Environment and Climate Change Canada   Environnement et Changement climatique Canada

Canada

Towards semi-Lagrangian advection in NEMO

Christopher Subich, Pierre Pellerin,
Greg Smith, Frederic Dupont

Christopher.Subich@canada.ca

21 October 2020

Outline

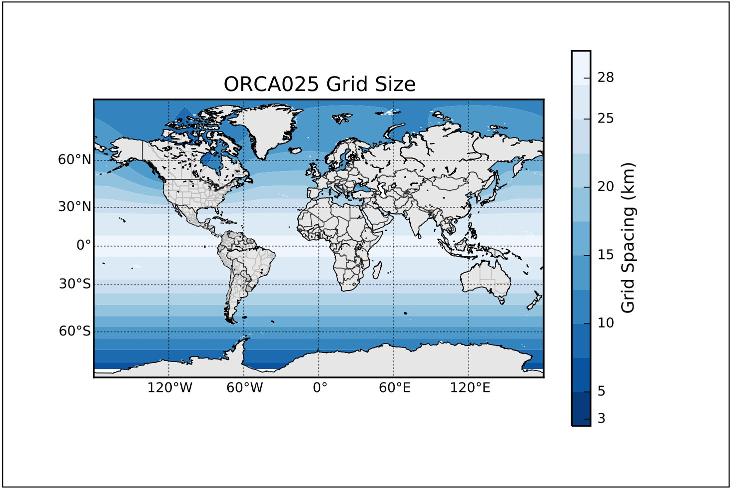
Introduction & Motivation

- Canadian Meteorological Centre runs a number of coupled atmosphere/ocean forecasting systems
- Resolutions increasing with time
  - Atmospheric system moved from 25 to 15km resolution ($1/4^\circ$ ocean)
  - Coupled ensembles in the works
- Coupled forecasting systems are *expensive*
  - Would help if we could increase the coupled timestep
  - GEM (atmospheric model) already has semi-Lagrangian advection, why not try this in NEMO also?
- Objective: *to develop a semi-Lagrangian advection scheme for NEMO that allows us to increase the timestep in operational configurations*
  - …while retaining compatibility with ongoing NEMO development
  - …and while maintaining or improving accuracy

Semi-Lagrangian challenges in NEMO

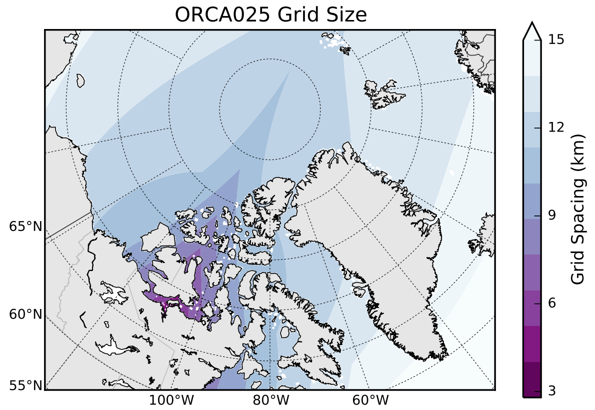
- Grid:
  - Z-level grid (in coupled forecasting system) with partial cells at the ocean-bottom layer
  - Non-uniform resolutions, with grid stretching in both horizontal and vertical directions
  - Staggering of momentum and tracer points
- Boundaries:
  - Free surface, bottom, and lateral boundaries
  - Interactions between lateral boundaries and grid staggering
- Math:
  - NEMO (currently) structured around *leapfrog* timestepping
  - Expects advection to be just one of many forcings

ORCA grid



- “Tripolar” ORCA grid at nominal $\frac{1}{4}^{\circ}$ resolution

ORCA grid



- "Tripolar" ORCA grid at nominal $\frac{1}{4}^{\circ}$ resolution

Semi-Lagrangian time-stepping

- $\frac{Df}{Dt} + g = 0$
- $\frac{D\vec{x}}{Dt} = \vec{u}$
  - Continuous, Lagrangian representation following the flow ($\frac{D}{Dt}$)
  - Fluid parcels ($\vec{x}$) definitionally follow the local velocity ($\vec{u}$).
- $f^A = f^D - \frac{\Delta t}{2}(g^A + g^D)$
- $\vec{x}^A = \vec{x}^D + \frac{\Delta t}{2}(\vec{u}^A + \vec{u}^D)$
  - Fluid properties at arrival point ($\vec{x}^A$) governed by departure-point f^D and forcing over the trajectory
  - Arrival/departure points determined by local velocities $\rightarrow$ implicit relationship to solve
  - *Semi*-Lagrangian takes $\vec{x}^A = \vec{x}^{ref}$ as the grid & solves for x^D
  - Requires off-grid interpolation at each timestep
  - Finite difference framing of equations

