HPC^3 2012 WENO group report

Matthew Emmett

February 28, 2012

1 Introduction

The focus of this group was to get WENO reconstructions working on 1d mapped grids. Briefly, this was accomplished by:

1. Generating generic WENO routines (of various orders) that require smoothness indicator coefficients, optimal weights, and reconstruction coefficients to be supplied at run time and including them in SharpClaw.
2. Modifying PyClaw to detect when a nonuniform grid is being used, and subsequently calling PyWENO to compute the above coefficients and weights. These are attached to the PyClaw "state" object and passed down to the SharpClaw reconstructor if necessary.

The 1D homogeneous acoustics[1] example was used as a simple test case. The example runs to completion and the solution appears to be correct when a nonuniform grid is used. Further testing should be done.

2 1D homogeneous acoustics example

The 1D homogeneous acoustics example was used to do preliminary testing. The final solution should be the same as the initial condition. The error between these states is printed as the last line of output when running the example.

To run the example on a mapped grid run:

```
$ python acoustics.py solver_type='sharpclaw' mapped=True iplot=True
...
0.026433527054
```

To run on a mapped grid using the uniform WENO routines:

```
$ python acoustics.py solver_type='sharpclaw' mapped=True unifcoeffs=True iplot=True
...
0.0264796563622
```

To run on a mapped grid using the classic solver:

[1] https://github.com/memmett/pyclaw/tree/hpc3-weno/apps/acoustics_1d_homogeneous

```
$ python acoustics.py mapped=True unifcoeffs=True iplot=True
...
0.141743703601
```

Note that the WENO runs are more accurate than the classic run. Also note that the WENO run using uniform coefficients is only slightly less accurate than the WENO run using proper nonuniform coefficients.

The grid used in this example is computed according to

```
def mapc2p(patch, x):
    m = x

    idx = (x >= 0.2)*(x <= 0.5)
    m[idx] = x[idx] + 0.008*np.sin(2*np.pi*(x[idx]-0.2)/0.3)

    return m
```

The plotting routines were not updated to handle mapped grids (this was beyond the scope of this working group). The existing plotting routines can be used regardless, but one needs to be careful when interpreting the output.

3 Future work

More thorough testing (ie, a proper convergence analysis) should be done, and extensions to higher dimensions need to be carefully thought through.

The items enumerated below (see Known issues) should also be addressed.

4 Git repository and notable changes

The `hpc3-weno` branch (which is the default) of `http://github.com/memmett/pyclaw` contains PyClaw related code. Fresh versions of PyWENO and SymPy are also necessary.

Integrating the code developed by this group during the workshop should be as easy as merging the `hpc3-weno` branch of the above Git repository. Notable changes include:

1. `src/pyclaw/state.py`: A new method called `precompute_mapped_weno` was added. This is called directly by the user script if mapped grids are used (similar in spirited to the requirement that `compute_p_centers` and `compute_p_edges` are called by the user (AFAIK)).
2. `src/pyclaw/sharpclaw/sharpclaw.py`: If the state's `weno_mapped` attribute is `True`, the WENO coeffs are passed down to `flux1`.
3. `src/pyclaw/sharpclaw/flux1.f90`: The calling convention was changed to accept WENO coefficients, and call `weno_comp_mapped` instead of `weno_comp` during the reconstruction phase where appropriate.
4. `src/pyclaw/sharpclaw/reconstruct.f90`: A new `weno_comp_mapped` routine was added.
5. `src/pyclaw/sharpclaw/weno.py`: This now generates generic WENO routines for mapped grids alongside the existing WENO routines for uniform grids.
6. `src/pyclaw/sharpclaw/weno.f90`: Re-generated.

5 Known issues

1. Unfortunately, computing the smoothness indicators is quite time consuming (I am going to include a cached set of smoothness polynomials in PyWENO to speed this up).
2. Ideally users would compute WENO coefficients once when they settle upon a grid and load them from a cache for subsequent runs. Perhaps this could be done automatically by PyClaw.
3. To save time during the conference, only a few orders of WENO routines were generated by `src/pyclaw/sharpclaw/wenc` This should be increased back to 17th order and re-run.
4. Non-uniformities near domain boundaries are probably not handled correctly. Either PyClaw or PyWENO should be tweaked to handle this properly.