

AccFFT

Generated by Doxygen 1.8.9.1

Tue Mar 1 2016 21:51:53

Contents

1	My Main Page	1
2	Timing AccFFT	3
3	AccFFT Flags	5
4	File Index	7
4.1	File List	7
5	File Documentation	9
5.1	/Users/new/accfft/src/accfft.cpp File Reference	9
5.1.1	Detailed Description	9
5.1.2	Function Documentation	9
5.1.2.1	accfft_destroy_plan	9
5.1.2.2	accfft_execute_c2c	10
5.1.2.3	accfft_execute_c2r	10
5.1.2.4	accfft_execute_r2c	10
5.1.2.5	accfft_local_size_dft_c2c	11
5.1.2.6	accfft_local_size_dft_r2c	11
5.1.2.7	accfft_plan_dft_3d_c2c	11
5.1.2.8	accfft_plan_dft_3d_r2c	12
5.2	/Users/new/accfft/src/accfft_common.cpp File Reference	12
5.2.1	Detailed Description	12
5.2.2	Function Documentation	13
5.2.2.1	accfft_alloc	13
5.2.2.2	accfft_cleanup	14
5.2.2.3	accfft_create_comm	14
5.2.2.4	accfft_free	14
5.2.2.5	accfft_init	14
5.2.2.6	accfft_init	14
5.3	/Users/new/accfft/src/accfft_gpu.cpp File Reference	14
5.3.1	Detailed Description	15
5.3.2	Function Documentation	15

5.3.2.1	accfft_cleanup_gpu	15
5.3.2.2	accfft_destroy_plan	15
5.3.2.3	accfft_destroy_plan_gpu	15
5.3.2.4	accfft_execute_c2c_gpu	16
5.3.2.5	accfft_execute_c2r_gpu	16
5.3.2.6	accfft_execute_r2c_gpu	16
5.3.2.7	accfft_local_size_dft_c2c_gpu	17
5.3.2.8	accfft_local_size_dft_r2c_gpu	17
5.3.2.9	accfft_plan_dft_3d_c2c_gpu	17
5.3.2.10	accfft_plan_dft_3d_r2c_gpu	18
5.4	/Users/new/accfft/src/accfft_gpuf.cpp File Reference	18
5.4.1	Detailed Description	19
5.4.2	Function Documentation	19
5.4.2.1	accfft_cleanup_gpuf	19
5.4.2.2	accfft_destroy_plan	19
5.4.2.3	accfft_destroy_plan_gpu	19
5.4.2.4	accfft_execute_c2c_gpuf	19
5.4.2.5	accfft_execute_c2r_gpuf	20
5.4.2.6	accfft_execute_r2c_gpuf	20
5.4.2.7	accfft_local_size_dft_c2c_gpuf	20
5.4.2.8	accfft_local_size_dft_r2c_gpuf	21
5.4.2.9	accfft_plan_dft_3d_c2c_gpuf	21
5.4.2.10	accfft_plan_dft_3d_r2c_gpuf	21
5.5	/Users/new/accfft/src/accfft.cpp File Reference	22
5.5.1	Detailed Description	22
5.5.2	Function Documentation	22
5.5.2.1	accfft_destroy_plan	22
5.5.2.2	accfft_execute_c2cf	23
5.5.2.3	accfft_execute_c2rf	23
5.5.2.4	accfft_execute_r2cf	23
5.5.2.5	accfft_local_size_dft_c2cf	24
5.5.2.6	accfft_local_size_dft_r2cf	24
5.5.2.7	accfft_plan_dft_3d_c2cf	24
5.5.2.8	accfft_plan_dft_3d_r2cf	25
5.6	/Users/new/accfft/src/operators.cpp File Reference	25
5.6.1	Detailed Description	26
5.6.2	Function Documentation	26
5.6.2.1	accfft_biharmonic	26
5.6.2.2	accfft_divergence	26
5.6.2.3	accfft_grad	26

5.6.2.4	accfft_inv_biharmonic	27
5.6.2.5	accfft_inv_laplace	27
5.6.2.6	accfft_laplace	27
5.6.2.7	read_pnetcdf	27
5.6.2.8	write_pnetcdf	28
5.7	/Users/new/accfft/src/operators_gpu.cpp File Reference	28
5.7.1	Detailed Description	28
5.7.2	Function Documentation	28
5.7.2.1	accfft_biharmonic_gpu	28
5.7.2.2	accfft_divergence_gpu	29
5.7.2.3	accfft_grad_gpu	29
5.7.2.4	accfft_laplace_gpu	29
5.8	/Users/new/accfft/src/operators_gpuf.cpp File Reference	30
5.8.1	Detailed Description	30
5.8.2	Function Documentation	30
5.8.2.1	accfft_biharmonic_gpuf	30
5.8.2.2	accfft_divergence_gpuf	30
5.8.2.3	accfft_grad_gpuf	31
5.8.2.4	accfft_laplace_gpuf	31
5.9	/Users/new/accfft/src/operatorsf.cpp File Reference	32
5.9.1	Detailed Description	32
5.9.2	Function Documentation	32
5.9.2.1	accfft_biharmonicf	32
5.9.2.2	accfft_divergencef	32
5.9.2.3	accfft_gradf	33
5.9.2.4	accfft_laplacef	33
5.9.2.5	read_pnetcdf	33
5.9.2.6	write_pnetcdf	34
	Index	35

Chapter 1

My Main Page

To access the manual for each function click on the *Files* tab above. You can also download a pdf version of the document [here](#).

Chapter 2

Timing AccFFT

AccFFT *execute* functions get a double timer of size 5, where the timing for different parts of the algorithm is written to:

1. timer[2]: Comm time
2. timer[4]: Local FFT execution time.
3. timer[0],timer[1],timer[3]: Scratch data

None of the entries in timer denote the total time. To get the total execution time, you should wrap the execution function call with `MPI_Wtime()`. For example:

```
double exec_time=0; exec_time-=MPI_Wtime(); accfft_execute(...,timer); exec_time+=MPI_Wtime();
```

It is recommended that you perform 1-2 warmup runs by calling the corresponding *execute* function, before profiling the code.

Chapter 3

AccFFT Flags

Flags are used to tune the library during the setup time to use the best configuration. Currently two flags are supported:

1. ACCFFT_ESTIMATE: Minimal/no tuning. This option does not over write the input array data.
2. ACCFFT_MEASURE: Tunes for local FFT execution algorithm, as well as global transposes to reduce communication time. **Note that with this flag, the setup will overwrite the input array data to perform the tunings.**

Chapter 4

File Index

4.1 File List

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

/Users/new/accfft/src/accfft.cpp	9
/Users/new/accfft/src/accfft_common.cpp	12
/Users/new/accfft/src/accfft_gpu.cpp	14
/Users/new/accfft/src/accfft_gpuf.cpp	18
/Users/new/accfft/src/accfftf.cpp	22
/Users/new/accfft/src/operators.cpp	25
/Users/new/accfft/src/operators_gpu.cpp	28
/Users/new/accfft/src/operators_gpuf.cpp	30
/Users/new/accfft/src/operatorsf.cpp	32

Chapter 5

File Documentation

5.1 /Users/new/accfft/src/accfft.cpp File Reference

```
#include <mpi.h>
#include <fftw3.h>
#include <omp.h>
#include <iostream>
#include <cmath>
#include <math.h>
#include "transpose.h"
#include <string.h>
#include "accfft.h"
#include "accfft_common.h"
```

Functions

- int [accfft_local_size_dft_r2c](#) (int *n, int *isize, int *istart, int *osize, int *ostart, MPI_Comm c_comm)
- accfft_plan * [accfft_plan_dft_3d_r2c](#) (int *n, double *data, double *data_out, MPI_Comm c_comm, unsigned flags)
- int [accfft_local_size_dft_c2c](#) (int *n, int *isize, int *istart, int *osize, int *ostart, MPI_Comm c_comm)
- accfft_plan * [accfft_plan_dft_3d_c2c](#) (int *n, Complex *data, Complex *data_out, MPI_Comm c_comm, unsigned flags)
- void [accfft_execute_r2c](#) (accfft_plan *plan, double *data, Complex *data_out, double *timer, std::bitset< 3 > XYZ)
- void [accfft_execute_c2r](#) (accfft_plan *plan, Complex *data, double *data_out, double *timer, std::bitset< 3 > XYZ)
- void [accfft_execute_c2c](#) (accfft_plan *plan, int direction, Complex *data, Complex *data_out, double *timer, std::bitset< 3 > XYZ)
- void [accfft_destroy_plan](#) (accfft_plan *plan)

5.1.1 Detailed Description

CPU functions of AccFFT

5.1.2 Function Documentation

5.1.2.1 void [accfft_destroy_plan](#) (accfft_plan * *plan*)

Destroy AccFFT CPU plan.