Leapfrog in NEMO

- $$\frac{\partial f}{\partial t} = g$$
  - NEMO takes an Eulerian, finite-volume view of flow
  - Fluid properties always expressed over static (z-coordinate!) locations, but forcing *G* includes advective fluxes
- $$\frac{f^A}{2\Delta t} = \frac{f^B}{2\Delta t} + g^N$$
  - Semi-discretized via leapfrog method
  - Properties “after” $(\cdot)^A$ are governed by properties “before” $(\cdot)^B$ and forcing “now” $(\cdot)^N$
  - Explicit timestepping, no need for iteration
- How does a semi-Lagrangian method fit in this framework?
  - Split the advection operator and match the product

Semi-Lagrangian leapfrog

- Consider tracer flow with *only* advection:
  - Tracers conserved following a fluid parcel
- (Semi-)Lagrangian: $\dfrac{Df}{Dt} = 0$
  - Define arrival and departure points
  - Take arrival at “after” time-level, departure at “before”
- Eulerian: $\dfrac{\partial f}{\partial t} + \vec{u} \cdot \nabla f = 0$
  - Discretize with “after”, “before”, and “now” levels
  - Solve for “after” tracer
  - Define the (*trend*) term that NEMO needs
- Equate: $(trend) = \dfrac{1}{2\Delta t}(f^D - f^B)$
- Semi-Lagrangian advection gives a time-trend that looks just like any other advective process

Semi-Lagrangian leapfrog

- Consider tracer flow with *only* advection:
  - Tracers conserved following a fluid parcel
- (Semi-)Lagrangian: $f^A = f^D$
  - Define arrival and departure points
  - Take arrival at “after” time-level, departure at “before”
- Eulerian: $\dfrac{f^A - f^B}{2\Delta t} + \vec{u}^N \cdot \nabla f^N = 0$
  - Discretize with “after”, “before”, and “now” levels
  - Solve for “after” tracer
  - Define the (*trend*) term that NEMO needs
- Equate: $(trend) = \dfrac{1}{2\Delta t}(f^D - f^B)$
- Semi-Lagrangian advection gives a time-trend that looks just like any other advective process

Semi-Lagrangian leapfrog

- Consider tracer flow with *only* advection:
  - Tracers conserved following a fluid parcel
- (Semi-)Lagrangian: $f^A = f^D$
  - Define arrival and departure points
  - Take arrival at "after" time-level, departure at "before"
- Eulerian: $f^A = f^B - 2\Delta t(\vec{u}^N \cdot \nabla f^N)$
  - Discretize with "after", "before", and "now" levels
  - Solve for "after" tracer
  - Define the (*trend*) term that NEMO needs
- Equate: $(trend) = \frac{1}{2\Delta t}(f^D - f^B)$
- Semi-Lagrangian advection gives a time-trend that looks just like any other advective process

Semi-Lagrangian leapfrog

- Consider tracer flow with *only* advection:
  - Tracers conserved following a fluid parcel
- (Semi-)Lagrangian: $f^A = f^D$
  - Define arrival and departure points
  - Take arrival at "after" time-level, departure at "before"
- Eulerian: $f^A = f^B + 2\Delta t(\mathit{trend})$
  - Discretize with "after", "before", and "now" levels
  - Solve for "after" tracer
  - Define the (trend) term that NEMO needs
- Equate: $(\mathit{trend}) = \frac{1}{2\Delta t}(f^D - f^B)$
- Semi-Lagrangian advection gives a time-trend that looks just like any other advective process

Semi-Lagrangian leapfrog

- Consider tracer flow with *only* advection:
  - Tracers conserved following a fluid parcel
- (Semi-)Lagrangian: $f^A = f^D$
  - Define arrival and departure points
  - Take arrival at “after” time-level, departure at “before”
- Eulerian: $f^A = f^B + 2\Delta t(\mathit{trend})$
  - Discretize with “after”, “before”, and “now” levels
  - Solve for “after” tracer
  - Define the (trend) term that NEMO needs
- Equate: $(\mathit{trend}) = \frac{1}{2\Delta t}(f^D - f^B)$
- Semi-Lagrangian advection gives a time-trend that looks just like any other advective process

