CGEM-SCHISM Status Report

Lisa L. Lowe (lllowe@ncsu.edu) - 11/11/2023

Summary

We have completed a working draft of CGEM-SCHISM. Box model testing seems to confirm expected behavior; here is a link to a pdf from the test case. Videos of the final testing run with an LSC2 hybrid grid are available online for inspection. This document describes any modifications to the SCHISM source code, a description of the CGEM subroutines, notes about the implementation, and recommendations for next steps.

There is also a description of changes to stoichiometry - these are CGEM algorithm related changes rather than SCHISM related changes, although the motivation driving the changes was that the current implementation of stoichiometry would make incorporation into SCHISM very difficult, and would have made a leap to something like Framework for Aquatic Biogeochemical Models(FABM) impossible. Efforts toward leveraging existing frameworks, like FABM, will benefit the community.

When significant changes are made to a couple thousand lines of code, bugs are to be expected; the code should be thoroughly tested by numerical experiments before jumping into comparing model results with measured data. 'Working draft' means the code runs without crashing, appears to be giving reasonable values for the state variables, and state variables are behaving as expected relative to each other, all according to the current input file and the limited number of tests completed so far.

Code development

Starting from the code base currently in EPA repository: https://github.com/USEPA/CGEM

- Rewrote biogeochem variables from `ff(i,j,k,nf)` grid to a cell `ff(nf)`
- Changes include array allocation, namelist inputs, change to `real(rkind)`, etc.
- Added hooks to schism: `S, T, Wind, Rad, dz, lat, lon`, etc.
- Rewrote stoichiometry (CGEM related, not because of SCHISM): details below.

I attempted to keep CGEM and SCHISM completely separate, so that the rewritten CGEM subroutines could more easily be incorporated into different models. The only thing I did not separate was the last minute change to `schism_glbl rkind`. (For most subroutines, I did a simple 'query replace' for `real` to `real(rkind)`.) This should not be a problem when linking the code to other models; create a dummy 'schism_glbl.F90' that defines `rkind`, or add `rkind` to the cgem module.

Some features have not been implemented due to time constraints. To incrementally add back the omitted features, get the original code from the EPA repository, and modify variable and array declarations to match with the new code.

Subroutines

Modification to SCHISM code

I am using the "GEN" option, which avoids the need to modify additional SCHISM code. No additional 'tracers model' has been added; do not offer to push this version to the official SCHISM repo as an additional tracer model. To simplify the search for changes, I usually add `!L3` to lines before and after any modifications to original code.

I forked the original SCHISM repo to keep track of the changes.
https://github.com/oybcst/schism
The code used in the final testing is tagged v0.1 and can be downloaded here
https://github.com/oybcst/schism/releases/tag/v0.1

Core/schism_glbl.F90

- CGEM has 30 variables. SCHISM allocates a maximum of 30 tracers, 2 of which are reserved for S and T. I increased `mntracers` to 32. This number will need to be increased if increasing the number of phytoplankton and zooplankton groups.

Hydro/schism_init.F90

- Use `grid` module
- Define number of tracers
- Define CGEM time variables according to SCHISM internal time variables
- Call `grid_setup` and `cgem_setup`, to define variables, read namelists, and allocate arrays

Hydro/schism_step.F90

- Calls `cgem_run`
- I attempted to change the `writeout_nc` call from nodes to elements, but the metadata still says nodes. I didn't have time to investigate further. The annoying issue I was trying to avoid is that when you are trying to interpret timeseries data, it looks like there is an extra layer, and the extra layer looks wrong.

CGEM code

grid.F90

- Defines number of layers and time variables grid_setup.F90

- Calculates `START_SECONDS`, used in calculating the julian day for solar radiation, and for the solar zenith angle, which is used in the light model regardless of whether Rad is calculated or read in from fluxes.