Parameters

<i>plan</i>	Input plan to be destroyed.
-------------	-----------------------------

5.1.2.2 void `accfft_execute_c2c` (`accfft_plan * plan`, int `direction`, Complex * `data`, Complex * `data_out`, double * `timer`, std::bitset< 3 > `XYZ`)

Execute C2C plan. This function is blocking and only returns after the transform is completed.

Note

For inplace transforms, `data_out` should point to the same memory address as `data`, AND the plan must have been created as inplace.

Parameters

<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c .
<i>data</i>	Input data in frequency domain.
<i>data_out</i>	Output data in frequency domain.
<i>timer</i>	See Timing AccFFT for more details.
<i>XYZ</i>	a bit set field that determines which directions FFT should be executed

5.1.2.3 void `accfft_execute_c2r` (`accfft_plan * plan`, Complex * `data`, double * `data_out`, double * `timer`, std::bitset< 3 > `XYZ`)

Execute C2R plan. This function is blocking and only returns after the transform is completed.

Note

For inplace transform, `data_out` should point to the same memory address as `data`, AND the plan must have been created as inplace.

Parameters

<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c .
<i>data</i>	Input data in frequency domain.
<i>data_out</i>	Output data in frequency domain.
<i>timer</i>	See Timing AccFFT for more details.
<i>XYZ</i>	a bit set field that determines which directions FFT should be executed

5.1.2.4 void `accfft_execute_r2c` (`accfft_plan * plan`, double * `data`, Complex * `data_out`, double * `timer`, std::bitset< 3 > `XYZ`)

Execute R2C plan. This function is blocking and only returns after the transform is completed.

Note

For inplace transforms, `data_out` should point to the same memory address as `data`, AND the plan must have been created as inplace.

Parameters

<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c .
<i>data</i>	Input data in spatial domain.
<i>data_out</i>	Output data in frequency domain.
<i>timer</i>	See Timing AccFFT for more details.
<i>XYZ</i>	a bit set field that determines which directions FFT should be executed

5.1.2.5 `int accfft_local_size_dft_c2c (int * n, int * isize, int * istart, int * osize, int * ostart, MPI_Comm c_comm)`

Get the local sizes of the distributed global data for a C2C transform

Parameters

<i>n</i>	Integer array of size 3, corresponding to the global data size
<i>isize</i>	The size of the data that is locally distributed to the calling process
<i>istart</i>	The starting index of the data that locally resides on the calling process
<i>osize</i>	The output size of the data that locally resides on the calling process, after the C2C transform is finished
<i>ostart</i>	The output starting index of the data that locally resides on the calling process, after the R2C transform is finished
<i>c_comm</i>	Cartesian communicator returned by accfft_create_comm

Returns

5.1.2.6 `int accfft_local_size_dft_r2c (int * n, int * isize, int * istart, int * osize, int * ostart, MPI_Comm c_comm)`

Get the local sizes of the distributed global data for a R2C transform

Parameters

<i>n</i>	Integer array of size 3, corresponding to the global data size
<i>isize</i>	The size of the data that is locally distributed to the calling process
<i>istart</i>	The starting index of the data that locally resides on the calling process
<i>osize</i>	The output size of the data that locally resides on the calling process, after the R2C transform is finished
<i>ostart</i>	The output starting index of the data that locally resides on the calling process, after the R2C transform is finished
<i>c_comm</i>	Cartesian communicator returned by accfft_create_comm

Returns

5.1.2.7 `accfft_plan* accfft_plan_dft_3d_c2c (int * n, Complex * data, Complex * data_out, MPI_Comm c_comm, unsigned flags)`

Creates a 3D C2C parallel FFT plan. If *data_out* point to the same location as the input data, then an inplace plan will be created. Otherwise the plan would be outplace.

Parameters

<i>n</i>	Integer array of size 3, corresponding to the global data size
<i>data</i>	Input data in spatial domain
<i>data_out</i>	Output data in frequency domain
<i>c_comm</i>	Cartesian communicator returned by accfft_create_comm
<i>flags</i>	AccFFT flags, See AccFFT Flags for more details.

Returns

5.1.2.8 `accfft_plan* accfft_plan_dft_3d_r2c (int * n, double * data, double * data_out, MPI_Comm c_comm, unsigned flags)`

Creates a 3D R2C parallel FFT plan. If `data_out` point to the same location as the input data, then an inplace plan will be created. Otherwise the plan would be outplace.

Parameters

<i>n</i>	Integer array of size 3, corresponding to the global data size
<i>data</i>	Input data in spatial domain
<i>data_out</i>	Output data in frequency domain
<i>c_comm</i>	Cartesian communicator returned by accfft_create_comm
<i>flags</i>	AccFFT flags, See AccFFT Flags for more details.

Returns

5.2 /Users/new/accfft/src/accfft_common.cpp File Reference

```
#include <mpi.h>
#include <omp.h>
#include <iostream>
#include <cmath>
#include <math.h>
#include "accfft_common.h"
```

Functions

- void * [accfft_alloc](#) (ptrdiff_t size)
- void [accfft_free](#) (void *ptr)
- void [accfft_create_comm](#) (MPI_Comm in_comm, int *c_dims, MPI_Comm *c_comm)
- int [accfft_init](#) ()
- int [accfft_init](#) (int nthreads)
- void [accfft_cleanup](#) ()

5.2.1 Detailed Description

common functions in AccFFT.

5.2.2 Function Documentation

5.2.2.1 void* accfft_alloc (ptrdiff_t size)

Allocates aligned memory to enable SIMD

Parameters

<i>size</i>	Allocation size in Bytes
-------------	--------------------------

5.2.2.2 void accfft_cleanup ()

Cleanup all CPU resources

5.2.2.3 void accfft_create_comm (MPI_Comm *in_comm*, int * *c_dims*, MPI_Comm * *c_comm*)

Creates a Cartesian communicator of size *c_dims*[0]x*c_dims*[1] from its input. If *c_dims*[0]x*c_dims*[1] would not match the size of *in_comm*, then the function prints an error and automatically sets *c_dims* to the correct values.

Parameters

<i>in_comm</i>	Input MPI communicator handle
<i>c_dims</i>	A 2D integer array, which sets the size of the Cartesian array to <i>c_dims</i> [0]x <i>c_dims</i> [1]
<i>c_comm</i>	A pointer to the Cartesian communicator which will be created

5.2.2.4 void accfft_free (void * *ptr*)

Free memory allocated by [accfft_alloc](#)

Parameters

<i>ptr</i>	Address of the memory to be freed.
------------	------------------------------------

5.2.2.5 int accfft_init ()

Initialize AccFFT library.

Returns

0 if successful.

