) ([fp_t](#) **conc_old, [fp_t](#) **conc_new, [fp_t](#) **conc_lap, int nx, int ny, int nm, [fp_t](#) D, [fp_t](#) dt, [fp_t](#) *elapsed)
Update the scalar composition field using old and Laplacian values.
- void [check_solution](#) ([fp_t](#) **conc_new, int nx, int ny, [fp_t](#) dx, [fp_t](#) dy, int nm, [fp_t](#) elapsed, [fp_t](#) D, [fp_t](#) bc[2][2], [fp_t](#) *rss)
Compare numerical and analytical solutions of the diffusion equation.

Variables

- `__constant__` [fp_t](#) Mc [MAX_MASK_W * MAX_MASK_H]
Allocate constant memory on the GPU for the convolution mask.

4.6.1 Detailed Description

4.6.2 Function Documentation

4.6.2.1 `apply_initial_conditions()`

```
void apply_initial_conditions (
    fp_t ** conc,
    int nx,
    int ny,
    int nm,
    fp_t bc[2][2] )
```

Initialize flat composition field with fixed boundary conditions.

The boundary conditions are fixed values of c_{hi} along the lower-left half and upper-right half walls, no flux everywhere else, with an initial values of c_{lo} everywhere. These conditions represent a carburizing process, with partial exposure (rather than the entire left and right walls) to produce an inhomogeneous workload and highlight numerical errors at the boundaries.

Definition at line 56 of file `gpu-cuda-diffusion/boundaries.c`.

4.6.2.2 `check_solution()`

```
void check_solution (
    fp_t ** conc_new,
    int nx,
    int ny,
    fp_t dx,
    fp_t dy,
    int nm,
    fp_t elapsed,
    fp_t D,
    fp_t bc[2][2],
    fp_t * rss )
```

Compare numerical and analytical solutions of the diffusion equation.

Returns the residual sum of squares (RSS), normalized to the domain size.

Definition at line 243 of file `discretization.cu`.

4.6.2.3 compute_convolution()

```
void compute_convolution (
    fp_t ** conc_old,
    fp_t ** conc_lap,
    fp_t ** mask_lap,
    int nx,
    int ny,
    int nm )
```

Perform the convolution of the mask matrix with the composition matrix.

If the convolution mask is the Laplacian stencil, the convolution evaluates the discrete Laplacian of the composition field. Other masks are possible, for example the Sobel filters for edge detection. This function is general purpose: as long as the dimensions `nx`, `ny`, and `nm` are properly specified, the convolution will be correctly computed.

Definition at line 145 of file `discretization.cu`.

4.6.2.4 set_boundaries()

```
void set_boundaries (
    fp_t bc[2][2] )
```

Set values to be used along the simulation domain boundaries.

Indexing is row-major, i.e. $A[y][x]$, so `bc` = $[[y_{lo}, y_{hi}], [x_{lo}, x_{hi}]]$.

Definition at line 38 of file `gpu-cuda-diffusion/boundaries.c`.

4.7 Openacc

Files

- file [gpu-openacc-diffusion/boundaries.c](#)
Implementation of boundary condition functions with OpenMP threading.
- file [gpu-openacc-diffusion/boundaries.h](#)
Declaration of boundary condition function prototypes for OpenAcc benchmarks.
- file [gpu-openacc-diffusion/discretization.c](#)
Implementation of boundary condition functions with OpenACC threading.
- file [gpu-openacc-diffusion/discretization.h](#)
Declaration of discretized mathematical function prototypes for OpenAcc benchmarks.

Functions

- void [set_boundaries](#) ([fp_t](#) bc[2][2])
Set values to be used along the simulation domain boundaries.
- void [apply_initial_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Initialize flat composition field with fixed boundary conditions.
- void [apply_boundary_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Set fixed value (c_{hi}) along left and bottom, zero-flux elsewhere.
- void [set_threads](#) (int n)
Set number of OpenMP threads to use in CPU code sections.
- void [compute_convolution](#) ([fp_t](#) **conc_old, [fp_t](#) **conc_lap, [fp_t](#) **mask_lap, int nx, int ny, int nm)
Perform the convolution of the mask matrix with the composition matrix.
- void [solve_diffusion_equation](#) ([fp_t](#) **conc_old, [fp_t](#) **conc_new, [fp_t](#) **conc_lap, int nx, int ny, int nm, [fp_t](#) D, [fp_t](#) dt, [fp_t](#) *elapsed)
Update the scalar composition field using old and Laplacian values.
- void [check_solution](#) ([fp_t](#) **conc_new, int nx, int ny, [fp_t](#) dx, [fp_t](#) dy, int nm, [fp_t](#) elapsed, [fp_t](#) D, [fp_t](#) bc[2][2], [fp_t](#) *rss)
Compare numerical and analytical solutions of the diffusion equation.

4.7.1 Detailed Description

4.7.2 Function Documentation

4.7.2.1 [apply_initial_conditions\(\)](#)

```
void apply_initial_conditions (
    fp\_t ** conc,
    int nx,
    int ny,
    int nm,
    fp\_t bc[2][2] )
```

Initialize flat composition field with fixed boundary conditions.

The boundary conditions are fixed values of c_{hi} along the lower-left half and upper-right half walls, no flux everywhere else, with an initial values of c_{lo} everywhere. These conditions represent a carburizing process, with partial exposure (rather than the entire left and right walls) to produce an inhomogeneous workload and highlight numerical errors at the boundaries.

Definition at line 56 of file [gpu-openacc-diffusion/boundaries.c](#).

4.7.2.2 check_solution()

```
void check_solution (
    fp_t ** conc_new,
    int nx,
    int ny,
    fp_t dx,
    fp_t dy,
    int nm,
    fp_t elapsed,
    fp_t D,
    fp_t bc[2][2],
    fp_t * rss )
```

Compare numerical and analytical solutions of the diffusion equation.

Returns the residual sum of squares (RSS), normalized to the domain size.

For 1D diffusion through a semi-infinite domain with initial and far-field composition c_∞ and boundary value $c(x = 0, t) = c_0$ with constant diffusivity D , the solution to Fick's second law is

$$c(x, t) = c_0 - (c_0 - c_\infty) \operatorname{erf} \left(\frac{x}{\sqrt{4Dt}} \right)$$

which reduces, when $c_\infty = 0$, to

$$c(x, t) = c_0 \left[1 - \operatorname{erf} \left(\frac{x}{\sqrt{4Dt}} \right) \right].$$

Definition at line 116 of file gpu-openacc-diffusion/discretization.c.

4.7.2.3 compute_convolution()

```
void compute_convolution (
    fp_t ** conc_old,
    fp_t ** conc_lap,
    fp_t ** mask_lap,
    int nx,
    int ny,
    int nm )
```

Perform the convolution of the mask matrix with the composition matrix.

If the convolution mask is the Laplacian stencil, the convolution evaluates the discrete Laplacian of the composition field. Other masks are possible, for example the Sobel filters for edge detection. This function is general purpose: as long as the dimensions `nx`, `ny`, and `nm` are properly specified, the convolution will be correctly computed.

Definition at line 52 of file gpu-openacc-diffusion/discretization.c.

4.7.2.4 set_boundaries()

```
void set_boundaries (
    fp_t bc[2][2] )
```

Set values to be used along the simulation domain boundaries.

Indexing is row-major, i.e. $A[y][x]$, so `bc` = $[[y_{lo}, y_{hi}], [x_{lo}, x_{hi}]]$.

Definition at line 38 of file gpu-openacc-diffusion/boundaries.c.

4.8 Opencil

Files

- file [discretization.cl](#)
Implementation of boundary condition functions with OpenCL acceleration.
- file [gpu-opencil-diffusion/discretization.h](#)
Declaration of discretized mathematical function prototypes for OpenCL benchmarks.

Macros

- `#define MAX_TILE_W 32`
Maximum width of an input tile, including halo cells, for GPU memory allocation.
- `#define MAX_TILE_H 32`
Maximum height of an input tile, including halo cells, for GPU memory allocation.
- `#define MAX_MASK_W 3`
Maximum height and width of the mask array, for GPU memory allocation.