cgem_setup.F90

- Sets `nf`, number of state variables, and does a check that the number of tracers set in param.nml matches the current number of state variables.
- Calls subroutines to `cgem_allocate`, `cgem_read`, and `cgem_initialize.`

cgem.F90

- `cgem_allocate`: allocates arrays from `nf`
- `cgem_read`: reads namelists
- `cgem_init`: initializations based on reading namelists
- Variable declarations, enumeration definitions, and allocation of arrays. `nf` is the number of state variables.
- The array `f_min(nf)` has been added for the purpose of resetting values when values are negative, in lieu of zeros, where `f_min(A,Z)=1`, `f_min(Q)=Qmin`, and all others are `f_min=tiny(x).`
- Maximum number of Zooplankton is 2, because of the way it reads in namelists. To fix this, either check and document how to properly read in an array with dimensions (nospA, nospZ), or modify the initialization of those arrays. Original CGEM reads nospZ number of lines from the GEM_InputFile.
- Converts input sinking values from m/d to m/s. SCHISM uses 'positive' for sinking rates.

cgem_run.F90

- This is the interface between SCHISM and CGEM.
- The main difference is array indexing. I implemented it in such a way that it was less confusing for me, and less disruptive to the existing CGEM code. There are several things that could be modified for efficiency.
    - In a loop over columns, `k=1,km`, I would use 1 as the surface and SCHISM would use km as the surface.
    - In a loop over arrays, I use `ff` from `1:nf`, but the first two tracers in the SCHISM `tr_el` are temperature and salinity, so `tr_el` is from `1:nf+2`.
    - The light model needs all the state variables over a column, `ff(km,nf).`
    - `cgem_step` needs 1 cell of the state vectors, `ff(nf).`
    - `cgem_sink` needs one state variable over a column, `ff(km).`
- I only implemented a sinking routine because I was getting errors related to the sinking rate and wasn't quite sure what SCHISM was doing. The benefit of using `cgemsink` is that it is easier to add write statements and find zeros. Ultimately, I think it would be much better to use SCHISM's native sinking.
- I copied the code for solar and wind directly from the SCHISM COSINE bio module, and used the same pattern of indexing to define `lat/lon`. It seems to work. You can check

all three by setting `checkwindrad=1`. If doing this check, run only for a *very* short time, else the resulting output file will be huge - possibly so huge you can't open it anyway - and try to start the simulation during the day.

- Solar flux input is W/m2 and is converted to photons/cm2/s. getSolar is in photons/cm2/s.

Rad…please check and decide whether Rad needs to be multiplied by some factor when reading in your solar fluxes. Also decide whether you want to add the option `PARfac`. Here are comments from the original code:

```
! Rad is just above sea surface.
! Rad was short wave generated by NRL, multiplied by SWtoPAR: ratio
of PAR to
! shortwave radiation (hardcoded 4/30/14 to 0.43).
! Hardcoded to 0.47 on 2/11/16, Re: Tsubo and Walker, 2005
! PARfac is a multiplication factor for testing
Rad = (0.47*Rad) * PARfac
```

Latitude and longitude are used in `getSolar`, `calculate_solar_zenith`, and mocsy's `vars` in the following format:

- Latitude (deg); lat > 0 means North of the Equator
- Longitude (deg E, 0 <= lon < 360);

If not using a SCHISM grid with proper lat/lon coordinates defined in SCHISM's `xlon`, `xlat`, choose `cgemcoords=.TRUE.`and pick a single lat/lon for the simulation (`cgemlat, cgemlon`).

calcAgrow.F90

- Input and outputs are cells instead of columns
- `calc_Agrow`: phytoplankton growth and respiration
- `func_E`: Factor for light dependent growth rate
- `func_S`: Nutrient(S) dependent growth functions
- `func_T`: Temperature adjustment

call_iop_par.F90

- Takes array of all state variables in a column, `ff(km,nf)`
- Does formatting and calculations required to call Brad Penta's light model

IOP_PARattenuation.F90

- Brad Penta's light model

surface_flux.F90

- Calculates O2 and DIC flux

moc_src:

- We did not write these, they are a library, and I forget where they came from.
- Mocsy is problematic because of hard coded mixed precision: in declarations *and* use of `REAL()` or `DBLE()`. Also, moscy uses 'assumed array shapes', meaning you cannot pass a single number, you need to create an array with one element. I attempted to modify each of moscy's 16 subroutines from mixed precision and assumed arrays to SCHISM's `rkind` and for a cell-not-column with quick vi query-replace, but briefly: it didn't work out. You will see in `cgem_step` that mocsy variables are handled differently. When you pass a value as one type of precision to a function expecting another, if you are lucky you will get an error, but more likely you will get hard-to-find bugs in the form of zeros or undefined variables.
- `cgem_step` calls mocsy to calculate pH, which is used in the DIC surface flux.
- Mocsy calculates pCO2, but we had been using pCO2 as an input parameter, rather than using the mocsy calculated one. I left that alone, pending further testing of mocsy.