5.2.2.6 int accfft_init (int *nthreads*)

Initializes the library.

Parameters

<i>nthreads</i>	The number of OpenMP threads to use for execution of local FFT.
-----------------	---

Returns

0 if successful

5.3 /Users/new/accfft/src/accfft_gpu.cpp File Reference

```
#include "accfft_gpu.h"
```

```

#include <mpi.h>
#include <omp.h>
#include <iostream>
#include <cmath>
#include <math.h>
#include "transpose_cuda.h"
#include <cuda_runtime_api.h>
#include <string.h>
#include <cuda.h>
#include <cufft.h>
#include "accfft_common.h"

```

Functions

- void [accfft_cleanup_gpu](#) ()
- int [accfft_local_size_dft_r2c_gpu](#) (int *n, int *isize, int *istart, int *osize, int *ostart, MPI_Comm c_comm, bool inplace)
- [accfft_plan_gpu](#) * [accfft_plan_dft_3d_r2c_gpu](#) (int *n, double *data_d, double *data_out_d, MPI_Comm c_comm, unsigned flags)
- void [accfft_execute_r2c_gpu](#) ([accfft_plan_gpu](#) *plan, double *data, Complex *data_out, double *timer, std::bitset< 3 > xyz)
- void [accfft_execute_c2r_gpu](#) ([accfft_plan_gpu](#) *plan, Complex *data, double *data_out, double *timer, std::bitset< 3 > xyz)
- int [accfft_local_size_dft_c2c_gpu](#) (int *n, int *isize, int *istart, int *osize, int *ostart, MPI_Comm c_comm)
- [accfft_plan_gpu](#) * [accfft_plan_dft_3d_c2c_gpu](#) (int *n, Complex *data_d, Complex *data_out_d, MPI_Comm c_comm, unsigned flags)
- void [accfft_execute_c2c_gpu](#) ([accfft_plan_gpu](#) *plan, int direction, Complex *data_d, Complex *data_out_d, double *timer, std::bitset< 3 > xyz)
- void [accfft_destroy_plan](#) ([accfft_plan_gpu](#) *plan)
- void [accfft_destroy_plan_gpu](#) ([accfft_plan_gpu](#) *plan)

5.3.1 Detailed Description

GPU functions of AccFFT

5.3.2 Function Documentation

5.3.2.1 void [accfft_cleanup_gpu](#) ()

Cleanup all CPU resources

5.3.2.2 void [accfft_destroy_plan](#) ([accfft_plan_gpu](#) * *plan*)

Destroy AccFFT CPU plan. This function calls [accfft_destroy_plan_gpu](#).

Parameters

<i>plan</i>	Input plan to be destroyed.
-------------	-----------------------------

5.3.2.3 void [accfft_destroy_plan_gpu](#) ([accfft_plan_gpu](#) * *plan*)

Destroy AccFFT GPU plan.

Parameters

<i>plan</i>	Input plan to be destroyed.
-------------	-----------------------------

```
5.3.2.4 void accfft_execute_c2c_gpu ( accfft_plan_gpu * plan, int direction, Complex * data_d, Complex * data_out_d,
double * timer, std::bitset< 3 > xyz )
```

Execute C2C plan. This function is blocking and only returns after the transform is completed.

Note

For inplace transforms, *data_out* should point to the same memory address as *data*, AND the plan must have been created as inplace.

Parameters

<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c .
<i>data</i>	Input data in frequency domain.
<i>data_out</i>	Output data in frequency domain.
<i>timer</i>	See Timing AccFFT for more details.
XYZ	a bit set field that determines which directions FFT should be executed

```
5.3.2.5 void accfft_execute_c2r_gpu ( accfft_plan_gpu * plan, Complex * data, double * data_out, double * timer,
std::bitset< 3 > xyz )
```

Execute C2R plan. This function is blocking and only returns after the transform is completed.

Note

For inplace transforms, *data_out* should point to the same memory address as *data*, AND the plan must have been created as inplace.

Parameters

<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c .
<i>data</i>	Input data in frequency domain.
<i>data_out</i>	Output data in frequency domain.
<i>timer</i>	See Timing AccFFT for more details.
XYZ	a bit set field that determines which directions FFT should be executed

```
5.3.2.6 void accfft_execute_r2c_gpu ( accfft_plan_gpu * plan, double * data, Complex * data_out, double * timer,
std::bitset< 3 > xyz )
```

Execute R2C plan. This function is blocking and only returns after the transform is completed.

Note

For inplace transforms, *data_out* should point to the same memory address as *data*, AND the plan must have been created as inplace.

Parameters

<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c .
<i>data</i>	Input data in spatial domain.
<i>data_out</i>	Output data in frequency domain.
<i>timer</i>	See Timing AccFFT for more details.
<i>XYZ</i>	a bit set field that determines which directions FFT should be executed

5.3.2.7 `int accfft_local_size_dft_c2c_gpu (int * n, int * isize, int * istart, int * osize, int * ostart, MPI_Comm c_comm)`

Get the local sizes of the distributed global data for a GPU C2C transform

Parameters

<i>n</i>	Integer array of size 3, corresponding to the global data size
<i>isize</i>	The size of the data that is locally distributed to the calling process
<i>istart</i>	The starting index of the data that locally resides on the calling process
<i>osize</i>	The output size of the data that locally resides on the calling process, after the C2C transform is finished
<i>ostart</i>	The output starting index of the data that locally resides on the calling process, after the R2C transform is finished
<i>c_comm</i>	Cartesian communicator returned by accfft_create_comm

Returns

5.3.2.8 `int accfft_local_size_dft_r2c_gpu (int * n, int * isize, int * istart, int * osize, int * ostart, MPI_Comm c_comm, bool inplace)`

Get the local sizes of the distributed global data for a GPU R2C transform

Parameters

<i>n</i>	Integer array of size 3, corresponding to the global data size
<i>isize</i>	The size of the data that is locally distributed to the calling process
<i>istart</i>	The starting index of the data that locally resides on the calling process
<i>osize</i>	The output size of the data that locally resides on the calling process, after the R2C transform is finished
<i>ostart</i>	The output starting index of the data that locally resides on the calling process, after the R2C transform is finished
<i>c_comm</i>	Cartesian communicator returned by accfft_create_comm

Returns

5.3.2.9 `accfft_plan_gpu* accfft_plan_dft_3d_c2c_gpu (int * n, Complex * data_d, Complex * data_out_d, MPI_Comm c_comm, unsigned flags)`

Creates a 3D C2C parallel FFT plan. If `data_out` point to the same location as the input data, then an inplace plan will be created. Otherwise the plan would be outplace.

Parameters

<i>n</i>	Integer array of size 3, corresponding to the global data size
<i>data</i>	Input data in spatial domain
<i>data_out</i>	Output data in frequency domain
<i>c_comm</i>	Cartesian communicator returned by accfft_create_comm
<i>flags</i>	AccFFT flags, See AccFFT Flags for more details.

Returns

5.3.2.10 `accfft_plan_gpu* accfft_plan_dft_3d_r2c_gpu (int * n, double * data_d, double * data_out_d, MPI_Comm c_comm, unsigned flags)`

Creates a 3D R2C parallel FFT plan. If `data_out` point to the same location as the input data, then an inplace plan will be created. Otherwise the plan would be outplace.

Parameters

<i>n</i>	Integer array of size 3, corresponding to the global data size
<i>data</i>	Input data in spatial domain
<i>data_out</i>	Output data in frequency domain
<i>c_comm</i>	Cartesian communicator returned by accfft_create_comm
<i>flags</i>	AccFFT flags, See AccFFT Flags for more details.