Functions

- void [set_boundaries](#) ([fp_t](#) bc[2][2])
Set values to be used along the simulation domain boundaries.
- void [apply_initial_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Initialize flat composition field with fixed boundary conditions.
- void [apply_boundary_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Set fixed value (c_{hi}) along left and bottom, zero-flux elsewhere.
- void [set_threads](#) (int n)
Set number of OpenMP threads to use in CPU code sections.
- void [compute_convolution](#) ([fp_t](#) **conc_old, [fp_t](#) **conc_lap, [fp_t](#) **mask_lap, int nx, int ny, int nm, int bs)
Perform the convolution of the mask matrix with the composition matrix.
- void [solve_diffusion_equation](#) ([fp_t](#) **conc_old, [fp_t](#) **conc_new, [fp_t](#) **conc_lap, int nx, int ny, int nm, int bs, [fp_t](#) D, [fp_t](#) dt, [fp_t](#) *elapsed)
Update the scalar composition field using old and Laplacian values.
- void [check_solution](#) ([fp_t](#) **conc_new, int nx, int ny, [fp_t](#) dx, [fp_t](#) dy, int nm, [fp_t](#) elapsed, [fp_t](#) D, [fp_t](#) bc[2][2], [fp_t](#) *rss)
Compare numerical and analytical solutions of the diffusion equation.

4.8.1 Detailed Description

4.8.2 Function Documentation

4.8.2.1 apply_initial_conditions()

```
void apply_initial_conditions (
    fp_t ** conc,
    int nx,
    int ny,
    int nm,
    fp_t bc[2][2] )
```

Initialize flat composition field with fixed boundary conditions.

The boundary conditions are fixed values of c_{hi} along the lower-left half and upper-right half walls, no flux everywhere else, with an initial values of c_{lo} everywhere. These conditions represent a carburizing process, with partial exposure (rather than the entire left and right walls) to produce an inhomogeneous workload and highlight numerical errors at the boundaries.

Definition at line 56 of file gpu-opencl-diffusion/boundaries.c.

4.8.2.2 check_solution()

```
void check_solution (
    fp_t ** conc_new,
    int nx,
    int ny,
    fp_t dx,
    fp_t dy,
    int nm,
    fp_t elapsed,
    fp_t D,
    fp_t bc[2][2],
    fp_t * rss )
```

Compare numerical and analytical solutions of the diffusion equation.

Returns the residual sum of squares (RSS), normalized to the domain size.

Definition at line 250 of file discretization.cl.

4.8.2.3 compute_convolution()

```
void compute_convolution (
    fp_t ** conc_old,
    fp_t ** conc_lap,
    fp_t ** mask_lap,
    int nx,
    int ny,
    int nm,
    int bs )
```

Perform the convolution of the mask matrix with the composition matrix.

If the convolution mask is the Laplacian stencil, the convolution evaluates the discrete Laplacian of the composition field. Other masks are possible, for example the Sobel filters for edge detection. This function is general purpose: as long as the dimensions nx , ny , and nm are properly specified, the convolution will be correctly computed.

Definition at line 147 of file discretization.cl.

4.8.2.4 set_boundaries()

```
void set_boundaries (
    fp_t bc[2][2] )
```

Set values to be used along the simulation domain boundaries.

Indexing is row-major, i.e. $A[y][x]$, so $bc = [[y_{lo}, y_{hi}], [x_{lo}, x_{hi}]]$.

Definition at line 38 of file gpu-opencl-diffusion/boundaries.c.

5 Class Documentation

5.1 ResidualSumOfSquares2D Class Reference

Comparison algorithm for execution on the block of threads.

5.1.1 Detailed Description

Comparison algorithm for execution on the block of threads.

Definition at line 99 of file discretization.cpp.

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

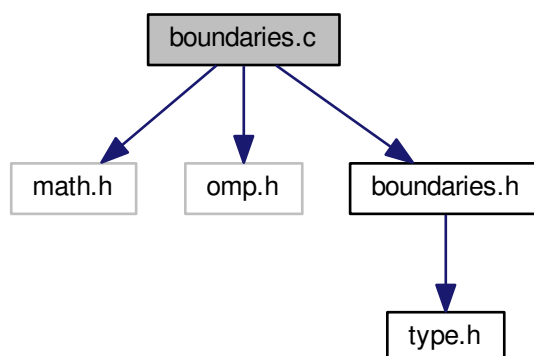
- [discretization.cpp](#)

6 File Documentation

6.1 boundaries.c File Reference

Implementation of boundary condition functions with OpenMP threading.

```
#include <math.h>
#include <omp.h>
#include "boundaries.h"
Include dependency graph for cpu-openmp-diffusion/boundaries.c:
```



Functions

- void `set_boundaries` (`fp_t` bc[2][2])
Set values to be used along the simulation domain boundaries.
- void `apply_initial_conditions` (`fp_t` **conc, int nx, int ny, int nm, `fp_t` bc[2][2])
Initialize flat composition field with fixed boundary conditions.
- void `apply_boundary_conditions` (`fp_t` **conc, int nx, int ny, int nm, `fp_t` bc[2][2])
Set fixed value (c_{hi}) along left and bottom, zero-flux elsewhere.

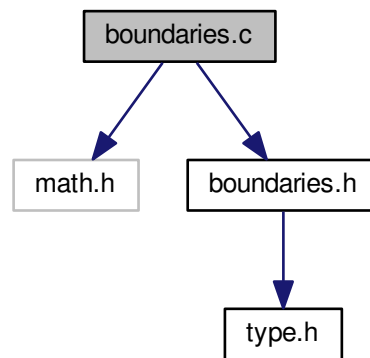
6.1.1 Detailed Description

Implementation of boundary condition functions with OpenMP threading.

6.2 boundaries.c File Reference

Implementation of boundary condition functions without threading.

```
#include <math.h>
#include "boundaries.h"
Include dependency graph for cpu-serial-diffusion/boundaries.c:
```



Functions

- void `set_boundaries` (`fp_t` bc[2][2])
Set values to be used along the simulation domain boundaries.
- void `apply_initial_conditions` (`fp_t` **conc, int nx, int ny, int nm, `fp_t` bc[2][2])
Initialize flat composition field with fixed boundary conditions.
- void `apply_boundary_conditions` (`fp_t` **conc, int nx, int ny, int nm, `fp_t` bc[2][2])
Set fixed value (c_{hi}) along left and bottom, zero-flux elsewhere.

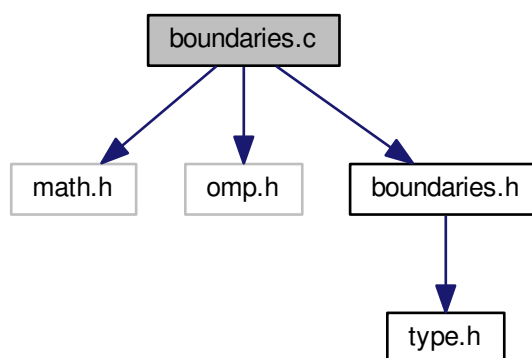
6.2.1 Detailed Description

Implementation of boundary condition functions without threading.

6.3 boundaries.c File Reference

Implementation of boundary condition functions with OpenMP threading.

```
#include <math.h>
#include <omp.h>
#include "boundaries.h"
Include dependency graph for gpu-cuda-diffusion/boundaries.c:
```



Functions

- void `set_boundaries` (`fp_t` bc[2][2])
Set values to be used along the simulation domain boundaries.
- void `apply_initial_conditions` (`fp_t` **conc, int nx, int ny, int nm, `fp_t` bc[2][2])
Initialize flat composition field with fixed boundary conditions.
- void `apply_boundary_conditions` (`fp_t` **conc, int nx, int ny, int nm, `fp_t` bc[2][2])
Set fixed value (c_{hi}) along left and bottom, zero-flux elsewhere.

6.3.1 Detailed Description

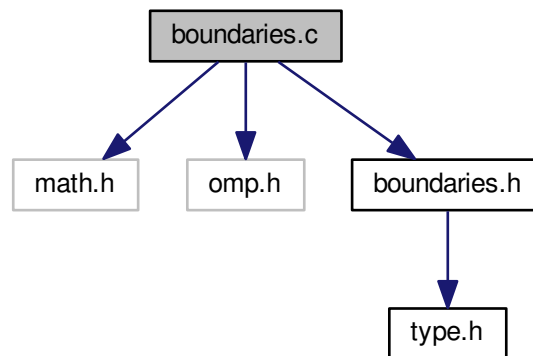
Implementation of boundary condition functions with OpenMP threading.