cgem_utils.F90:
- Contains functions to calculate Schmidt number, water density, etc.

calc_solar_zenith.F90
- Calculates solar zenith angle needed for Brad's light routine

get_solar.F90
- Calculates solar radiation when no fluxes are available.
- To use, set `Which_rad = 0`

Phyto and Zooplankton Groups

Due to time constraints, the code has only been tested for one phytoplankton and two zooplankton groups. The only anticipated problem in the CGEM code is reading in zooplankton-related arrays from the namelist (see above). To run with different group numbers, the param.nml file would need to be modified, to redefine `iof_gen(nf)`. The `iof_gen(nf)` starts with `ff(iA)` because that is how the arrays are defined in the CGEM code. Scripts for initial conditions and extracting timeseries read the cgem.nml file which includes the number of groups and should work as-is, but the VisIt scripts use a dictionary `cgem_vars=["name"= "GEN_1"]` and would need some modifications.

Changes for Stoichiometry

In CGEM:
- Stoichiometry is the C:N:P ratios for the organic matter, normalized so P=1. It is fixed for OM in rivers and BC.
- The stoichiometry varies based on N&P content of detritus, which is due to nutrient quotas for Phytoplankton (Qn, Qp).

- The stoichiometry is used in the reaction subroutine, for remineralization.

CGEM equations are
` ff_new = ff_old + ff(y_old)*dt`
which I'll rewrite as
 all the new = **all** the old + **bits** from new calculations that depend on all the old

The CGEM code
- Calculates the amount of C, N, and P in the **bits** (dead stuff, sloppy feeding, and whatnot) generated during that timestep. Thus, stoichiometry calculated with those values would be the stoichiometry for the *new* bits of OM.
- The `reaction` subroutine takes as arguments OM1, OM2, and stoichiometry, and returns the rates.

The original-original CGEM, meaning my copy of CGEM that was hardcoded to run in the Gulf with NCOM data, was using OM(all the old) and stoich(just the new bits). It did not store any information about previous stoichiometry. But OM should be remineralized based on stoichiometry of *all* OM, not just stoichiometry of the new bits. In the CGEM code in the EPA repository, we calculated and tracked stoichiometry(all the old). The 'stoichiometry' was saved in 3D arrays, which was unwieldy, but it worked and mass balanced.

When I was rewriting CGEM for SCHISM, ‘unwieldy’ became 'very problematic' for SCHISM infrastructure and memory allocation, and it occured to me that the previous improvement was not *exactly* correct either, because OM is moving around, and the stored stoichiometry was for the OM that 'used to be in that cell'.

In the first pass of CGEM-SCHISM, to make the stoichiometry move around with its OM, I made it a state variable, (actually 8 of them: sx, sy for 4 OMs, with all sz's fixed to 1), analogous to Qn, Qp moving around with their A's. This was still awkward and inconvenient, so when updating the CGEM-SCHISM code to work on a hybrid grid, I rewrote the tracking of stoichiometry as follows.

Chemistry equation is of the form
CxNyPz -> (products depending on ratios of x, y, and z)

The reaction subroutine takes CxNyPz(mmolC/m3) and calculates the rate of change for CxNyPz(mmolC/m3).

The code changes are as follows:
- Instead of tracking OM(mmolC/m3), along with ratios sx=C:P, and sy=N:P, track state variables OMC(mmolC/m3), OMN(mmolN/m3), OMP(mmolP/m3)
- Use OMC, OMN, OMP to calculate stoichiometry as input to the reaction subroutine
- Use stoichiometry to scale the output from the reaction subroutine, ROM, which is by default mmolC/m3
- Equation is OM = OM_old + (ROM + OM-generated-by-A-Z)*dt