Returns

5.4 /Users/new/accfft/src/accfft_gpuf.cpp File Reference

```
#include "accfft_gpuf.h"
#include <mpi.h>
#include <omp.h>
#include <iostream>
#include <cmath>
#include <math.h>
#include "transpose_cuda.h"
#include <cuda_runtime_api.h>
#include <string.h>
#include <cuda.h>
#include <cufft.h>
#include "accfft_common.h"
```

Functions

- void [accfft_cleanup_gpuf](#) ()
- int [accfft_local_size_dft_r2c_gpuf](#) (int *n, int *isize, int *istart, int *osize, int *ostart, MPI_Comm c_comm, bool inplace)
- `accfft_plan_gpuf * accfft_plan_dft_3d_r2c_gpuf` (int *n, float *data_d, float *data_out_d, MPI_Comm c_↵comm, unsigned flags)
- void [accfft_execute_r2c_gpuf](#) (accfft_plan_gpuf *plan, float *data, Complexf *data_out, double *timer, std_↵::bitset< 3 > xyz)

- void [accfft_execute_c2r_gpuf](#) (accfft_plan_gpuf *plan, Complexf *data, float *data_out, double *timer, std::bitset< 3 > xyz)
- int [accfft_local_size_dft_c2c_gpuf](#) (int *n, int *isize, int *istart, int *osize, int *ostart, MPI_Comm c_comm)
- accfft_plan_gpuf * [accfft_plan_dft_3d_c2c_gpuf](#) (int *n, Complexf *data_d, Complexf *data_out_d, MPI_Comm c_comm, unsigned flags)
- void [accfft_execute_c2c_gpuf](#) (accfft_plan_gpuf *plan, int direction, Complexf *data_d, Complexf *data_out_d, double *timer, std::bitset< 3 > xyz)
- void [accfft_destroy_plan](#) (accfft_plan_gpuf *plan)
- void [accfft_destroy_plan_gpu](#) (accfft_plan_gpuf *plan)

5.4.1 Detailed Description

Single precision GPU functions of AccFFT

5.4.2 Function Documentation

5.4.2.1 void accfft_cleanup_gpuf ()

Cleanup all CPU resources

5.4.2.2 void accfft_destroy_plan (accfft_plan_gpuf * plan)

Destroy single precision AccFFT CPU plan. This function calls [accfft_destroy_plan_gpu](#).

Parameters

<i>plan</i>	Input plan to be destroyed.
-------------	-----------------------------

5.4.2.3 void accfft_destroy_plan_gpu (accfft_plan_gpuf * plan)

Destroy single precision AccFFT GPU plan.

Parameters

<i>plan</i>	Input plan to be destroyed.
-------------	-----------------------------

5.4.2.4 void accfft_execute_c2c_gpuf (accfft_plan_gpuf * plan, int direction, Complexf * data_d, Complexf * data_out_d, double * timer, std::bitset< 3 > xyz)

Execute single precision C2C plan. This function is blocking and only returns after the transform is completed.

Note

For inplace transforms, data_out should point to the same memory address as data, AND the plan must have been created as inplace.

Parameters

<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2cf .
<i>data</i>	Input data in frequency domain.

<i>data_out</i>	Output data in frequency domain.
<i>timer</i>	See Timing AccFFT for more details.
<i>XYZ</i>	a bit set field that determines which directions FFT should be executed

5.4.2.5 `void accfft_execute_c2r_gpuf (accfft_plan_gpuf * plan, Complex * data, float * data_out, double * timer, std::bitset<3> xyz)`

Execute single precision C2R plan. This function is blocking and only returns after the transform is completed.

Note

For inplace transforms, *data_out* should point to the same memory address as *data*, AND the plan must have been created as inplace.

Parameters

<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2cf .
<i>data</i>	Input data in frequency domain.
<i>data_out</i>	Output data in frequency domain.
<i>timer</i>	See Timing AccFFT for more details.
<i>XYZ</i>	a bit set field that determines which directions FFT should be executed

5.4.2.6 `void accfft_execute_r2c_gpuf (accfft_plan_gpuf * plan, float * data, Complex * data_out, double * timer, std::bitset<3> xyz)`

Execute single precision R2C plan. This function is blocking and only returns after the transform is completed.

Note

For inplace transforms, *data_out* should point to the same memory address as *data*, AND the plan must have been created as inplace.

Parameters

<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2cf .
<i>data</i>	Input data in spatial domain.
<i>data_out</i>	Output data in frequency domain.
<i>timer</i>	See Timing AccFFT for more details.
<i>XYZ</i>	a bit set field that determines which directions FFT should be executed

5.4.2.7 `int accfft_local_size_dft_c2c_gpuf (int * n, int * isize, int * istart, int * osize, int * ostart, MPI_Comm c_comm)`

Get the local sizes of the distributed global data for a GPU single precision C2C transform

Parameters

<i>n</i>	Integer array of size 3, corresponding to the global data size
<i>isize</i>	The size of the data that is locally distributed to the calling process
<i>istart</i>	The starting index of the data that locally resides on the calling process
<i>osize</i>	The output size of the data that locally resides on the calling process, after the C2C transform is finished

<i>ostart</i>	The output starting index of the data that locally resides on the calling process, after the R2C transform is finished
<i>c_comm</i>	Cartesian communicator returned by accfft_create_comm

Returns

5.4.2.8 `int accfft_local_size_dft_r2c_gpuf (int * n, int * isize, int * istart, int * osize, int * ostart, MPI_Comm c_comm, bool inplace)`

Get the local sizes of the distributed global data for a GPU single precision R2C transform

Parameters

<i>n</i>	Integer array of size 3, corresponding to the global data size
<i>isize</i>	The size of the data that is locally distributed to the calling process
<i>istart</i>	The starting index of the data that locally resides on the calling process
<i>osize</i>	The output size of the data that locally resides on the calling process, after the R2C transform is finished
<i>ostart</i>	The output starting index of the data that locally resides on the calling process, after the R2C transform is finished
<i>c_comm</i>	Cartesian communicator returned by accfft_create_comm

Returns

5.4.2.9 `accfft_plan_gpuf* accfft_plan_dft_3d_c2c_gpuf (int * n, Complex * data_d, Complex * data_out_d, MPI_Comm c_comm, unsigned flags)`

Creates a 3D single precision C2C parallel FFT plan. If data_out point to the same location as the input data, then an inplace plan will be created. Otherwise the plan would be outplace.

Parameters

<i>n</i>	Integer array of size 3, corresponding to the global data size
<i>data</i>	Input data in spatial domain
<i>data_out</i>	Output data in frequency domain
<i>c_comm</i>	Cartesian communicator returned by accfft_create_comm
<i>flags</i>	AccFFT flags, See AccFFT Flags for more details.

Returns

5.4.2.10 `accfft_plan_gpuf* accfft_plan_dft_3d_r2c_gpuf (int * n, float * data_d, float * data_out_d, MPI_Comm c_comm, unsigned flags)`

Creates a 3D single precision R2C parallel FFT plan. If data_out point to the same location as the input data, then an inplace plan will be created. Otherwise the plan would be outplace.

Parameters

<i>n</i>	Integer array of size 3, corresponding to the global data size
<i>data</i>	Input data in spatial domain
<i>data_out</i>	Output data in frequency domain
<i>c_comm</i>	Cartesian communicator returned by accfft_create_comm
<i>flags</i>	AccFFT flags, See AccFFT Flags for more details.