6.4 boundaries.c File Reference

Implementation of boundary condition functions with OpenMP threading.

```
#include <math.h>
#include <omp.h>
#include "boundaries.h"
```

Include dependency graph for gpu-openacc-diffusion/boundaries.c:



Functions

- void [set_boundaries](#) (fp_t bc[2][2])
Set values to be used along the simulation domain boundaries.
- void [apply_initial_conditions](#) (fp_t **conc, int nx, int ny, int nm, fp_t bc[2][2])
Initialize flat composition field with fixed boundary conditions.
- void [apply_boundary_conditions](#) (fp_t **conc, int nx, int ny, int nm, fp_t bc[2][2])
Set fixed value (c_{hi}) along left and bottom, zero-flux elsewhere.

6.4.1 Detailed Description

Implementation of boundary condition functions with OpenMP threading.

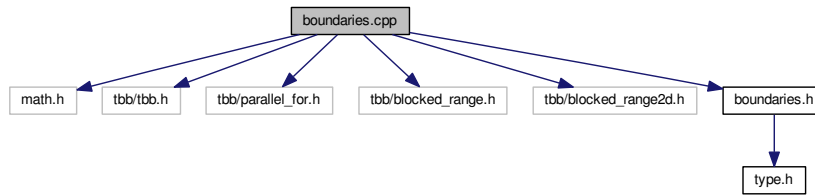
6.5 boundaries.cpp File Reference

Implementation of boundary condition functions with TBB threading.

```
#include <math.h>
#include <tbb/tbb.h>
#include <tbb/parallel_for.h>
#include <tbb/blocked_range.h>
#include <tbb/blocked_range2d.h>
```

```
#include "boundaries.h"
```

Include dependency graph for boundaries.cpp:



Functions

- void `set_boundaries` (`fp_t` bc[2][2])
Set values to be used along the simulation domain boundaries.
- void `apply_initial_conditions` (`fp_t` **conc, int nx, int ny, int nm, `fp_t` bc[2][2])
Initialize flat composition field with fixed boundary conditions.
- void `apply_boundary_conditions` (`fp_t` **conc, int nx, int ny, int nm, `fp_t` bc[2][2])
Set fixed value (c_{hi}) along left and bottom, zero-flux elsewhere.

6.5.1 Detailed Description

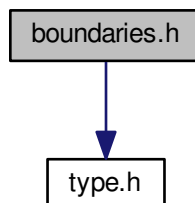
Implementation of boundary condition functions with TBB threading.

6.6 boundaries.h File Reference

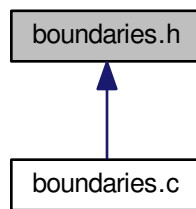
Declaration of boundary condition function prototypes for OpenMP benchmarks.

```
#include "type.h"
```

Include dependency graph for cpu-openmp-diffusion/boundaries.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [set_boundaries](#) ([fp_t](#) bc[2][2])
Set values to be used along the simulation domain boundaries.
- void [apply_initial_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Initialize flat composition field with fixed boundary conditions.
- void [apply_boundary_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Set fixed value (c_{hi}) along left and bottom, zero-flux elsewhere.

6.6.1 Detailed Description

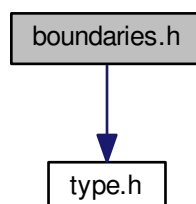
Declaration of boundary condition function prototypes for OpenMP benchmarks.

6.7 boundaries.h File Reference

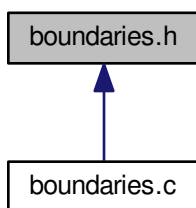
Declaration of boundary condition function prototypes for CPU benchmarks.

```
#include "type.h"
```

Include dependency graph for cpu-serial-diffusion/boundaries.h:



This graph shows which files directly or indirectly include this file:



Functions

- void `set_boundaries` (`fp_t` bc[2][2])
Set values to be used along the simulation domain boundaries.
- void `apply_initial_conditions` (`fp_t` **conc, int nx, int ny, int nm, `fp_t` bc[2][2])
Initialize flat composition field with fixed boundary conditions.
- void `apply_boundary_conditions` (`fp_t` **conc, int nx, int ny, int nm, `fp_t` bc[2][2])
Set fixed value (c_{hi}) along left and bottom, zero-flux elsewhere.

6.7.1 Detailed Description

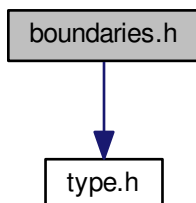
Declaration of boundary condition function prototypes for CPU benchmarks.

6.8 boundaries.h File Reference

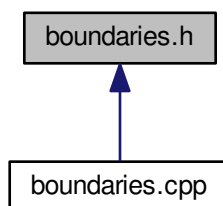
Declaration of boundary condition function prototypes for TBB benchmarks.

```
#include "type.h"
```

Include dependency graph for cpu-tbb-diffusion/boundaries.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [set_boundaries](#) ([fp_t](#) bc[2][2])
Set values to be used along the simulation domain boundaries.
- void [apply_initial_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Initialize flat composition field with fixed boundary conditions.
- void [apply_boundary_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Set fixed value (c_{hi}) along left and bottom, zero-flux elsewhere.

6.8.1 Detailed Description

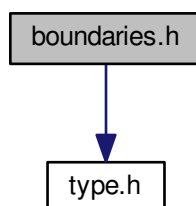
Declaration of boundary condition function prototypes for TBB benchmarks.

6.9 boundaries.h File Reference

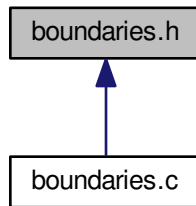
Declaration of boundary condition function prototypes for CUDA benchmarks.

```
#include "type.h"
```

Include dependency graph for gpu-cuda-diffusion/boundaries.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [set_boundaries](#) ([fp_t](#) bc[2][2])
Set values to be used along the simulation domain boundaries.
- void [apply_initial_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Initialize flat composition field with fixed boundary conditions.
- void [apply_boundary_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Set fixed value (c_{hi}) along left and bottom, zero-flux elsewhere.

6.9.1 Detailed Description

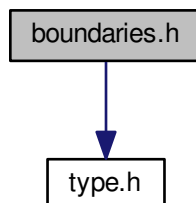
Declaration of boundary condition function prototypes for CUDA benchmarks.

6.10 boundaries.h File Reference

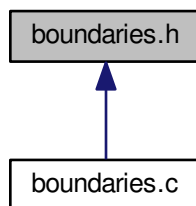
Declaration of boundary condition function prototypes for OpenAcc benchmarks.

```
#include "type.h"
```

Include dependency graph for gpu-openacc-diffusion/boundaries.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [set_boundaries](#) ([fp_t](#) bc[2][2])
Set values to be used along the simulation domain boundaries.
- void [apply_initial_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Initialize flat composition field with fixed boundary conditions.
- void [apply_boundary_conditions](#) ([fp_t](#) **conc, int nx, int ny, int nm, [fp_t](#) bc[2][2])
Set fixed value (c_{hi}) along left and bottom, zero-flux elsewhere.

6.10.1 Detailed Description

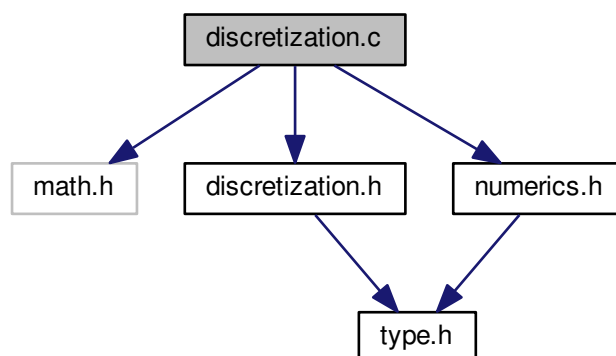
Declaration of boundary condition function prototypes for OpenAcc benchmarks.