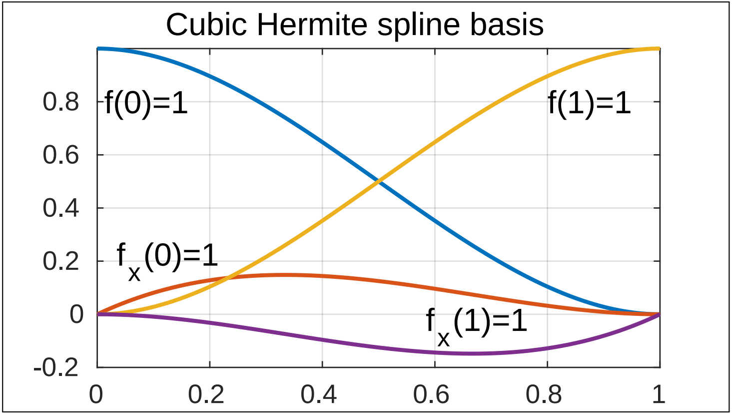
Related questions & answers

- Where did the "now" fields go?
  - f^n genuinely disappears
  - $\vec{u}^n$ is defines trajectories – effectively a frozen, time-centered flow
- What about other forcing?
  - Preserve NEMO's computation of all non-advection terms
  - Effectively operator splitting – no interaction between semi-Lagrangian advection and other forcing terms
- What about conservation?
  - Classic advection routines discretize with finite-volume form, conserving tracers following (incompressible) flow
  - Potential for non-conservation via interpolation – semi-Lagrangian implicitly uses a finite-difference framework
- What about velocity?
  - Velocity components are *not* left unchanged following motion
  - . . . but NEMO has separate forcing for Coriolis forces and metric terms
  - Semi-Lagrangian advection replaces *flux form* momentum advection schemes

Interpolation

- Key problem: compute f^D, off-grid interpolation of “before” fields
- Tradeoff between interpolation error and stencil size/computational work
  - Three-dimensional interpolation
  - $4 \times 4 \times 4$ stencil can exactly reproduce cubic polynomials
- Split interpolation by grid dimension, and apply repeated 1D interpolation schemes
  - Interpolate first in vertical, then in horizontal
  - Better compatibility with z-level coordinate system and partial cells
- Base interpolation on cubic Hermite splines

Cubic Hermite splines



- Basis functions have $f = \pm 1$ or $f_x = \pm 1$ at the endpoints
- 4-point finite difference stencils for derivatives reproduce Lagrange interpolation

Vertical interpolation

- Take $\vec{x}^D = (x_d, y_d, z_d)$
- Vertical interpolation finds $F(x_i, y_j, z_d)$, for x_i, y_j at grid points inside 4×4 stencil
- Also masks points inside land boundary
- Vertical interpolation needs derivative continuity – 4-point stencil has discontinuous derivatives at grid points
- Schematic: oscillatory motion
  - Fluid parcel goes down by ϵ, $f(x_i, y_j, z_k)$ decreases by ϵF_z^-
  - Fluid parcel goes up by ϵ, $f(x_i, y_j, z_k)$ increases by ϵF_z^+
  - Net drift proportional to the difference in one-sided derivatives, acts like vertical diffusion
- Solution: use *3*-point central stencil to precompute f_z

Horizontal interpolation

- After vertical interpolation: we have $F(x_i, y_j, z_d)$ and want $F(x_d, y_d, z_d)$
- Repeat dimension splitting: interpolate in 1D to $F(x_d, y_j, z_d)$, then $F(x_d, y_d, z_d)$
- Horizontal flow is less oscillatory, more driven by mean currents and long-lived eddies
- Use more accurate, one-sided stencils for endpoint derivatives
  - Full fit of 3rd-order polynomial to 4 points
  - Minimizes numerical diffusion
- Further approximation: interpolate on grid-index basis
  - Avoids complications from horizontal coordinate transformations
  - Justified because grid changes slowly over the horizontal interpolation stencil
- Boundaries (horizontal and vertical) implemented by symmetry conditions

