Implicit time stepping schemes in PyClaw

[HPC]^3 Workshop 2012

1 Target

During the workshop HPC^3, the implicit group has focused on the definition and implementation of the main building blocks for the construction of implicit time marching schemes in PyClaw. Only method of lines solver has been considered, i.e. the sharpclaw solver.

The main objectives of the group have been:

- discuss and develop a preconditioning matrix based on the wave decomposition computed by the Riemann solver to accelerate the convergence of the algorithm choosed for the solution of the nonlinear system.
- complete the PETSc/SNES interface in PyClaw for the implicit sharpclaw solver
- implement and test the implicit downwind-biased WENO approach with explicit and implicit SSPRK methods

All the objectives have been achieved, though the final implementation of all features still requires some changes and refactoring of the code. In the next sections each achievements will be briefly described.

2 Solution of the nonlinear system of equations

Consider the semi-discretization of a system of hyperbolic PDEs:

$$\frac{dq}{dt} = R(q), \tag{1}$$

where q and $R(q)$ are the vector of the unknowns (in many cases it is called vector of the conserved variables if the system is written in conservation form) and the residual which is the result of the discretization of the spatial derivatives. Equation (1) is a system of ordinary differential in time that can be integrated in principle with any methods for such a system.

An implicit time discretization for (1) appears as

$$D_t Q^{n+1} + R(Q^{n+1}) = F(Q^{n+1}) \tag{2}$$

where D_t is the implicit time stepping operator. Equation (2) represent a nonlinear problem which must be solved at every time step (or stage). In order to achieve the temporal accuracy of the temporal operator, the nonlinear equation must be solved to an appropriately small error. The solution at each time step can be found using a dual time stepping technique or solving directly the system (2) with for instance the Newton-Krylov algorithm.

2.1 Newton-Krylov method

Expanding Equation (2) in a Taylor series gives

$$F(Q^{k+1}) = F(Q^k) + \frac{\partial F(Q^k)}{\partial Q^k}\Delta Q^k + \ldots = 0 \tag{3}$$

Ignoring the higher order terms gives the following linear system, which is solved at each iteration:

$$A^k \Delta Q^k = -F(Q^k), \tag{4}$$

where A^k is the Jacobian of $F(Q^k)$ given by

$$A^k = \frac{\partial F(Q^k)}{\partial Q^k} = \frac{1}{\alpha \Delta t} I + \frac{\partial R(Q^k)}{\partial Q^k}, \tag{5}$$

where I is the unit matrix and α is the coefficient of the time stepping scheme. If this linear system is solved iteratively to some finite tolerance, the resulting method is called inexact Newton method.

It is seen that, at each Newton iteration a linear system of equations have to be solved. We plan to use the Krylov-based iterative method for solving such a system. In order to be effective, the Krylov-based iterative solver requires the use of preconditioning. Preconditioning can be used to transform the coefficient matrix of the system which arises at each time step so that it is nearly positive definite. We have planned to use right preconditioning:

$$AP^{-1}Px = b, \tag{6}$$

where $P \approx A$. To form the preconditioner in PyClaw we are planning to use the wave decomposition already available from the Riemann solver.

Note that for the WENO5 discretization three terms are computed by the Riemann solver call: $A^-\Delta Q$, $A^+\Delta Q$ and $A\Delta Q$. For smooth solution the last term, also called total fluctuation, is of the order $O(\Delta x^5)$ and thus it is negligible. In this case, the preconditioning matrix can be constructed just using the classical fluctuation terms.

2.2 Dual Time Step

The problem can also be recast as a modified steady-state calculation pseudotime,

$$D_\tau Q + R^*(Q) = 0, \tag{7}$$

where the modified residual $R^*(Q)$ represent the left-hand side of Equation (2) and D_τ is a pseudotime smoother operator. Updating the solution from time level n to $n+1$ can be done by computing a steady-state solution to Equation (7) using multigrid (already available in PETSc).

One way to accelerate the convergence at each pseudotime step is the use of a preconditioner that is intended to cluster residual eigenvalues away from the origin into a region of the complex plane for which the smoother can provide a rapid damping. This preconditioning matrix will still constructed using the wave decomposition available from the Riemann solver.

3 PETSc interface to SNES and setup of a new class of solver

The implementation of the interface to the nonlinear algebraic solver library PETSc/SNES has been implemented at `https://github.com/mparsani/pyclaw/commits/implicit-hpc3` in a new module called *implicitsharpclaw.py* which is stored in `src/petclaw/sharpclaw`. This module contains a new class of solver for the solution of 1D system of PDEs using WENO spatial discretizations and backward Euler time integration. The implicit class solver inherits from the *Solver* class.

The current implementation does not provide any preconditioning matrix. However, it can easily use geometric multigrid and polynomial smoothers (on each MG level) available in PETSc to accelerate the convergence of the nonlinear solve at each time step. The development of a good preconditioning for the shallow water equations and stiff gravity waves is the goal of an ongoing project (see also previous objective). The SNES coloring option has been tested but due to scaling problems have not been successful, for a CFL number larger than one.

The extension of the implicit solver to 2D systems of PDEs should be straightforward.

4 Downwind WENO spatial discretization

The downwind-biased WENO discretization has been implemented at `https://github.com/LuluLiu/pyclaw/commits/downwind-biased`. To avoid the duplication of Fortran functions, the interface of the *flux1.f90* function has been modified to get an additional input parameter, i.e. *upwind*. The value of the parameter is one for an upwind WENO reconstruction and zero for a downwind-biased spatial discretization.

The new solver has been tested with several explicit strong stability preserving Runge-Kutta (SSPRK) schemes (SSP22,SSP43 and SSP105) and one implicit SSPRK scheme. Convergence studies have verified the order of accuracy of the explicit solvers. In an ongoing work we are testing the implicit SSPRK.

The implementation has been done in sharpclaw.py and the solution of the nonlinear system arising from the implicit time discretization has been solved with the scipy.optimize.fsolve Python function.

5 Future work

In the future we plan to complete the work started during the workshop with:

- implement and test the right preconditioning matrix defined in 2.
- extend the implicit sharpclaw solver for the solution of 2D system of PDEs
- design preconditioning matrix for the shallow water equations with stiff gravity waves
- use PETSc/SNES nonlinear solver to solve the nonlinear system arising from the implicit time discretization with the downwind-biased WENO discretization