6.11 discretization.c File Reference

Implementation of analytical solution functions.

```
#include <math.h>
#include "discretization.h"
#include "numerics.h"
```

Include dependency graph for cpu-analytic-diffusion/discretization.c:



Functions

- void `solve_diffusion_equation` (`fp_t **conc_old`, `fp_t **conc_new`, `fp_t **conc_lap`, `int nx`, `int ny`, `fp_t dx`, `fp_t dy`, `int nm`, `fp_t D`, `fp_t dt`, `fp_t elapsed`)

Update the scalar composition field using analytical solution.

6.11.1 Detailed Description

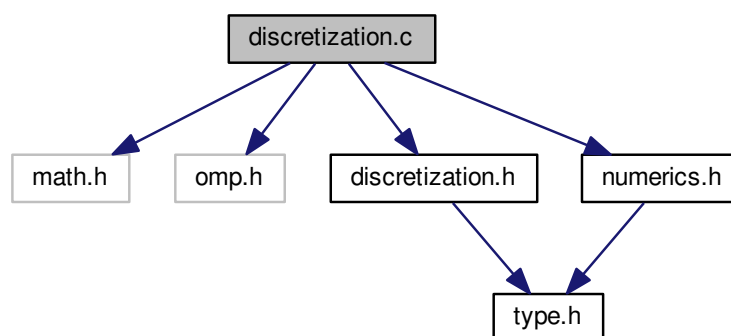
Implementation of analytical solution functions.

6.12 discretization.c File Reference

Implementation of boundary condition functions with OpenMP threading.

```
#include <math.h>
#include <omp.h>
#include "discretization.h"
#include "numerics.h"
```

Include dependency graph for `cpu-openmp-diffusion/discretization.c`:



Functions

- void `set_threads` (`int n`)
Set number of OpenMP threads to use in parallel code sections.
- void `compute_convolution` (`fp_t **conc_old`, `fp_t **conc_lap`, `fp_t **mask_lap`, `int nx`, `int ny`, `int nm`)
Perform the convolution of the mask matrix with the composition matrix.
- void `solve_diffusion_equation` (`fp_t **conc_old`, `fp_t **conc_new`, `fp_t **conc_lap`, `int nx`, `int ny`, `int nm`, `fp_t D`, `fp_t dt`, `fp_t elapsed`)
Update the scalar composition field using old and Laplacian values.

- void `check_solution` (`fp_t **conc_new`, `int nx`, `int ny`, `fp_t dx`, `fp_t dy`, `int nm`, `fp_t elapsed`, `fp_t D`, `fp_t bc[2][2]`, `fp_t *rss`)
Compare numerical and analytical solutions of the diffusion equation.

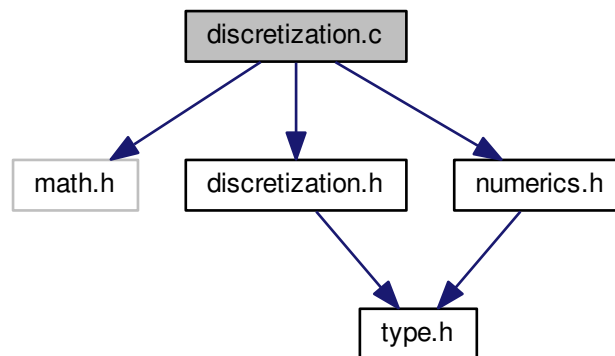
6.12.1 Detailed Description

Implementation of boundary condition functions with OpenMP threading.

6.13 discretization.c File Reference

Implementation of boundary condition functions without threading.

```
#include <math.h>
#include "discretization.h"
#include "numerics.h"
Include dependency graph for cpu-serial-diffusion/discretization.c:
```



Functions

- void `set_threads` (int n)
Set number of OpenMP threads to use in parallel code sections.
- void `compute_convolution` (fp_t **conc_old, fp_t **conc_lap, fp_t **mask_lap, int nx, int ny, int nm)
Perform the convolution of the mask matrix with the composition matrix.
- void `solve_diffusion_equation` (fp_t **conc_old, fp_t **conc_new, fp_t **conc_lap, int nx, int ny, int nm, fp_t D, fp_t dt, fp_t *elapsed)
Update the scalar composition field using old and Laplacian values.
- void `check_solution` (fp_t **conc_new, int nx, int ny, fp_t dx, fp_t dy, int nm, fp_t elapsed, fp_t D, fp_t bc[2][2], fp_t *rss)
Compare numerical and analytical solutions of the diffusion equation.

6.13.1 Detailed Description

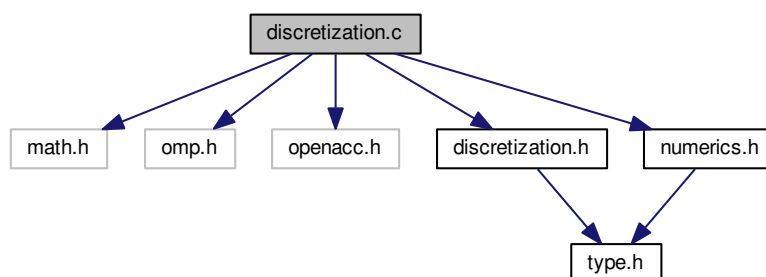
Implementation of boundary condition functions without threading.

6.14 discretization.c File Reference

Implementation of boundary condition functions with OpenACC threading.

```
#include <math.h>
#include <omp.h>
#include <openacc.h>
#include "discretization.h"
#include "numerics.h"
```

Include dependency graph for gpu-openacc-diffusion/discretization.c:



Functions

- void `set_threads` (int n)
Set number of OpenMP threads to use in CPU code sections.
- void `compute_convolution` (fp_t **conc_old, fp_t **conc_lap, fp_t **mask_lap, int nx, int ny, int nm)
Perform the convolution of the mask matrix with the composition matrix.
- void `solve_diffusion_equation` (fp_t **conc_old, fp_t **conc_new, fp_t **conc_lap, int nx, int ny, int nm, fp_t D, fp_t dt, fp_t *elapsed)
Update the scalar composition field using old and Laplacian values.
- void `check_solution` (fp_t **conc_new, int nx, int ny, fp_t dx, fp_t dy, int nm, fp_t elapsed, fp_t D, fp_t bc[2][2], fp_t *rss)
Compare numerical and analytical solutions of the diffusion equation.

6.14.1 Detailed Description

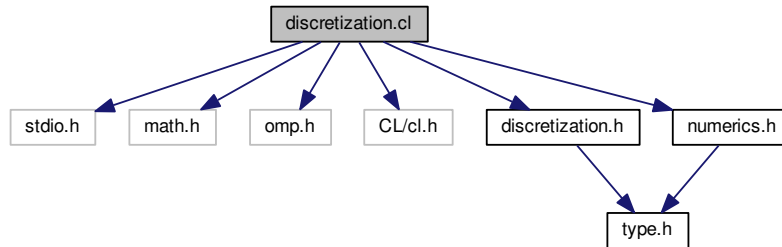
Implementation of boundary condition functions with OpenACC threading.

6.15 discretization.cl File Reference

Implementation of boundary condition functions with OpenCL acceleration.

```
#include <stdio.h>
#include <math.h>
#include <omp.h>
#include <CL/cl.h>
```

```
#include "discretization.h"
#include "numerics.h"
Include dependency graph for discretization.cl:
```



Macros

- `#define MAX_TILE_W 32`
Maximum width of an input tile, including halo cells, for GPU memory allocation.
- `#define MAX_TILE_H 32`
Maximum height of an input tile, including halo cells, for GPU memory allocation.
- `#define MAX_MASK_W 3`
Maximum height and width of the mask array, for GPU memory allocation.

Functions

- `void set_threads (int n)`
Set number of OpenMP threads to use in CPU code sections.
- `void compute_convolution (fp_t **conc_old, fp_t **conc_lap, fp_t **mask_lap, int nx, int ny, int nm, int bs)`
Perform the convolution of the mask matrix with the composition matrix.
- `void solve_diffusion_equation (fp_t **conc_old, fp_t **conc_new, fp_t **conc_lap, int nx, int ny, int nm, int bs, fp_t D, fp_t dt, fp_t *elapsed)`
Update the scalar composition field using old and Laplacian values.
- `void check_solution (fp_t **conc_new, int nx, int ny, fp_t dx, fp_t dy, int nm, fp_t elapsed, fp_t D, fp_t bc[2][2], fp_t *rss)`
Compare numerical and analytical solutions of the diffusion equation.