Returns

5.5 /Users/new/accfft/src/accfft.cpp File Reference

```
#include <mpi.h>
#include <fftw3.h>
#include <omp.h>
#include <iostream>
#include <cmath>
#include <math.h>
#include "transpose.h"
#include <string.h>
#include "accfftf.h"
#include "accfft_common.h"
```

Functions

- int [accfft_local_size_dft_r2cf](#) (int *n, int *isize, int *istart, int *osize, int *ostart, MPI_Comm c_comm)
- accfft_planf * [accfft_plan_dft_3d_r2cf](#) (int *n, float *data, float *data_out, MPI_Comm c_comm, unsigned flags)
- int [accfft_local_size_dft_c2cf](#) (int *n, int *isize, int *istart, int *osize, int *ostart, MPI_Comm c_comm)
- accfft_planf * [accfft_plan_dft_3d_c2cf](#) (int *n, Complexf *data, Complexf *data_out, MPI_Comm c_comm, unsigned flags)
- void [accfft_execute_r2cf](#) (accfft_planf *plan, float *data, Complexf *data_out, double *timer, std::bitset< 3 > XYZ)
- void [accfft_execute_c2rf](#) (accfft_planf *plan, Complexf *data, float *data_out, double *timer, std::bitset< 3 > XYZ)
- void [accfft_execute_c2cf](#) (accfft_planf *plan, int direction, Complexf *data, Complexf *data_out, double *timer, std::bitset< 3 > XYZ)
- void [accfft_destroy_plan](#) (accfft_planf *plan)

5.5.1 Detailed Description

Single precision CPU functions of AccFFT

5.5.2 Function Documentation

5.5.2.1 void [accfft_destroy_plan](#) (accfft_planf * plan)

Destroy single precision AccFFT CPU plan.

Parameters

<i>plan</i>	Input plan to be destroyed.
-------------	-----------------------------

5.5.2.2 `void accfft_execute_c2cf (accfft_planf * plan, int direction, Complex * data, Complex * data_out, double * timer, std::bitset< 3 > XYZ)`

Execute single precision C2C plan. This function is blocking and only returns after the transform is completed.

Note

For inplace transforms, `data_out` should point to the same memory address as `data`, AND the plan must have been created as inplace.

Parameters

<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2cf .
<i>data</i>	Input data in frequency domain.
<i>data_out</i>	Output data in frequency domain.
<i>timer</i>	See Timing AccFFT for more details.
<i>XYZ</i>	a bit set field that determines which directions FFT should be executed

5.5.2.3 `void accfft_execute_c2rf (accfft_planf * plan, Complex * data, float * data_out, double * timer, std::bitset< 3 > XYZ)`

Execute single precision C2R plan. This function is blocking and only returns after the transform is completed.

Note

For inplace transform, `data_out` should point to the same memory address as `data`, AND the plan must have been created as inplace.

Parameters

<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2cf .
<i>data</i>	Input data in frequency domain.
<i>data_out</i>	Output data in frequency domain.
<i>timer</i>	See Timing AccFFT for more details.
<i>XYZ</i>	a bit set field that determines which directions FFT should be executed

5.5.2.4 `void accfft_execute_r2cf (accfft_planf * plan, float * data, Complex * data_out, double * timer, std::bitset< 3 > XYZ)`

Execute single precision R2C plan. This function is blocking and only returns after the transform is completed.

Note

For inplace transforms, `data_out` should point to the same memory address as `data`, AND the plan must have been created as inplace.

Parameters

<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2cf .
<i>data</i>	Input data in spatial domain.
<i>data_out</i>	Output data in frequency domain.
<i>timer</i>	See Timing AccFFT for more details.
<i>XYZ</i>	a bit set field that determines which directions FFT should be executed

5.5.2.5 `int accfft_local_size_dft_c2cf (int * n, int * isize, int * istart, int * osize, int * ostart, MPI_Comm c_comm)`

Get the local sizes of the distributed global data for a C2C single precision transform

Parameters

<i>n</i>	Integer array of size 3, corresponding to the global data size
<i>isize</i>	The size of the data that is locally distributed to the calling process
<i>istart</i>	The starting index of the data that locally resides on the calling process
<i>osize</i>	The output size of the data that locally resides on the calling process, after the C2C transform is finished
<i>ostart</i>	The output starting index of the data that locally resides on the calling process, after the R2C transform is finished
<i>c_comm</i>	Cartesian communicator returned by accfft_create_comm

Returns

5.5.2.6 `int accfft_local_size_dft_r2cf (int * n, int * isize, int * istart, int * osize, int * ostart, MPI_Comm c_comm)`

Get the local sizes of the distributed global data for a R2C single precision transform

Parameters

<i>n</i>	Integer array of size 3, corresponding to the global data size
<i>isize</i>	The size of the data that is locally distributed to the calling process
<i>istart</i>	The starting index of the data that locally resides on the calling process
<i>osize</i>	The output size of the data that locally resides on the calling process, after the R2C transform is finished
<i>ostart</i>	The output starting index of the data that locally resides on the calling process, after the R2C transform is finished
<i>c_comm</i>	Cartesian communicator returned by accfft_create_comm

Returns

5.5.2.7 `accfft_plan* accfft_plan_dft_3d_c2cf (int * n, Complex * data, Complex * data_out, MPI_Comm c_comm, unsigned flags)`

Creates a 3D C2C single precision parallel FFT plan. If *data_out* point to the same location as the input data, then an inplace plan will be created. Otherwise the plan would be outplace.

Parameters

<i>n</i>	Integer array of size 3, corresponding to the global data size
<i>data</i>	Input data in spatial domain
<i>data_out</i>	Output data in frequency domain
<i>c_comm</i>	Cartesian communicator returned by accfft_create_comm
<i>flags</i>	AccFFT flags, See AccFFT Flags for more details.

Returns**5.5.2.8** `accfft_plan* accfft_plan_dft_3d_r2cf (int * n, float * data, float * data_out, MPI_Comm c_comm, unsigned flags)`

Creates a 3D R2C single precision parallel FFT plan. If *data_out* point to the same location as the input data, then an inplace plan will be created. Otherwise the plan would be outplace.

Parameters

<i>n</i>	Integer array of size 3, corresponding to the global data size
<i>data</i>	Input data in spatial domain
<i>data_out</i>	Output data in frequency domain
<i>c_comm</i>	Cartesian communicator returned by accfft_create_comm
<i>flags</i>	AccFFT flags, See AccFFT Flags for more details.

Returns**5.6 /Users/new/accfft/src/operators.cpp File Reference**

```
#include <mpi.h>
#include <omp.h>
#include <iostream>
#include <cmath>
#include <math.h>
#include <string.h>
#include <accfft.h>
#include <../src/operators.txx>
#include <pnetcdf.h>
#include <cstdlib>
#include <string>
```

Functions

- void [accfft_grad](#) (double *A_x, double *A_y, double *A_z, double *A, accfft_plan *plan, std::bitset< 3 > *pXYZ, double *timer)
- void [accfft_laplace](#) (double *LA, double *A, accfft_plan *plan, double *timer)
- void [accfft_divergence](#) (double *divA, double *A_x, double *A_y, double *A_z, accfft_plan *plan, double *timer)
- void [accfft_biharmonic](#) (double *BA, double *A, accfft_plan *plan, double *timer)
- void [accfft_inv_laplace](#) (double *invLA, double *A, accfft_plan *plan, double *timer)
- void [accfft_inv_biharmonic](#) (double *invBA, double *A, accfft_plan *plan, double *timer)
- void [read_pnetcdf](#) (const std::string &filename, MPI_Offset starts[3], MPI_Offset counts[3], int gsizes[3], double *localData)
- void [write_pnetcdf](#) (const std::string &filename, MPI_Offset starts[3], MPI_Offset counts[3], int gsizes[3], double *localData)

5.6.1 Detailed Description

CPU functions of AccFFT operators

5.6.2 Function Documentation

5.6.2.1 `void accfft_biharmonic (double * BA, double * A, accfft_plan * plan, double * timer)`

Computes double precision biharmonic of its input real data A, and writes the output into LA.

Parameters

<i>BA</i>	$\Delta^2 A$
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>timer</i>	See Timing AccFFT for more details.