e.g.,
```
!-----------------------------------------------------------
 sx1 = OM1CA/OM1PA
 sy1 = OM1NA/OM1PA
 sx2 = OM2CA/OM2PA
 sy2 = OM2NA/OM2PA
 call reaction( OM1CA, OM2CA, O2, NO3, KG1, KG2, KO2, KstarO2, KNO3,&
 & sx1, sy1, sz, sx2, sy2, sz, T, RC )
 RC = one_d_365 * RC !Change units from /year to /day
 ROM1CA = RC(1)     ! units are /m3/day
 ROM2CA = RC(2)
 ROM1NA = RC(1)*sy1/sx1
 ROM2NA = RC(2)*sy2/sx2
 ROM1PA = RC(1)/sx1
 ROM2PA = RC(2)/sx2
!-----------------------------------------
 !-OM1_A: (mmol-C/m3-- Dead Phytoplankton Particulate)
 ff_new(iOM1CA) = OM1CA + (ROM1CA + OM1_CA)*dTd
 ff_new(iOM1NA) = OM1NA + (ROM1NA + OM1_NA)*dTd
 ff_new(iOM1PA) = OM1PA + (ROM1PA + OM1_PA)*dTd
```

The code change from tracking stoichiometry ratios to tracking OMC, OMN, OMP did not appear to affect the results, which is what we expect; the new code does the same thing, only in a more efficient and less convoluted way.

Fluxes, Options, and Negative Values

I did not implement any fluxes other than O2 and DIC surface fluxes. The base CGEM-SCHISM code should be thoroughly tested before adding advanced options. The work involved with re-coding the subroutines is non-trivial and time consuming, and would require a great deal of testing.

To avoid confusion for end-users testing the code, the non-implemented flux options are removed from the namelists. The code defining the variables remains, but has been commented out, making it easier to add back when appropriate.

Before rewriting for the hybrid grid, the code contained untested drafts of most surface and bottom fluxes (but not SDM). The surface fluxes were re-coded and moved to surface_fluxes.F90. The bottom fluxes were *not* re-coded but are saved in bottom_fluxes.F90.stub, so that the next developer can finish the modifications.

Switches for the light model and carbon to chlorophyll have been removed, because we always use Brad's light model and we didn't trust the Cloern implementation. All other temperature and growth switches remain, although they have not been tested.

Instant remineralization remains as a 'flux' option, although it is not a flux, rather it modifies the remineralization rate for OMA and OMZ at the bottom layer. As an 'if' statement: if the cell is on the bottom, use `KGbot` instead of `KG`. I did a quick test of instant remineralization and found it created negative values. As it turns out, I had not added checks to the CGEM equations (cgem_step.F90) to check for negative values, but I had not had negative values in any testing - even in the bigger models - until trying instant remineralization. Since the timestep of 100 seconds is small enough that the biogeochem equations should never go negative, I left it as-is, and changed `KGbot` to a number significantly larger(x365) than `KG`, but not large enough to cause negative values. As a consequence, the option is now 'fast remineralization' rather than 'instant remineralization'. `KGbot` can be calibrated, or `DMIN(ff,f_min))` statements may be added. Note, however, that every time a negative value is reset to zero, it creates mass imbalance.

Since CGEM-SCHISM is in draft mode, we should anticipate finding errors during testing, and when there are bugs, you want the code to fail in a very noticeable way. If something goes negative, we should figure out why. Setting things to zero will mask these errors, potentially having end-users calibrate away the bugs. Other than instant remineralization, the only time I encountered negative values was when setting sinking rates higher than warranted by the courant condition. The cgem sinking routine (`sinkwcgem=.TRUE.`) checks for and has a 'stop' if `ff_new.lt.0`. With `sinkwcgem=.FALSE.`, SCHISM sinking is used (`wsett=ws`) and the code will probably happily run along with negative values until mocsy gets a bad value; mocsy has a stop statement. Check the outputs periodically with VisIt during a run.