6.15.1 Detailed Description

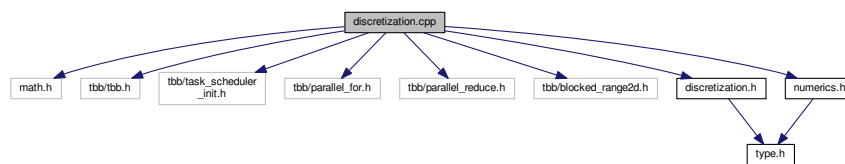
Implementation of boundary condition functions with OpenCL acceleration.

6.16 discretization.cpp File Reference

Implementation of boundary condition functions with TBB threading.

```
#include <math.h>
#include <tbb/tbb.h>
#include <tbb/task_scheduler_init.h>
#include <tbb/parallel_for.h>
#include <tbb/parallel_reduce.h>
#include <tbb/blocked_range2d.h>
#include "discretization.h"
#include "numerics.h"
```

Include dependency graph for discretization.cpp:



Classes

- class [ResidualSumOfSquares2D](#)

Comparison algorithm for execution on the block of threads.

Functions

- void [set_threads](#) (int n)
Requested number of TBB threads to use in parallel code sections.
- void [compute_convolution](#) (fp_t **conc_old, fp_t **conc_lap, fp_t **mask_lap, int nx, int ny, int nm)
Perform the convolution of the mask matrix with the composition matrix.
- void [solve_diffusion_equation](#) (fp_t **conc_old, fp_t **B, fp_t **conc_lap, int nx, int ny, int nm, fp_t D, fp_t dt, fp_t *elapsed)
Update the scalar composition field using old and Laplacian values.
- void [check_solution](#) (fp_t **conc_new, int nx, int ny, fp_t dx, fp_t dy, int nm, fp_t elapsed, fp_t D, fp_t bc[2][2], fp_t *rss)
Compare numerical and analytical solutions of the diffusion equation.

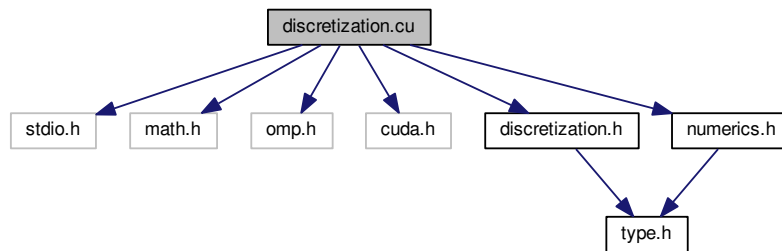
6.16.1 Detailed Description

Implementation of boundary condition functions with TBB threading.

6.17 discretization.cu File Reference

Implementation of boundary condition functions with CUDA acceleration.

```
#include <stdio.h>
#include <math.h>
#include <omp.h>
#include <cuda.h>
#include "discretization.h"
#include "numerics.h"
Include dependency graph for discretization.cu:
```



Macros

- `#define MAX_TILE_W 32`
Maximum width of an input tile, including halo cells, for GPU memory allocation.
- `#define MAX_TILE_H 32`
Maximum height of an input tile, including halo cells, for GPU memory allocation.

Functions

- `void set_threads (int n)`
Set number of OpenMP threads to use in CPU code sections.
- `__global__ void convolution_kernel (fp_t *conc_old, fp_t *conc_lap, int nx, int ny, int nm)`
*Tiled convolution algorithm for execution on the GPU
This function accesses 1D data rather than the 2D array representation of the scalar composition field, mapping into 2D tiles on the GPU with halo cells before computing the convolution.*
- `void compute_convolution (fp_t **conc_old, fp_t **conc_lap, fp_t **mask_lap, int nx, int ny, int nm)`
Perform the convolution of the mask matrix with the composition matrix.
- `__global__ void diffusion_kernel (fp_t *conc_old, fp_t *conc_new, fp_t *conc_lap, int nx, int ny, int nm, fp_t D, fp_t dt)`
*Vector addition algorithm for execution on the GPU
This function accesses 1D data rather than the 2D array representation of the scalar composition field.*
- `void solve_diffusion_equation (fp_t **conc_old, fp_t **conc_new, fp_t **conc_lap, int nx, int ny, int nm, fp_t D, fp_t dt, fp_t *elapsed)`
Update the scalar composition field using old and Laplacian values.
- `void check_solution (fp_t **conc_new, int nx, int ny, fp_t dx, fp_t dy, int nm, fp_t elapsed, fp_t D, fp_t bc[2][2], fp_t *rss)`
Compare numerical and analytical solutions of the diffusion equation.

Variables

- `__constant__ fp_t Mc [MAX_MASK_W * MAX_MASK_H]`
Allocate constant memory on the GPU for the convolution mask.

6.17.1 Detailed Description

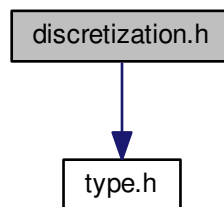
Implementation of boundary condition functions with CUDA acceleration.

6.18 discretization.h File Reference

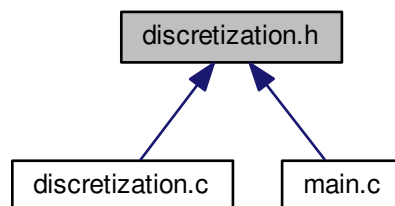
Declaration of analytical solution prototypes.

```
#include "type.h"
```

Include dependency graph for `cpu-analytic-diffusion/discretization.h`:



This graph shows which files directly or indirectly include this file:



Functions

- void `solve_diffusion_equation` (`fp_t **conc_old`, `fp_t **conc_new`, `fp_t **conc_lap`, `int nx`, `int ny`, `fp_t dx`, `fp_t dy`, `int nm`, `fp_t D`, `fp_t dt`, `fp_t elapsed`)
Update the scalar composition field using analytical solution.

6.18.1 Detailed Description

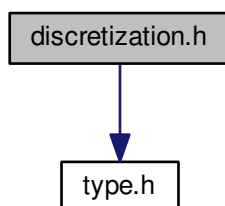
Declaration of analytical solution prototypes.

6.19 discretization.h File Reference

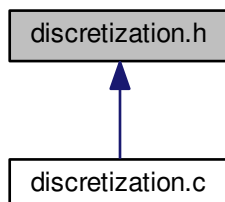
Declaration of discretized mathematical function prototypes for OpenMP benchmarks.

```
#include "type.h"
```

Include dependency graph for cpu-openmp-diffusion/discretization.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [set_threads](#) (int n)
Set number of OpenMP threads to use in parallel code sections.
- void [compute_convolution](#) (fp_t **conc_old, fp_t **conc_lap, fp_t **mask_lap, int nx, int ny, int nm)
Perform the convolution of the mask matrix with the composition matrix.
- void [solve_diffusion_equation](#) (fp_t **conc_old, fp_t **conc_new, fp_t **conc_lap, int nx, int ny, int nm, fp_t D, fp_t dt, fp_t *elapsed)
Update the scalar composition field using old and Laplacian values.
- void [check_solution](#) (fp_t **conc_new, int nx, int ny, fp_t dx, fp_t dy, int nm, fp_t elapsed, fp_t D, fp_t bc[2][2], fp_t *rss)
Compare numerical and analytical solutions of the diffusion equation.

6.19.1 Detailed Description

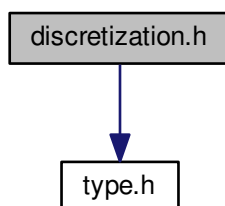
Declaration of discretized mathematical function prototypes for OpenMP benchmarks.