5.6.2.2 `void accfft_divergence (double * divA, double * A_x, double * A_y, double * A_z, accfft_plan * plan, double * timer)`

Computes double precision divergence of its input vector data A_x, A_y, and A_z. The output data is written to divA.

Parameters

<i>divA</i>	$\nabla \cdot (A_x i + A_y j + A_z k)$
<i>A_x</i>	The x component of ∇A
<i>A_y</i>	The y component of ∇A
<i>A_z</i>	The z component of ∇A
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>timer</i>	See Timing AccFFT for more details.

5.6.2.3 `void accfft_grad (double * A_x, double * A_y, double * A_z, double * A, accfft_plan * plan, std::bitset<3> * pXYZ, double * timer)`

Computes double precision gradient of its input real data A, and returns the x, y, and z components and writes the output into A_x, A_y, and A_z respectively.

Parameters

<i>A_x</i>	The x component of ∇A
<i>A_y</i>	The y component of ∇A
<i>A_z</i>	The z component of ∇A
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>pXYZ</i>	a bit set pointer field of size 3 that determines which gradient components are needed. If XYZ={111} then all the components are computed and if XYZ={100}, then only the x component is computed. This can save the user some time, when just one or two of the gradient components are needed.

<i>timer</i>	See Timing AccFFT for more details.
--------------	---

5.6.2.4 void accfft_inv_biharmonic (double * *invBA*, double * *A*, accfft_plan * *plan*, double * *timer*)

Computes double precision inverse biharmonic of its input real data *A*, and writes the output into *invBA*.

Parameters

<i>invBA</i>	$\Delta^{-2} A$
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>timer</i>	See Timing AccFFT for more details.

5.6.2.5 void accfft_inv_laplace (double * *invLA*, double * *A*, accfft_plan * *plan*, double * *timer*)

Computes double precision inverse Laplacian of its input real data *A*, and writes the output into *invLA*.

Parameters

<i>invLA</i>	$\Delta^{-1} A$
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>timer</i>	See Timing AccFFT for more details.

5.6.2.6 void accfft_laplace (double * *LA*, double * *A*, accfft_plan * *plan*, double * *timer*)

Computes double precision Laplacian of its input real data *A*, and writes the output into *LA*.

Parameters

<i>LA</i>	ΔA
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>timer</i>	See Timing AccFFT for more details.

5.6.2.7 void read_pnetcdf (const std::string & *filename*, MPI_Offset *starts*[3], MPI_Offset *counts*[3], int *gsizes*[3], double * *localData*)

Read a parallel-netcdf file.

We assume here that *localData* is a scalar.

Pnetcdf uses row-major format (same as FFTW).

Parameters

in	<i>filename</i>	: PnetCDF filename
in	<i>starts</i>	: offset to where to start reading data
in	<i>counts</i>	: number of elements read (3D sub-domain inside global)
in	<i>gsizes</i>	: global sizes
out	<i>localData</i>	: actual data buffer (size : $nx*ny*nz*sizeof(double)$)

localData must have been allocated prior to calling this routine.

5.6.2.8 void write_pnetcdf (const std::string & *filename*, MPI_Offset *starts*[3], MPI_Offset *counts*[3], int *gsizes*[3], double * *localData*)

Write a parallel-nedcdf file.

We assume here that *localData* is a scalar.

Pnetcdf uses row-major format (same as FFTW).

Parameters

in	<i>filename</i>	: PnetCDF filename
in	<i>starts</i>	: offset to where to start reading data
in	<i>counts</i>	: number of elements read (3D sub-domain inside global)
in	<i>gsizes</i>	: global sizes
in	<i>localData</i>	: actual data buffer (size : nx*ny*nz*sizeof(double))

5.7 /Users/new/accfft/src/operators_gpu.cpp File Reference

```
#include <mpi.h>
#include <omp.h>
#include <iostream>
#include <cmath>
#include <math.h>
#include <string.h>
#include <cuda_runtime_api.h>
#include <accfft_gpu.h>
#include <../src/operators_gpu.txx>
```

Functions

- void [accfft_grad_gpu](#) (double **A_x*, double **A_y*, double **A_z*, double **A*, accfft_plan_gpu **plan*, std::bitset< 3 > **pXYZ*, double **timer*)
- void [accfft_laplace_gpu](#) (double **LA*, double **A*, accfft_plan_gpu **plan*, double **timer*)
- void [accfft_divergence_gpu](#) (double **divA*, double **A_x*, double **A_y*, double **A_z*, accfft_plan_gpu **plan*, double **timer*)
- void [accfft_biharmonic_gpu](#) (double **BA*, double **A*, accfft_plan_gpu **plan*, double **timer*)

5.7.1 Detailed Description

CPU functions of AccFFT operators

5.7.2 Function Documentation

5.7.2.1 void [accfft_biharmonic_gpu](#) (double * *BA*, double * *A*, accfft_plan_gpu * *plan*, double * *timer*)

Computes double precision Biharmonic of its input real data *A*, and writes the output into *LA*. All the arrays must reside in the device (i.e. GPU) and must have been allocated with proper size using `cudaMalloc`.

Parameters

<i>BA</i>	ΔA
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c_gpu . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>timer</i>	See Timing AccFFT for more details.

5.7.2.2 `void accfft_divergence_gpu (double * divA, double * A_x, double * A_y, double * A_z, accfft_plan_gpu * plan, double * timer)`

Computes double precision divergence of its input vector data A_x, A_y, and A_z. The output data is written to divA. All the arrays must reside in the device (i.e. GPU) and must have been allocated with proper size using cudaMalloc.

Parameters

<i>divA</i>	$\nabla \cdot (A_x i + A_y j + A_z k)$
<i>A_x</i>	The x component of ∇A
<i>A_y</i>	The y component of ∇A
<i>A_z</i>	The z component of ∇A
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c_gpu . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>timer</i>	See Timing AccFFT for more details.

5.7.2.3 `void accfft_grad_gpu (double * A_x, double * A_y, double * A_z, double * A, accfft_plan_gpu * plan, std::bitset<3> * pXYZ, double * timer)`

Computes double precision gradient of its input real data A, and returns the x, y, and z components and writes the output into A_x, A_y, and A_z respectively. All the arrays must reside in the device (i.e. GPU) and must have been allocated with proper size using cudaMalloc.

Parameters

<i>A_x</i>	The x component of ∇A
<i>A_y</i>	The y component of ∇A
<i>A_z</i>	The z component of ∇A
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c_gpu . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>pXYZ</i>	a bit set pointer field of size 3 that determines which gradient components are needed. If XYZ={111} then all the components are computed and if XYZ={100}, then only the x component is computed. This can save the user some time, when just one or two of the gradient components are needed.
<i>timer</i>	See Timing AccFFT for more details.

5.7.2.4 `void accfft_laplace_gpu (double * LA, double * A, accfft_plan_gpu * plan, double * timer)`

Computes double precision Laplacian of its input real data A, and writes the output into LA. All the arrays must reside in the device (i.e. GPU) and must have been allocated with proper size using cudaMalloc.

Parameters

<i>LA</i>	ΔA
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c_gpu . Must be an outplace plan, otherwise the function will return without computing the gradient.