Post-processing

Scripts are in the OyBcSt/CGEM github repository:
https://github.com/oybcst/CGEM.git.
The scripts used in the final testing is tagged v0.1 and can be downloaded here
https://github.com/oybcst/CGEM/releases/tag/v0.1

Outputs from the box model can be viewed here:
https://renc.osn.xsede.org/ees210015-bucket01/CGEM_TESTING/outputs_10.pdf
Animations from the cgem-SA-LSC2 test run are on YouTube playlist:
CGEM-SCHISM SA-LSC2

To create the movies, I used databases.py to create VisIt database files. A database file is a text file containing a list of files that VisIt should consider to be one database. There are 17 filenames (GEN_X_1:17.nc), each having 10 days of data. I used the s2p_run.sh script which uses ser2par (MPI wrapper to parallelize serial tasks) to run parallel_plot_cgem.py on 30 cores, one for each state variable, for surface and bottom. Data is hourly: there are 4080 images for the surface and bottom for each state variable. Creating the images took 2.3 hours on 30 cores.

The images are concatenated into movies with ffmpeg.py, which was run in serial but uses the entire node because of memory. Creating a movie took 5-6 minutes each, for about 5.5 hours in total. This would be faster on a GPU node, but we'd need to request GPU time and figure out how to compile ffmpeg correctly; it would be worth it if you create a lot of movies, but for one-offs, using CPUs is probably fine. Note, however, this was only for 170 days, not the full year. To make movies more quickly, reduce video quality (make crf higher) or generate fewer images (i.e., 1-4 per day, rather than hourly).

The full year model run was 22.6 hours on 256 cores. A significant portion of runtime is spent writing output files. For faster test runs, do not output every variable, or write outputs less often.

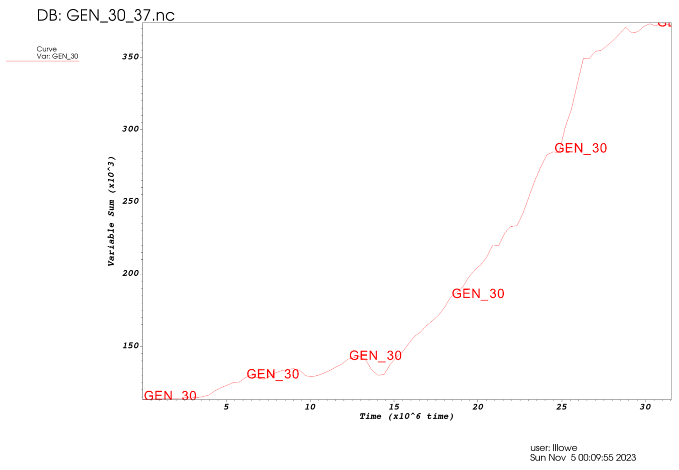
Testing Directories

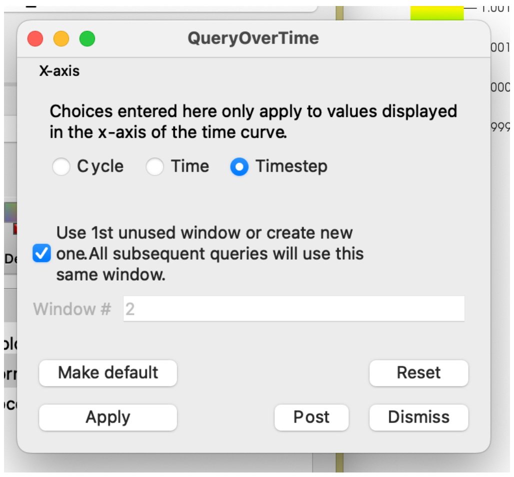
Files needed to replicate the test runs are on Expanse in `/home/llowe/CGEM_TESTING` and available for download via Open Storage Network(OSN) with `wget` or via web browser:
https://renc.osn.xsede.org/ees210015-bucket01/CGEM_TESTING/outputs_10.pdf
https://renc.osn.xsede.org/ees210015-bucket01/CGEM_TESTING/cgem-box.tar
https://renc.osn.xsede.org/ees210015-bucket01/CGEM_TESTING/cgem-SA-LSC2.tar
As a test of the code, the scripts, and the documentation, please compile the code and generate the initial conditions yourself. Start here:
https://github.com/oybcst/CGEM/blob/main/cgem-dev.md

Mass balance

This is a plot of the sum over the grid for each timestep of Tracer, with sinking rate 0.01, which seems to go off the rails after the video ends.

To create such a plot, open all files in VisIt as a database, do clipping and transform as usual, check the plot, set Query options to Time, and extract the Variable Sum at every 100th timestep. This issue needs further investigation.