6.20 discretization.h File Reference

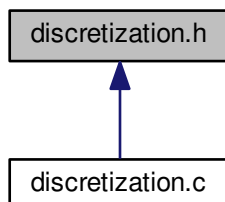
Declaration of discretized mathematical function prototypes for CPU benchmarks.

```
#include "type.h"
```

Include dependency graph for cpu-serial-diffusion/discretization.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [set_threads](#) (int n)
Set number of OpenMP threads to use in parallel code sections.
- void [compute_convolution](#) (fp_t **conc_old, fp_t **conc_lap, fp_t **mask_lap, int nx, int ny, int nm)
Perform the convolution of the mask matrix with the composition matrix.
- void [solve_diffusion_equation](#) (fp_t **conc_old, fp_t **conc_new, fp_t **conc_lap, int nx, int ny, int nm, fp_t D, fp_t dt, fp_t *elapsed)
Update the scalar composition field using old and Laplacian values.
- void [check_solution](#) (fp_t **conc_new, int nx, int ny, fp_t dx, fp_t dy, int nm, fp_t elapsed, fp_t D, fp_t bc[2][2], fp_t *rss)
Compare numerical and analytical solutions of the diffusion equation.

6.20.1 Detailed Description

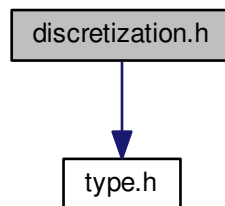
Declaration of discretized mathematical function prototypes for CPU benchmarks.

6.21 discretization.h File Reference

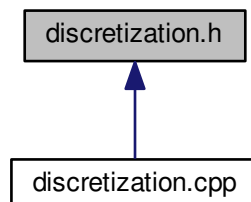
Declaration of discretized mathematical function prototypes for TBB benchmarks.

```
#include "type.h"
```

Include dependency graph for cpu-tbb-diffusion/discretization.h:



This graph shows which files directly or indirectly include this file:



Functions

- void `set_threads` (int n)
Requested number of TBB threads to use in parallel code sections.
- void `compute_convolution` (fp_t **conc_old, fp_t **conc_lap, fp_t **mask_lap, int nx, int ny, int nm)
Perform the convolution of the mask matrix with the composition matrix.
- void `solve_diffusion_equation` (fp_t **conc_old, fp_t **B, fp_t **conc_lap, int nx, int ny, int nm, fp_t D, fp_t dt, fp_t *elapsed)
Update the scalar composition field using old and Laplacian values.
- void `check_solution` (fp_t **conc_new, int nx, int ny, fp_t dx, fp_t dy, int nm, fp_t elapsed, fp_t D, fp_t bc[2][2], fp_t *rss)
Compare numerical and analytical solutions of the diffusion equation.

6.21.1 Detailed Description

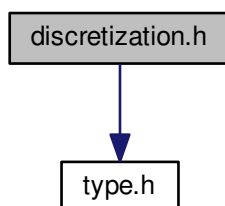
Declaration of discretized mathematical function prototypes for TBB benchmarks.

6.22 discretization.h File Reference

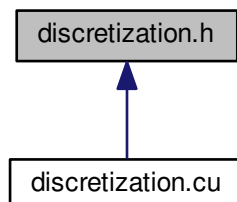
Declaration of discretized mathematical function prototypes for CUDA benchmarks.

```
#include "type.h"
```

Include dependency graph for gpu-cuda-diffusion/discretization.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [set_threads](#) (int n)
Set number of OpenMP threads to use in CPU code sections.
- void [compute_convolution](#) (fp_t **conc_old, fp_t **conc_lap, fp_t **mask_lap, int nx, int ny, int nm)
Perform the convolution of the mask matrix with the composition matrix.
- void [solve_diffusion_equation](#) (fp_t **conc_old, fp_t **conc_new, fp_t **conc_lap, int nx, int ny, int nm, fp_t D, fp_t dt, fp_t *elapsed)
Update the scalar composition field using old and Laplacian values.
- void [check_solution](#) (fp_t **conc_new, int nx, int ny, fp_t dx, fp_t dy, int nm, fp_t elapsed, fp_t D, fp_t bc[2][2], fp_t *rss)
Compare numerical and analytical solutions of the diffusion equation.

6.22.1 Detailed Description

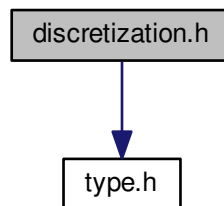
Declaration of discretized mathematical function prototypes for CUDA benchmarks.

6.23 discretization.h File Reference

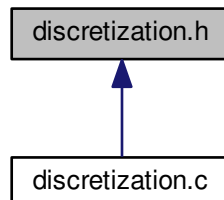
Declaration of discretized mathematical function prototypes for OpenAcc benchmarks.

```
#include "type.h"
```

Include dependency graph for gpu-openacc-diffusion/discretization.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [set_threads](#) (int n)
Set number of OpenMP threads to use in CPU code sections.
- void [compute_convolution](#) ([fp_t](#) **conc_old, [fp_t](#) **conc_lap, [fp_t](#) **mask_lap, int nx, int ny, int nm)
Perform the convolution of the mask matrix with the composition matrix.
- void [solve_diffusion_equation](#) ([fp_t](#) **conc_old, [fp_t](#) **conc_new, [fp_t](#) **conc_lap, int nx, int ny, int nm, [fp_t](#) D, [fp_t](#) dt, [fp_t](#) *elapsed)
Update the scalar composition field using old and Laplacian values.
- void [check_solution](#) ([fp_t](#) **conc_new, int nx, int ny, [fp_t](#) dx, [fp_t](#) dy, int nm, [fp_t](#) elapsed, [fp_t](#) D, [fp_t](#) bc[2][2], [fp_t](#) *rss)
Compare numerical and analytical solutions of the diffusion equation.

6.23.1 Detailed Description

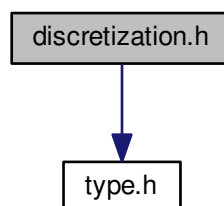
Declaration of discretized mathematical function prototypes for OpenAcc benchmarks.

6.24 discretization.h File Reference

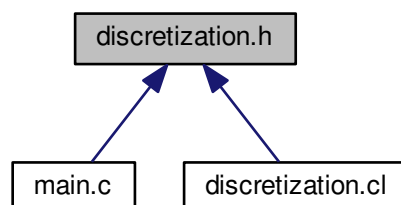
Declaration of discretized mathematical function prototypes for OpenCL benchmarks.

```
#include "type.h"
```

Include dependency graph for gpu-openc1-diffusion/discretization.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [set_threads](#) (int n)
Set number of OpenMP threads to use in CPU code sections.
- void [compute_convolution](#) (fp_t **conc_old, fp_t **conc_lap, fp_t **mask_lap, int nx, int ny, int nm, int bs)
Perform the convolution of the mask matrix with the composition matrix.
- void [solve_diffusion_equation](#) (fp_t **conc_old, fp_t **conc_new, fp_t **conc_lap, int nx, int ny, int nm, int bs, fp_t D, fp_t dt, fp_t *elapsed)
Update the scalar composition field using old and Laplacian values.
- void [check_solution](#) (fp_t **conc_new, int nx, int ny, fp_t dx, fp_t dy, int nm, fp_t elapsed, fp_t D, fp_t bc[2][2], fp_t *rss)
Compare numerical and analytical solutions of the diffusion equation.

6.24.1 Detailed Description

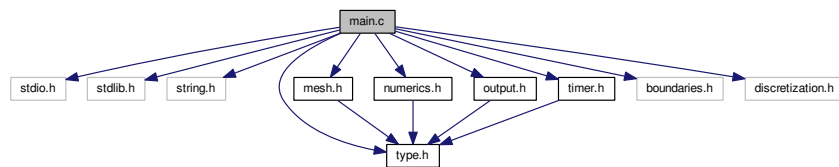
Declaration of discretized mathematical function prototypes for OpenCL benchmarks.