<i>timer</i>	See Timing AccFFT for more details.
--------------	---

5.8 /Users/new/accfft/src/operators_gpuf.cpp File Reference

```
#include <mpi.h>
#include <omp.h>
#include <iostream>
#include <cmath>
#include <math.h>
#include <string.h>
#include <cuda_runtime_api.h>
#include <accfft_gpuf.h>
#include <../src/operators_gpu.txx>
```

Functions

- void [accfft_grad_gpuf](#) (float *A_x, float *A_y, float *A_z, float *A, accfft_plan_gpuf *plan, std::bitset< 3 > *pXYZ, double *timer)
- void [accfft_laplace_gpuf](#) (float *LA, float *A, accfft_plan_gpuf *plan, double *timer)
- void [accfft_divergence_gpuf](#) (float *divA, float *A_x, float *A_y, float *A_z, accfft_plan_gpuf *plan, double *timer)
- void [accfft_biharmonic_gpuf](#) (float *BA, float *A, accfft_plan_gpuf *plan, double *timer)

5.8.1 Detailed Description

CPU functions of AccFFT operators

5.8.2 Function Documentation

5.8.2.1 void [accfft_biharmonic_gpuf](#) (float * *BA*, float * *A*, accfft_plan_gpuf * *plan*, double * *timer*)

Computes single precision Biharmonic of its input real data A, and writes the output into BA. All the arrays must reside in the device (i.e. GPU) and must have been allocated with proper size using cudaMalloc.

Parameters

<i>BA</i>	$\Delta^2 A$
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c_gpuf . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>timer</i>	See Timing AccFFT for more details.

5.8.2.2 void [accfft_divergence_gpuf](#) (float * *divA*, float * *A_x*, float * *A_y*, float * *A_z*, accfft_plan_gpuf * *plan*, double * *timer*)

Computes single precision divergence of its input vector data A_x, A_y, and A_z. The output data is written to divA. All the arrays must reside in the device (i.e. GPU) and must have been allocated with proper size using cudaMalloc.

Parameters

<i>divA</i>	$\nabla \cdot (A_x i + A_y j + A_z k)$
<i>A_x</i>	The x component of ∇A
<i>A_y</i>	The y component of ∇A
<i>A_z</i>	The z component of ∇A
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c_gpuf . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>timer</i>	See Timing AccFFT for more details.

5.8.2.3 `void accfft_grad_gpuf (float * A_x, float * A_y, float * A_z, float * A, accfft_plan_gpuf * plan, std::bitset< 3 > * pXYZ, double * timer)`

Computes single precision gradient of its input real data A, and returns the x, y, and z components and writes the output into A_x, A_y, and A_z respectively. All the arrays must reside in the device (i.e. GPU) and must have been allocated with proper size using cudaMalloc.

Parameters

<i>A_x</i>	The x component of ∇A
<i>A_y</i>	The y component of ∇A
<i>A_z</i>	The z component of ∇A
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c_gpuf . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>pXYZ</i>	a bit set pointer field of size 3 that determines which gradient components are needed. If XYZ={111} then all the components are computed and if XYZ={100}, then only the x component is computed. This can save the user some time, when just one or two of the gradient components are needed.
<i>timer</i>	See Timing AccFFT for more details.

5.8.2.4 `void accfft_laplace_gpuf (float * LA, float * A, accfft_plan_gpuf * plan, double * timer)`

Computes single precision Laplacian of its input real data A, and writes the output into LA. All the arrays must reside in the device (i.e. GPU) and must have been allocated with proper size using cudaMalloc.

Parameters

<i>LA</i>	ΔA
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2c_gpuf . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>timer</i>	See Timing AccFFT for more details.

5.9 /Users/new/accfft/src/operatorsf.cpp File Reference

```
#include <mpi.h>
#include <omp.h>
#include <iostream>
#include <cmath>
#include <math.h>
#include <string.h>
#include <accfftf.h>
#include <accfft_operators.h>
#include <../src/operators.txx>
#include <pnetcdf.h>
#include <cstdlib>
#include <string>
```

Functions

- void [accfft_gradf](#) (float *A_x, float *A_y, float *A_z, float *A, accfft_planf *plan, std::bitset< 3 > *pXYZ, double *timer)
- void [accfft_laplacef](#) (float *LA, float *A, accfft_planf *plan, double *timer)
- void [accfft_divergencef](#) (float *divA, float *A_x, float *A_y, float *A_z, accfft_planf *plan, double *timer)
- void [accfft_biharmonicf](#) (float *BA, float *A, accfft_planf *plan, double *timer)
- void [read_pnetcdf](#) (const std::string &filename, MPI_Offset starts[3], MPI_Offset counts[3], int gsizes[3], float *localData)
- void [write_pnetcdf](#) (const std::string &filename, MPI_Offset starts[3], MPI_Offset counts[3], int gsizes[3], float *localData)

5.9.1 Detailed Description

Single Precision CPU functions of AccFFT operators

5.9.2 Function Documentation

5.9.2.1 void accfft_biharmonicf (float * BA, float * A, accfft_planf * plan, double * timer)

Computes single precision Biharmonic of its input real data A, and writes the output into LA.

Parameters

<i>BA</i>	$\Delta^2 A$
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2cf . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>timer</i>	See Timing AccFFT for more details.

5.9.2.2 void accfft_divergencef (float * divA, float * A_x, float * A_y, float * A_z, accfft_planf * plan, double * timer)

Computes single precision divergence of its input vector data A_x, A_y, and A_z. The output data is written to divA.

Parameters

<i>divA</i>	$\nabla \cdot (A_x i + A_y j + A_z k)$
<i>A_x</i>	The x component of ∇A
<i>A_y</i>	The y component of ∇A
<i>A_z</i>	The z component of ∇A
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2cf . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>timer</i>	See Timing AccFFT for more details.

5.9.2.3 `void accfft_gradf (float * A_x, float * A_y, float * A_z, float * A, accfft_planf * plan, std::bitset<3> * pXYZ, double * timer)`

Computes single precision gradient of its input real data A, and returns the x, y, and z components and writes the output into A_x, A_y, and A_z respectively.

Parameters

<i>A_x</i>	The x component of ∇A
<i>A_y</i>	The y component of ∇A
<i>A_z</i>	The z component of ∇A
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2cf . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>pXYZ</i>	a bit set pointer field of size 3 that determines which gradient components are needed. If XYZ={111} then all the components are computed and if XYZ={100}, then only the x component is computed. This can save the user some time, when just one or two of the gradient components are needed.
<i>timer</i>	See Timing AccFFT for more details.

5.9.2.4 `void accfft_laplacef (float * LA, float * A, accfft_planf * plan, double * timer)`

Computes single precision Laplacian of its input real data A, and writes the output into LA.

Parameters

<i>LA</i>	ΔA
<i>plan</i>	FFT plan created by accfft_plan_dft_3d_r2cf . Must be an outplace plan, otherwise the function will return without computing the gradient.
<i>timer</i>	See Timing AccFFT for more details.

5.9.2.5 `void read_pnetcdf (const std::string & filename, MPI_Offset starts[3], MPI_Offset counts[3], int gsizes[3], float * localData)`

Read a single-precision parallel-netcdf file.

We assume here that localData is a scalar.

Pnetcdf uses row-major format (same as FFTW).

Parameters

<i>in</i>	<i>filename</i>	: PnetCDF filename
<i>in</i>	<i>starts</i>	: offset to where to start reading data
<i>in</i>	<i>counts</i>	: number of elements read (3D sub-domain inside global)

in	<i>gsizes</i>	: global sizes
out	<i>localData</i>	: actual data buffer (size : nx*ny*nz*sizeof(float))

localData must have been allocated prior to calling this routine.

5.9.2.6 void write_pnetcdf (const std::string & filename, MPI_Offset starts[3], MPI_Offset counts[3], int gsizes[3], float * localData)

Write a parallel-nedcdf file.

We assume here that localData is a scalar.

Pnetcdf uses row-major format (same as FFTW).

