LibPFASST x SWEET

Valentina Schüller

January 18, 2022

Abstract

This text outlines how to use the SWEET programs making use of LibPFASST. I describe central development decisions as well as some peculiarities of how these programs behave. See the SWEET tutorials for specific help with running the programs.

Contents

1 Using LibPFASST programs in SWEET

Currently supported programs:

- explicit SDC for the shallow water equations on the sphere (`libpfasst_swe_sphere_expl_sdc`)
- IMEX SDC for the shallow water equations on the sphere (`libpfasst_swe_sphere_imex_sdc`)
- MLSDC for the shallow water equations on the sphere (`libpfasst_swe_sphere_mlsdc`)

1.1 Compilation

This works:

```
scons --program=PROGRAM_STRING --quadmath=disable --libpfasst=enable
--sweet-mpi=enable --libsph=enable --plane-spectral-space=disable
--sphere-spectral-space=enable --threading=off --libfft=enable
--sphere-spectral-dealiasing=enable
```

1.2 Runtime

- so far, the programs have only been tested with the Galewsky benchmark!
- in addition to the SWEET runtime options, the LibPFASST programs have flags for customizing the PFASST algorithm.
  - Flags for all programs
    - `libpfasst-nodes-type`: type of nodes used by LibPFASST (Gauss-Lobatto or Gauss-Legendre)
    - `libpfasst-nnodes`: number of nodes used by PFASST algorithm
    - `libpfasst-niters`: number of iterations per time step used by PFASST algorithm
  - Additional flags for `libpfasst_swe_sphere_mlsdc`
    - `libpfasst-nlevels`: only 1 and 2 levels are supported so far.
    - `libpfasst-u2/4/6/8`: hyperviscosity values of order 2, 4, 6, 8. See [2, p. 500] for an explanation of hyperviscosity.
    - `libpfasst-u-fields`: which fields to apply hyperviscosity to. Can be any comma-separated combination of *phi_pert, div, vrt*. Additional options are *all* and *none*.
- **note:** the flags `timestepping-method` and `timestepping-order` are **not** used by the LibPFASST programs! To compute your achievable order, see section 2
- **note:** the flags `-u` and `-U` are **not** used by the LibPFASST programs.

1.3 Testing

Convergence tests are contained in `tests/70_program_libpfasst_*`, compilation tests are in `tests/10_compile/10_compile_libpfasst_*`.

2 Special hints/peculiarities

2.1 ... when using the programs

- `--output-file-mode=csv` is the default for the other SWEET programs but **not supported** in the LibPFASST programs. Use `--output-file-mode=bin`
- the LibPFASST programs do not support GUI output
- in general: achievable convergence order of SDC: $\mathcal{O}\left(\Delta t^{\min(n_{\text{iters}},m)}\right)$
  - m: order of quadrature nodes – not the integration but the *interpolation* order! (1 higher than for integration)
- achievable convergence order with Gauss-Lobatto nodes

$$\mathcal{O}\left(\Delta t^{\min(n_{\text{iters}},2n_{\text{nodes}}-2)}\right)$$

- achievable convergence order with Gauss-Legendre nodes (for Gauss-Legendre nodes, the interval boundaries are part of the node count)

$$\mathcal{O}\left(\Delta t^{\min(n_{\text{iters}},2(n_{\text{nodes}}-2))}\right)$$

2.2 ... when looking at the code

- in the code, `dt` is the time step size Δt provided at runtime with the `--dt` flag. `dtq` is $\Delta t \cdot \omega$, where ω is the current quadrature weight. **Note:** for Gauss-Legendre nodes, the interval boundaries are part of the node count but their weight is of course 0, thus `dtq` can be 0!
- `eval()` functions refer to an explicit evaluation of (parts of) the right hand side at a particular point in time
- `comp()` functions refer to an implicit solve of (parts of) the right hand side at a particular point in time

Explicit SDC

The explicit SDC program `libpfasst_swe_sphere_expl_sdc` uses LibPFASST's IMEX sweeper but does not use its IMEX functionality. Instead, we integrate the whole right hand side explicitly by providing one routine `ceval()` which evaluates the whole right-hand side $\frac{\partial \boldsymbol{U}}{\partial t}$ at once.