6.25 main.c File Reference

Implementation of semi-infinite diffusion equation.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "type.h"
#include "mesh.h"
#include "numerics.h"
#include "output.h"
#include "timer.h"
#include "boundaries.h"
#include "discretization.h"
```

Include dependency graph for common-diffusion/main.c:



Functions

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

Run simulation using input parameters specified on the command line.

6.25.1 Detailed Description

Implementation of semi-infinite diffusion equation.

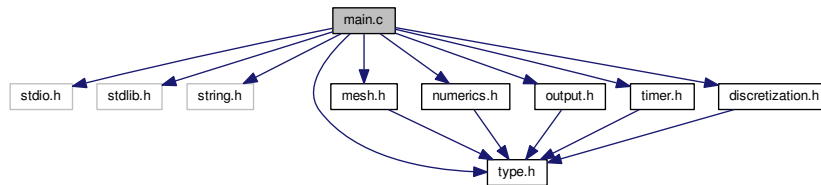
6.26 main.c File Reference

Analytical solution to semi-infinite diffusion equation.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "type.h"
#include "mesh.h"
#include "numerics.h"
#include "output.h"
#include "timer.h"
```

```
#include "discretization.h"
```

Include dependency graph for cpu-analytic-diffusion/main.c:



Functions

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

Run simulation using input parameters specified on the command line.

6.26.1 Detailed Description

Analytical solution to semi-infinite diffusion equation.

6.27 mesh.c File Reference

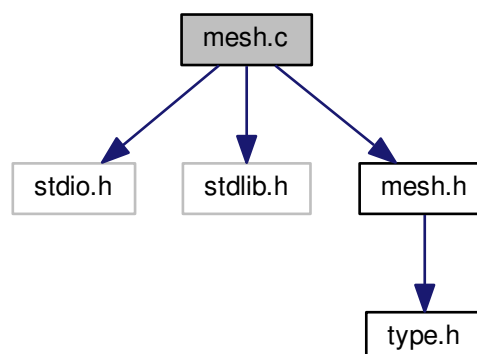
Implementation of mesh handling functions for diffusion benchmarks.

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include "mesh.h"
```

Include dependency graph for mesh.c:



Functions

- void `make_arrays` (`fp_t ***conc_old`, `fp_t ***conc_new`, `fp_t ***conc_lap`, `fp_t ***mask_lap`, `int nx`, `int ny`, `int nm`)
Allocate 2D arrays to store scalar composition values.
- void `free_arrays` (`fp_t **conc_old`, `fp_t **conc_new`, `fp_t **conc_lap`, `fp_t **mask_lap`)
Free dynamically allocated memory.
- void `swap_pointers` (`fp_t ***conc_old`, `fp_t ***conc_new`)
Swap pointers to data underlying two arrays.

6.27.1 Detailed Description

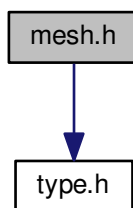
Implementation of mesh handling functions for diffusion benchmarks.

6.28 mesh.h File Reference

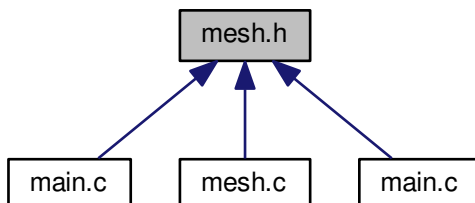
Declaration of mesh function prototypes for diffusion benchmarks.

```
#include "type.h"
```

Include dependency graph for mesh.h:



This graph shows which files directly or indirectly include this file:



Functions

- void `make_arrays` (`fp_t ***conc_old`, `fp_t ***conc_new`, `fp_t ***conc_lap`, `fp_t ***mask_lap`, `int nx`, `int ny`, `int nm`)
Allocate 2D arrays to store scalar composition values.
- void `free_arrays` (`fp_t **conc_old`, `fp_t **conc_new`, `fp_t **conc_lap`, `fp_t **mask_lap`)
Free dynamically allocated memory.
- void `swap_pointers` (`fp_t ***conc_old`, `fp_t ***conc_new`)
Swap pointers to data underlying two arrays.

6.28.1 Detailed Description

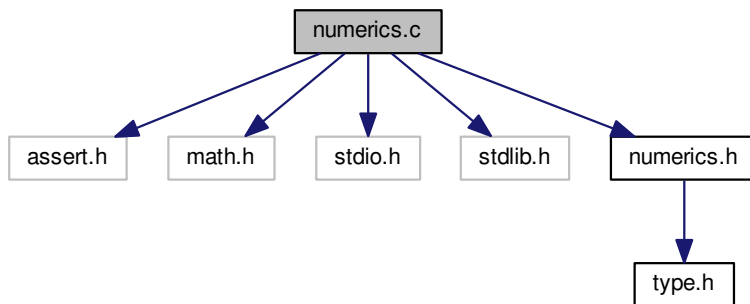
Declaration of mesh function prototypes for diffusion benchmarks.

6.29 numerics.c File Reference

Implementation of boundary condition functions with OpenMP threading.

```
#include <assert.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include "numerics.h"
```

Include dependency graph for numerics.c:



Functions

- void `set_mask` (`fp_t dx`, `fp_t dy`, `int code`, `fp_t **mask_lap`, `int nm`)
Specify which stencil (mask) to use for the Laplacian (convolution)
- void `five_point_Laplacian_stencil` (`fp_t dx`, `fp_t dy`, `fp_t **mask_lap`, `int nm`)
Write 5-point Laplacian stencil into convolution mask.
- void `nine_point_Laplacian_stencil` (`fp_t dx`, `fp_t dy`, `fp_t **mask_lap`, `int nm`)
Write 9-point Laplacian stencil into convolution mask.
- void `slow_nine_point_Laplacian_stencil` (`fp_t dx`, `fp_t dy`, `fp_t **mask_lap`, `int nm`)

Write 9-point Laplacian stencil into convolution mask.

- `fp_t euclidean_distance` (`fp_t ax`, `fp_t ay`, `fp_t bx`, `fp_t by`)

Compute Euclidean distance between two points, a and b.

- `fp_t manhattan_distance` (`fp_t ax`, `fp_t ay`, `fp_t bx`, `fp_t by`)

Compute Manhattan distance between two points, a and b.

- `fp_t distance_point_to_segment` (`fp_t ax`, `fp_t ay`, `fp_t bx`, `fp_t by`, `fp_t px`, `fp_t py`)

Compute minimum distance from point p to a line segment bounded by points a and b.

- `void analytical_value` (`fp_t x`, `fp_t t`, `fp_t D`, `fp_t bc[2][2]`, `fp_t *c`)

Analytical solution of the diffusion equation for a carburizing process.

6.29.1 Detailed Description

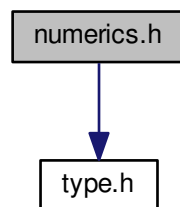
Implementation of boundary condition functions with OpenMP threading.

6.30 numerics.h File Reference

Declaration of Laplacian operator and analytical solution functions.

```
#include "type.h"
```

Include dependency graph for numerics.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_MASK_W 5`

Maximum width of the convolution mask (Laplacian stencil) array.

- `#define MAX_MASK_H 5`

Maximum height of the convolution mask (Laplacian stencil) array.

Functions

- void `set_mask` (`fp_t` dx, `fp_t` dy, int code, `fp_t` **mask_lap, int nm)
Specify which stencil (mask) to use for the Laplacian (convolution)
- void `five_point_Laplacian_stencil` (`fp_t` dx, `fp_t` dy, `fp_t` **mask_lap, int nm)
Write 5-point Laplacian stencil into convolution mask.
- void `nine_point_Laplacian_stencil` (`fp_t` dx, `fp_t` dy, `fp_t` **mask_lap, int nm)
Write 9-point Laplacian stencil into convolution mask.
- void `slow_nine_point_Laplacian_stencil` (`fp_t` dx, `fp_t` dy, `fp_t` **mask_lap, int nm)
Write 9-point Laplacian stencil into convolution mask.
- `fp_t` `euclidean_distance` (`fp_t` ax, `fp_t` ay, `fp_t` bx, `fp_t` by)
Compute Euclidean distance between two points, a and b.
- `fp_t` `manhattan_distance` (`fp_t` ax, `fp_t` ay, `fp_t` bx, `fp_t` by)
Compute Manhattan distance between two points, a and b.
- `fp_t` `distance_point_to_segment` (`fp_t` ax, `fp_t` ay, `fp_t` bx, `fp_t` by, `fp_t` px, `fp_t` py)
Compute minimum distance from point p to a line segment bounded by points a and b.
- void `analytical_value` (`fp_t` x, `fp_t` t, `fp_t` D, `fp_t` bc[2][2], `fp_t` *c)
Analytical solution of the diffusion equation for a carburizing process.

6.30.1 Detailed Description

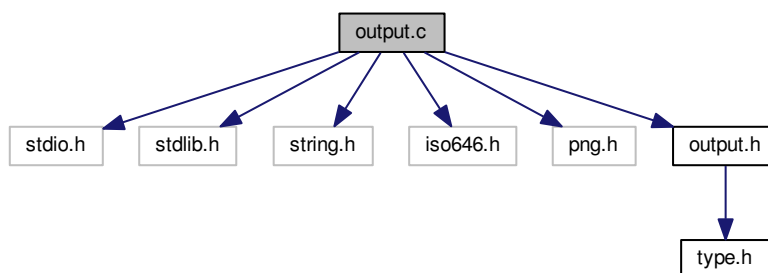
Declaration of Laplacian operator and analytical solution functions.

6.31 output.c File Reference

Implementation of file output functions for diffusion benchmarks.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iso646.h>
#include <png.h>
#include "output.h"
```