Parameters

in	<i>filename</i>	: PnetCDF filename
in	<i>starts</i>	: offset to where to start reading data
in	<i>counts</i>	: number of elements read (3D sub-domain inside global)
in	<i>gsizes</i>	: global sizes
in	<i>localData</i>	: actual data buffer (size : nx*ny*nz*sizeof(float))

Index

/Users/new/accfft/src/accfft.cpp, 9
/Users/new/accfft/src/accfft_common.cpp, 12
/Users/new/accfft/src/accfft_gpu.cpp, 14
/Users/new/accfft/src/accfft_gpuf.cpp, 18
/Users/new/accfft/src/accfftf.cpp, 22
/Users/new/accfft/src/operators.cpp, 25
/Users/new/accfft/src/operators_gpu.cpp, 28
/Users/new/accfft/src/operators_gpuf.cpp, 30
/Users/new/accfft/src/operatorsf.cpp, 32

accfft.cpp

accfft_destroy_plan, 9
accfft_execute_c2c, 10
accfft_execute_c2r, 10
accfft_execute_r2c, 10
accfft_local_size_dft_c2c, 11
accfft_local_size_dft_r2c, 11
accfft_plan_dft_3d_c2c, 11
accfft_plan_dft_3d_r2c, 12

accfft_alloc

accfft_common.cpp, 13

accfft_biharmonic

operators.cpp, 26

accfft_biharmonic_gpu

operators_gpu.cpp, 28

accfft_biharmonic_gpuf

operators_gpuf.cpp, 30

accfft_biharmonicf

operatorsf.cpp, 32

accfft_cleanup

accfft_common.cpp, 14

accfft_cleanup_gpu

accfft_gpu.cpp, 15

accfft_cleanup_gpuf

accfft_gpuf.cpp, 19

accfft_common.cpp

accfft_alloc, 13

accfft_cleanup, 14

accfft_create_comm, 14

accfft_free, 14

accfft_init, 14

accfft_create_comm

accfft_common.cpp, 14

accfft_destroy_plan

accfft.cpp, 9

accfft_gpu.cpp, 15

accfft_gpuf.cpp, 19

accfftf.cpp, 22

accfft_destroy_plan_gpu

accfft_gpu.cpp, 15

accfft_gpuf.cpp, 19

accfft_divergence

operators.cpp, 26

accfft_divergence_gpu

operators_gpu.cpp, 29

accfft_divergence_gpuf

operators_gpuf.cpp, 30

accfft_divergencef

operatorsf.cpp, 32

accfft_execute_c2c

accfft.cpp, 10

accfft_execute_c2c_gpu

accfft_gpu.cpp, 16

accfft_execute_c2c_gpuf

accfft_gpuf.cpp, 19

accfft_execute_c2cf

accfftf.cpp, 23

accfft_execute_c2r

accfft.cpp, 10

accfft_execute_c2r_gpu

accfft_gpu.cpp, 16

accfft_execute_c2r_gpuf

accfft_gpuf.cpp, 20

accfft_execute_c2rf

accfftf.cpp, 23

accfft_execute_r2c

accfft.cpp, 10

accfft_execute_r2c_gpu

accfft_gpu.cpp, 16

accfft_execute_r2c_gpuf

accfft_gpuf.cpp, 20

accfft_execute_r2cf

accfftf.cpp, 23

accfft_free

accfft_common.cpp, 14

accfft_gpu.cpp

accfft_cleanup_gpu, 15

accfft_destroy_plan, 15

accfft_destroy_plan_gpu, 15

accfft_execute_c2c_gpu, 16

accfft_execute_c2r_gpu, 16

accfft_execute_r2c_gpu, 16

accfft_local_size_dft_c2c_gpu, 17

accfft_local_size_dft_r2c_gpu, 17

accfft_plan_dft_3d_c2c_gpu, 17

accfft_plan_dft_3d_r2c_gpu, 18

accfft_gpuf.cpp

accfft_cleanup_gpuf, 19

accfft_destroy_plan, 19

- accfft_destroy_plan_gpu, 19
- accfft_execute_c2c_gpuf, 19
- accfft_execute_c2r_gpuf, 20
- accfft_execute_r2c_gpuf, 20
- accfft_local_size_dft_c2c_gpuf, 20
- accfft_local_size_dft_r2c_gpuf, 21
- accfft_plan_dft_3d_c2c_gpuf, 21
- accfft_plan_dft_3d_r2c_gpuf, 21
- accfft_grad
 - operators.cpp, 26
- accfft_grad_gpu
 - operators_gpu.cpp, 29
- accfft_grad_gpuf
 - operators_gpuf.cpp, 31
- accfft_gradf
 - operatorsf.cpp, 33
- accfft_init
 - accfft_common.cpp, 14
- accfft_inv_biharmonic
 - operators.cpp, 27
- accfft_inv_laplace
 - operators.cpp, 27
- accfft_laplace
 - operators.cpp, 27
- accfft_laplace_gpu
 - operators_gpu.cpp, 29
- accfft_laplace_gpuf
 - operators_gpuf.cpp, 31
- accfft_laplacef
 - operatorsf.cpp, 33
- accfft_local_size_dft_c2c
 - accfft.cpp, 11
- accfft_local_size_dft_c2c_gpu
 - accfft_gpu.cpp, 17
- accfft_local_size_dft_c2c_gpuf
 - accfft_gpuf.cpp, 20
- accfft_local_size_dft_c2cf
 - accfftf.cpp, 24
- accfft_local_size_dft_r2c
 - accfft.cpp, 11
- accfft_local_size_dft_r2c_gpu
 - accfft_gpu.cpp, 17
- accfft_local_size_dft_r2c_gpuf
 - accfft_gpuf.cpp, 21
- accfft_local_size_dft_r2cf
 - accfftf.cpp, 24
- accfft_plan_dft_3d_c2c
 - accfft.cpp, 11
- accfft_plan_dft_3d_c2c_gpu
 - accfft_gpu.cpp, 17
- accfft_plan_dft_3d_c2c_gpuf
 - accfft_gpuf.cpp, 21
- accfft_plan_dft_3d_c2cf
 - accfftf.cpp, 24
- accfft_plan_dft_3d_r2c
 - accfft.cpp, 12
- accfft_plan_dft_3d_r2c_gpu
 - accfft_gpu.cpp, 18
- accfft_plan_dft_3d_r2c_gpuf
 - accfft_gpuf.cpp, 21
- accfft_plan_dft_3d_r2cf
 - accfftf.cpp, 25
- accfftf.cpp
 - accfft_destroy_plan, 22
 - accfft_execute_c2cf, 23
 - accfft_execute_c2rf, 23
 - accfft_execute_r2cf, 23
 - accfft_local_size_dft_c2cf, 24
 - accfft_local_size_dft_r2cf, 24
 - accfft_plan_dft_3d_c2cf, 24
 - accfft_plan_dft_3d_r2cf, 25
- operators.cpp
 - accfft_biharmonic, 26
 - accfft_divergence, 26
 - accfft_grad, 26
 - accfft_inv_biharmonic, 27
 - accfft_inv_laplace, 27
 - accfft_laplace, 27
 - read_pnetcdf, 27
 - write_pnetcdf, 27
- operators_gpu.cpp
 - accfft_biharmonic_gpu, 28
 - accfft_divergence_gpu, 29
 - accfft_grad_gpu, 29
 - accfft_laplace_gpu, 29
- operators_gpuf.cpp
 - accfft_biharmonic_gpuf, 30
 - accfft_divergence_gpuf, 30
 - accfft_grad_gpuf, 31
 - accfft_laplace_gpuf, 31
- operatorsf.cpp
 - accfft_biharmonicf, 32
 - accfft_divergencef, 32
 - accfft_gradf, 33
 - accfft_laplacef, 33
 - read_pnetcdf, 33
 - write_pnetcdf, 34
- read_pnetcdf
 - operators.cpp, 27
 - operatorsf.cpp, 33
- write_pnetcdf
 - operators.cpp, 27
 - operatorsf.cpp, 34