IMEX-SDC

The IMEX SDC program `libpfasst_swe_sphere_imex_sdc` uses LibPFASST's IMEX sweeper and the following splitting (cf. [1]):

$$\frac{\partial \boldsymbol{U}}{\partial t} = \mathcal{L}_G(\boldsymbol{U}) + \mathcal{L}_F(\boldsymbol{U}) + \mathcal{N}(\boldsymbol{U})$$

Table 1: Right-hand side components of the MLSDC sweeper.

function	related physics	meaning	function calls / sweep
`ceval_f1()`	$\mathcal{L}_F(\boldsymbol{U}) + \mathcal{N}(\boldsymbol{U})$	explicit evaluation of non-stiff terms (the EX in IMEX)	in each correction
`ceval_f2()`	$\mathcal{L}_G(\boldsymbol{U})$	explicit evaluation of stiff terms	once, before the corrections
`ccomp_f2()`	$\mathcal{L}_G(\boldsymbol{U})$	implicit computation of stiff terms	in each correction
`ceval_f3()`	(hyper-)viscosity	explicit evaluation of viscosity	once, before the corrections
`ccomp_f3()`	(hyper-)viscosity	implicit computation of viscosity	in each correction

The IMEX sweeper uses a splitting of the right-hand side

$$f = f_1 + f_2$$

where f_1 is solved explicitly and f_2 is solved implicitly. In our implementation, $f_1 = \mathcal{L}_F(\boldsymbol{U}) + \mathcal{N}(\boldsymbol{U})$ is solved explicitly and the stiff linear gravity modes $f_2 = \mathcal{L}_G(\boldsymbol{U})$ are integrated implicitly, as described in [1].

It might seem surprising that there exists an evaluation function for the implicit piece in the IMEX sweeper. This is intended by the IMEX-SDC algorithm in [3, Algorithm 1].

MLSDC

The MLSDC implementation `libpfasst_swe_sphere_imex_sdc` uses LibPFASST's MISDC sweeper and splits the right-hand side into three pieces

$$f = f_1 + f_2 + f_3$$

where f_1 is the explicit piece and both f_2 and f_3 are solved implicitly. f_1 and f_2 are both implemented as given above, $f_1 = \mathcal{L}_F(\boldsymbol{U}) + \mathcal{N}(\boldsymbol{U})$ and $f_2 = \mathcal{L}_G(\boldsymbol{U})$. f_3 is used to apply artificial viscosity to the system.

It might seem surprising that there exists an evaluation function for the implicit pieces in the MISDC sweeper. This is intended by the IMEX-SDC algorithm in [3, Algorithm 1]. These functions are called once per sweep, before the corrections.

For an overview of these functions, see Table 1.

3 Development decisions

- whenever a new version of the sweeper is used, a new `libpfasst_swe_sphere_*` program should be created alongside a new compilation and convergence test

- headers shared by all LibPFASST programs are to be stored in `src/programs/libpfasst_interface`.
- the LibPFASST programs should behave as much as the `swe_sphere` program as possible, from a user's perspective. Particularly, this affects:
  - variable naming
  - output behavior
  - output file name and structure

4 More information

- LibPFASST repository and documentation
- The IMEX-SDC algorithm is described in [3, Algorithm 1]
- original SWEET x LibPFASST paper: [1]

References

[1] Francois Hamon, Martin Schreiber, and Michael Minion. Multi-Level Spectral Deferred Corrections Scheme for the Shallow Water Equations on the Rotating Sphere. *Journal of Computational Physics*, 376:435–454, January 2019. arXiv: 1805.07923.

[2] Peter Lauritzen, Christiane Jablonowski, Mark Taylor, and Ramachandran Nair, editors. *Numerical Techniques for Global Atmospheric Models*, volume 80 of *Lecture Notes in Computational Science and Engineering*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[3] Robert Speck, Daniel Ruprecht, Matthew Emmett, Michael Minion, Matthias Bolten, and Rolf Krause. A multi-level spectral deferred correction method. *BIT Numerical Mathematics*, 55(3):843–867, September 2015. arXiv: 1307.1312.