QueryOverTime

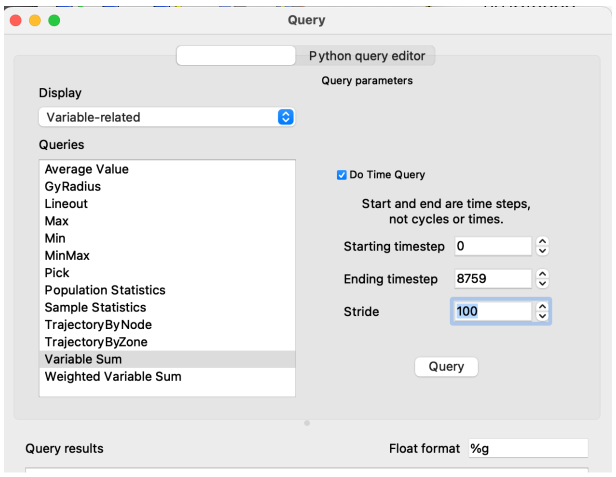
X-axis

Choices entered here only apply to values displayed in the x-axis of the time curve.

Cycle Time [✓ Timestep]

[✓] Use 1st unused window or create new one.All subsequent queries will use this same window.

Window # 2


Query

Display: Variable-related

Queries:
Average Value
GyRadius
Lineout
Max
Min
MinMax
Pick
Population Statistics
Sample Statistics
TrajectoryByNode
TrajectoryByZone
[✓ Variable Sum]
Weighted Variable Sum

[✓] Do Time Query

Start and end are time steps, not cycles or times.

Starting timestep 0

Ending timestep 8759

Stride 100

Query results

Float format %g

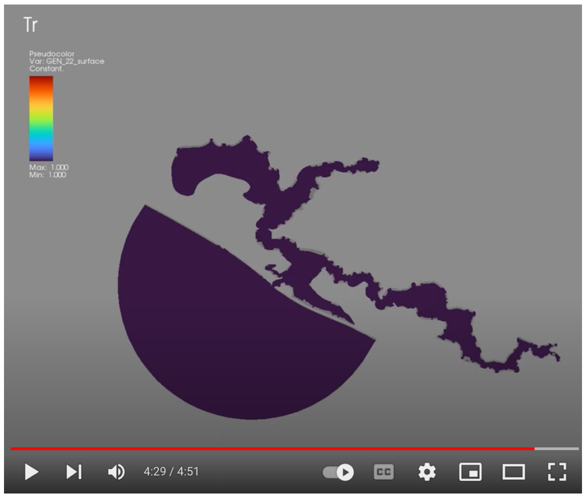
Additional issues

These are additional issues I encountered, in case it helps future developers to improve things.

Accounting for volume

Initial development was done on the ECO-Toy box model grid, having uniform temperature, salinity, elevation, and no velocity. Further development was done with a uniform layered grid of Saint Andrew’s Bay, Florida, created by Zhilong Liu. As expected, moving to a non-trivial grid gave us some difficulty. In particular, I had issues tied to the sinking rates. Typical sinking rates caused mass imbalance and instability. For development purposes, I set sinking to zero. Without sinking, CGEM seemed to work fine, meaning it appeared to behave reasonably and have the expected range of values, but I realized it was still *incorrect*.

In SCHISM, if all values are equal at the beginning of a simulation, they will remain so, mass being conserved. But CGEM values are concentrations, not mass. Here is the Tracer without sinking, which starts at 1 and stays constant over time at exactly 1. I think this is 'cheating'...no accounting for volume: Tracer - no change in volume



In original CGEM, this accounting for volume is done in the advection step, in Adv3D.F90, taking into account the current volume and previous volume:

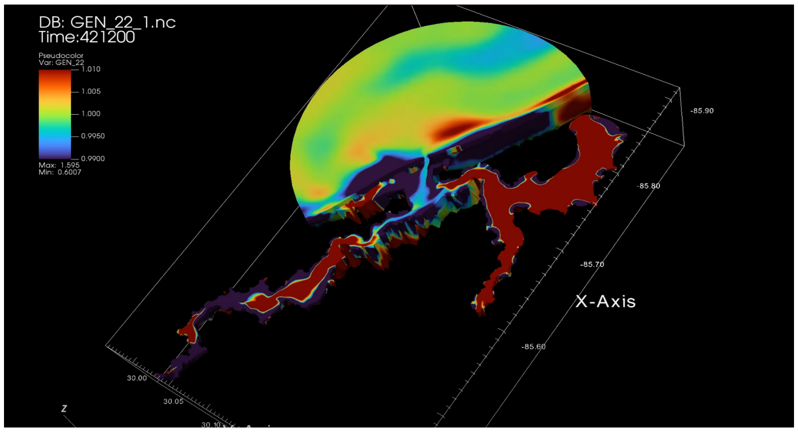
```
f_n(myi,j,k,ii) =                                  &
&    ( f(myi,j,k,ii)*Vol_prev(i,j,k)               &
&     +( cf*f(myi,j,k,ii)                          &
&     +(  (  (ufm*f(myim1,j,k,ii)+ufp*f(myip1,j,k,ii))    &
&       +(vfm*f(myi,jm1,k,ii)+vfp*f(myi,jp1,k,ii))  )    &
&  +(wfm*f(myi,j,km1,ii)+wfp*f(myi,j,min(k+1,nz),ii))) ) *dT &
&     )  / Vol(i,j,k)
```

I decided to add this piece via the `sinkwcgem` option by saving the previous volume (`Vol_prev`) and adding the term similar to the above (`Vol_prev/Vol`). But `Vol` is a huge number, and that just led to finding that we really needed to match SCHISM's double precision.

Looking through the other biogeochem models in SCHISM, I found code in ICM that looked like what I wanted to do, without using Volume (since the area cancels) and using native SCHISM code. Here is what I found that looked 'correct' to me, in SCHISM's Hydro/schism_step.F90, within a `#ifdef USE_ICM` statement, starting around line 7800:

```
!Inflation coef (ratio of volumes)
zrat=htot/dzz1
!if(abs(zrat-1)>rinflation_icm) cycle
!write(6,*) abs(zrat-1),htot,dzz1,zrat
!write(6,*) "htot,dzz1,zrat",htot,dzz1,zrat
do k=kbe(i)+1,nvrt
  tr_el(itmp1:itmp2,k,i)=tr_el(itmp1:itmp2,k,i)*zrat
enddo !k
```

I implemented the above, and at the time–although I couldn't replicate the results–it *seemed* to make things 'better', as far as the mass balance. Here is the Jellyfish Tracer which now accounts for volume. I ran for only a short amount of time for the movie.



In the code used for Tracers shown above, CGEM operated on data in columns, and assumed each column had the same number of layers. I did another modification to the code for it to work over a hybrid grid. The final test runs were performed on an LSC2 hybrid grid version of the St. Andrew's Bay model, also created by Zhilong Liu.

Once everything was working in general for the hybrid grid, I tried to add back the code from ICM to account for volume, but it didn't work, and with time constraints I could not investigate further. Since I was not able to replicate previous results, I'm not sure whether the change to account for volume actually improved the mass balance or whether the improvement was just the switch to double precision. I suspect the latter.

When I originally forked the SCHISM repo, the `USE_GEN` only showed how to add a sinking rate. In a newer version of SCHISM, it has code which mentions 'for concentration'. This is in Hydro/schism_step.F90, `#ifdef USE_GEN` comments on how to add stuff:

```
!-!Use upwind prism for concentration
! if(k>kbe(i)+1)
! tmp=tmp-wwint*tr_el(m,k,i)/(ze(k,i)-ze(k-1,i))
! if(k<nvrt) tmp=tmp+wwint*tr_el(m,k+1,i)/(ze(k,i)-ze(k-1,i))
```

It looks like this 'accounting for volume' is something everyone already knows about, and we will ask for advice moving forward.

I have added the 'uniform layers' draft version of cgem_run that accounted for volume to the repo: cgem_run_volprev.F90.stub. The movies from the test run do not account for volume.

SCHISM-native implementation

Originally, I implemented CGEM in a SCHISM-native way, using `wsett`, `bdy_frc`, `flx_sf`. (I did not get to `flx_bt`). I believe this is the preferable way to implement a tracer model. The implementation worked fine with the box model, but it was unstable with a non-trivial grid. Again, for development purposes, I implemented things 'my way', so I had more control over things and understood exactly what was going on.