Include dependency graph for output.c:



Functions

- void `print_progress` (const int step, const int steps)
Prints timestamps and a 20-point progress bar to stdout.
- void `write_csv` (fp_t **conc, int nx, int ny, fp_t dx, fp_t dy, int step)
Writes scalar composition field to diffusion.?????.csv.
- void `write_png` (fp_t **conc, int nx, int ny, int step)
Writes scalar composition field to diffusion.?????.png.

6.31.1 Detailed Description

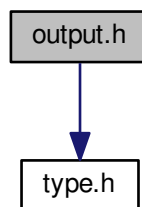
Implementation of file output functions for diffusion benchmarks.

6.32 output.h File Reference

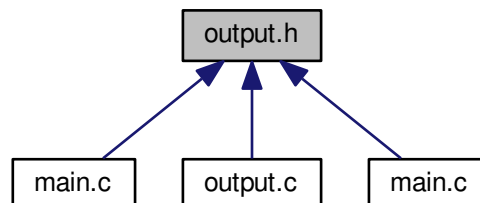
Declaration of output function prototypes for diffusion benchmarks.

```
#include "type.h"
```

Include dependency graph for output.h:



This graph shows which files directly or indirectly include this file:



Functions

- void `print_progress` (const int step, const int steps)
Prints timestamps and a 20-point progress bar to stdout.
- void `write_csv` (fp_t **conc, int nx, int ny, fp_t dx, fp_t dy, int step)
Writes scalar composition field to diffusion.?????.csv.
- void `write_png` (fp_t **conc, int nx, int ny, int step)
Writes scalar composition field to diffusion.?????.png.

6.32.1 Detailed Description

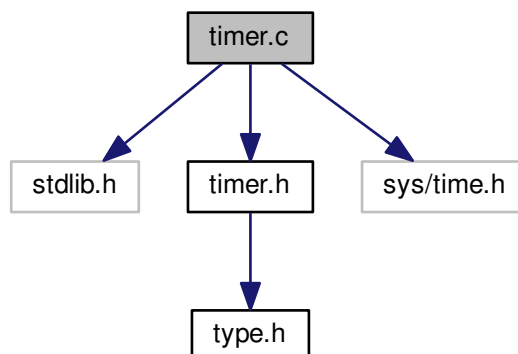
Declaration of output function prototypes for diffusion benchmarks.

6.33 timer.c File Reference

High-resolution cross-platform machine time reader.

```
#include <stdlib.h>
#include "timer.h"
#include <sys/time.h>
```

Include dependency graph for timer.c:



Functions

- void `StartTimer` ()
Set CPU frequency and begin timing.
- double `GetTimer` ()
Return elapsed time in seconds.

Variables

- struct timeval `timerStart`

6.33.1 Detailed Description

High-resolution cross-platform machine time reader.

Author

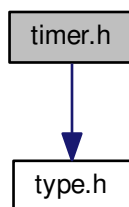
NVIDIA

6.34 timer.h File Reference

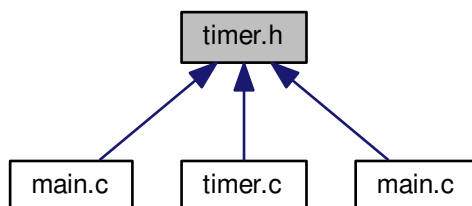
Declaration of timer function prototypes for diffusion benchmarks.

```
#include "type.h"
```

Include dependency graph for timer.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [StartTimer](#) ()
Set CPU frequency and begin timing.
- double [GetTimer](#) ()
Return elapsed time in seconds.

Index

- Analytic, 10
 - main, 10
- analytical_value
 - Diffusion equation benchmark, 6
- apply_initial_conditions
 - Cuda, 19
 - Openacc, 21
 - Opencl, 23
 - Openmp, 11
 - Serial, 13
 - Tbb, 15
- boundaries.c, 26–29
- boundaries.cpp, 29
- boundaries.h, 30–34
- check_solution
 - Cuda, 19
 - Openacc, 21
 - Opencl, 24
 - Openmp, 11
 - Serial, 13
 - Tbb, 16
- compute_convolution
 - Cuda, 19
 - Openacc, 22
 - Opencl, 24
 - Openmp, 12
 - Serial, 14
 - Tbb, 16
- Cuda, 18
 - apply_initial_conditions, 19
 - check_solution, 19
 - compute_convolution, 19
 - set_boundaries, 20
- Diffusion equation benchmark, 4
 - analytical_value, 6
 - distance_point_to_segment, 6
 - five_point_Laplacian_stencil, 6
 - fp_t, 5
 - main, 7
 - make_arrays, 7
 - nine_point_Laplacian_stencil, 7
 - print_progress, 8
 - set_mask, 8
 - slow_nine_point_Laplacian_stencil, 8
 - swap_pointers, 9
 - timerStart, 9
- discretization.c, 35–38
- discretization.cl, 38
- discretization.cpp, 40
- discretization.cu, 41
- discretization.h, 42–48
- distance_point_to_segment
 - Diffusion equation benchmark, 6
- five_point_Laplacian_stencil
 - Diffusion equation benchmark, 6
- fp_t
 - Diffusion equation benchmark, 5
- main
 - Analytic, 10
 - Diffusion equation benchmark, 7
- main.c, 49
- make_arrays
 - Diffusion equation benchmark, 7
- mesh.c, 50
- mesh.h, 51
- nine_point_Laplacian_stencil
 - Diffusion equation benchmark, 7
- numerics.c, 52
- numerics.h, 53
- Openacc, 21
 - apply_initial_conditions, 21
 - check_solution, 21
 - compute_convolution, 22
 - set_boundaries, 22
- Opencl, 23
 - apply_initial_conditions, 23
 - check_solution, 24
 - compute_convolution, 24
 - set_boundaries, 24
- Openmp, 11
 - apply_initial_conditions, 11
 - check_solution, 11
 - compute_convolution, 12
 - set_boundaries, 12
- output.c, 54
- output.h, 55
- print_progress
 - Diffusion equation benchmark, 8
- ResidualSumOfSquares2D, 26
- Serial, 13
 - apply_initial_conditions, 13
 - check_solution, 13
 - compute_convolution, 14
 - set_boundaries, 14
 - set_threads, 14
- set_boundaries
 - Cuda, 20
 - Openacc, 22
 - Opencl, 24
 - Openmp, 12
 - Serial, 14
 - Tbb, 16
- set_mask
 - Diffusion equation benchmark, 8

- set_threads
 - Serial, [14](#)
 - Tbb, [17](#)
- slow_nine_point_Laplacian_stencil
 - Diffusion equation benchmark, [8](#)
- swap_pointers
 - Diffusion equation benchmark, [9](#)
- Tbb, [15](#)
 - apply_initial_conditions, [15](#)
 - check_solution, [16](#)
 - compute_convolution, [16](#)
 - set_boundaries, [16](#)
 - set_threads, [17](#)
- timer.c, [56](#)
- timer.h, [57](#)
- timerStart
 - Diffusion equation benchmark, [9](#)
- type.h, [58](#)