To explain the above:
Integration is `y_new = y_old + f(y_old)*dt`

For the SCHISM-native integration, you would calculate `f(y_old)`, then set `bdy_frc=f(y_old)` and SCHISM would integrate things for you.

For my implementation, I calculated the entire `y_new` and set `bdy_frc=0`.

COSINE uses `bdy_frc`. ICM calculates `y_new`.

It is difficult to troubleshoot a complicated bio model with over 30 variables on a non-trivial grid. Before further CGEM development, we will investigate the stability issues further by using SCHISM's general tracer module with different sinking rates.

Acknowledgements

Thanks!!

The Mobile Bay Modeling Team

Members of NOAA RESTORE's Oysters, Blue Crabs, Seatrout project(OyBcSt) hydrodynamics modeling team, alphabetically listed:
Brian Dzwonkowski - Senior Marine Scientist II, Dauphin Island Sea Lab; PI NOAA RESTORE
John Lehrter - Senior Marine Scientist, Dauphin Island Sea Lab; Lead PI, NOAA RESTORE
Zhilong Liu - Research Assistant Professor, University of South Alabama
Aravind Puzhankara - PhD Student, University of South Alabama
Harikrishnan Sreeshylam - PhD Student, University of South Alabama

Zhilong Liu developed the hydrodynamics models that were used for testing CGEM.

A project portfolio for the OyBcSt hydrodynamics modeling is available here:
https://oybcst.github.io/intro.html

US EPA

Thanks to Brandon Jarvis, Wilson Melendez, Cody Simmons, and all other developers of the original CGEM code, shared in the spirit of open-source science.

- Information: https://www.epa.gov/ceam/coastal-generalized-ecosystem-model-cgem
- Code: https://github.com/USEPA/CGEM

NOAA

The hydrodynamics modeling and CGEM development work is a component of research funded by the National Oceanic and Atmospheric Administration's RESTORE Science Program under award NA19NOS4510194 to University of South Alabama and North Carolina State University.

NSF

The hydrodynamics modeling for this project is done on Expanse, at the San Diego Supercomputer Center (SDSC), and the modeling data is being hosted on Open Storage Network (OSN), through allocation EES210015 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

Work at both providers was done under the Extreme Science and Engineering Discovery Environment (XSEDE), which was supported by National Science Foundation grant number

#1548562. We thank Mark W. Van Moer of the National Center for Supercomputing Applications (NCSA) for advanced visualization support which was made possible through the XSEDE Extended Collaborative Support Service (ECSS) program.

SCHISM

SCHISM (Semi-implicit Cross-scale Hydroscience Integrated System Model) is a derivative product built from the original SELFE (v3.1dc; Zhang and Baptista 2008) and distributed with an open-source Apache v2 license, with many enhancements and upgrades including new extension to large-scale eddying regime and a seamless cross-scale capability from creek to ocean (Zhang et al. 2016).
- Zhang, Y. and Baptista, A.M. (2008) SELFE: A semi-implicit Eulerian-Lagrangian finite-element model for cross-scale ocean circulation", Ocean Modelling, 21(3-4), 71-96.
- Zhang, Y., Ye, F., Stanev, E.V., Grashorn, S. (2016) Seamless cross-scale modeling with SCHISM, Ocean Modelling, 102, 64-81.

SCHISM code: https://github.com/schism-dev/schism

SCHISM VisIt Plugin: https://github.com/schism-dev/schism_visit_plugin

VisIt

Visualizations were created with VisIt, an Open Source, interactive, scalable, visualization, animation and analysis tool: https://visit.llnl.gov

Childs, H., Brugger, E., Whitlock, B., Meredith, J., Ahern, S., Pugmire, D., Biagas, K., Miller, M. C., Harrison, C., Weber, G. H., Krishnan, H., Fogal, T., Sanderson, A., Garth, C., Bethel, E. W., Camp, D., Rubel, O., Durant, M., Favre, J. M., & Navratil, P. (2012). High Performance Visualization--Enabling Extreme-Scale Scientific Insight. https://doi.org/10.1201/b12985

You!

Thank you for your kind attention.