Limiting

- So far, interpolation has been defined without limiting
- Most accurate specification, but allows for new minima/maxima
- Undesirable, and early testing showed potential for instability with tracer overshoots near lateral boundaries
- Method implements weak limiting:
- Horizontal:
  - If $f(0)$ is a local minima, then $f'(0) \leftarrow \min(0, f'(0))$
  - If $f(0)$ is a local maxima, then $f'(0) \leftarrow \max(0, f'(0))$
  - Symmetric conditions on $f'(1)$
- Overshoots still possible in the middle of the interval, but these do not appear to cause problems

Vertical limiting

- Limiting everywhere is far too diffusive in the vertical direction
- Vertical interpolation is *not* limited, save near boundaries
- Limiting called for near partial cells, somewhat ad hoc:
  - If (x_i, y_j, z_d) corresponds to a partial cell with thickness $>$ 175% of its thinnest neighbour, strictly limit vertical interpolation to forbid an overshoot
  - Helps prevent an observed problem of deep-ocean cells developing extraordinary temperatures/salinities ($< -10^\circ$!) when partial-cell topography prevents lateral flow
  - Limiting everywhere in bottom layer would diffuse stratified flow near a sloping ocean bottom
- In progress question: whether limiting is necessary for all fields (current implementation) or tracers only

Trajectories

- Interpolation is half the problem
- Evaluating $f(\vec{x}^D)$ requires some way of specifying the departure points
- Lagrangian equation of motion: $\frac{D\vec{x}}{Dt} = \vec{u}(\vec{x}, t)$
- Want consistency with leapfrog timestepping
  - Freeze the flow, so RHS is $\vec{v}(\vec{x}, t_n)$ based on "now" timestep
  - Time-centered approximation
- Still face an iterative problem to solve for departure points
- Traditional approach: trapezoidal rule
- $\vec{x}^D \approx \vec{x}^A - \frac{\Delta t}{2}(\vec{u}^N(x) + \vec{u}^B(\vec{x}^D))$

Environment and Climate Change Canada
Environnement et Changement climatique Canada
Canada

The boundary problem

- Trapezoidal rule has a problem near lateral boundaries
- Trajectories must never cross boundaries – no from-land advection
  - Not guaranteed by trapezoidal calculation of trajectories
  - No robust way to fix this, e.g. with velocity extrapolation
  - Special case of Lipschitz trajectory-crossing criterion
- Solution: approximate the velocity field, but integrate *exactly* in time

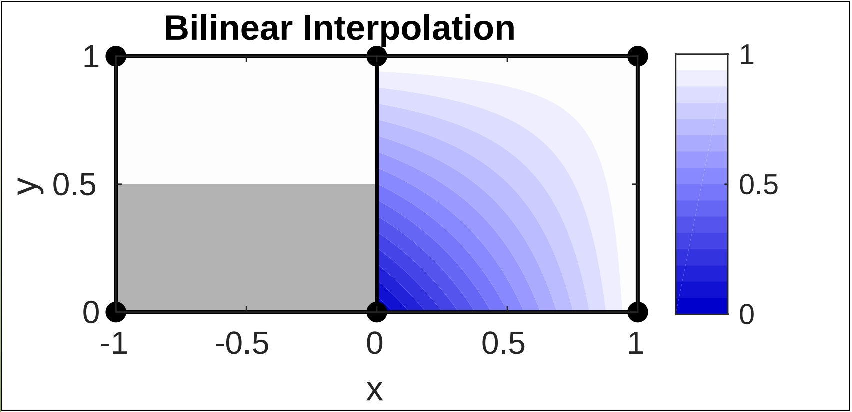
Exponential trajectories

- $\frac{d\vec{x}}{dt} = \vec{u}(\vec{x})$
- Analytic solution exists if $\vec{u}$ varies linearly
- Linearly-varying field can be constructed from two measurements
  - We *have* two measurements: $\vec{u}^A$ and $\vec{u}$ at a candidate departure point
  - Works perfectly inside trajectory iteration
- Physical intuition: fluid parcel arrives at $\vec{x}^A$ tangent to $\vec{u}^A$, defining a rotated coordinate axis
- $\vec{u} \approx \vec{u}^A + (\vec{u}^D - \vec{u}^A)\dfrac{(\vec{x} - \vec{x}^A) \cdot \vec{u}^A}{(\vec{x}^D - \vec{x}^A) \cdot \vec{u}^A}$
- Analytically solvable, with solution in terms of exponentials
- Speed optimization: trilinear interpolation to find $\vec{u}^D$ in trajectory calculations

A corner case

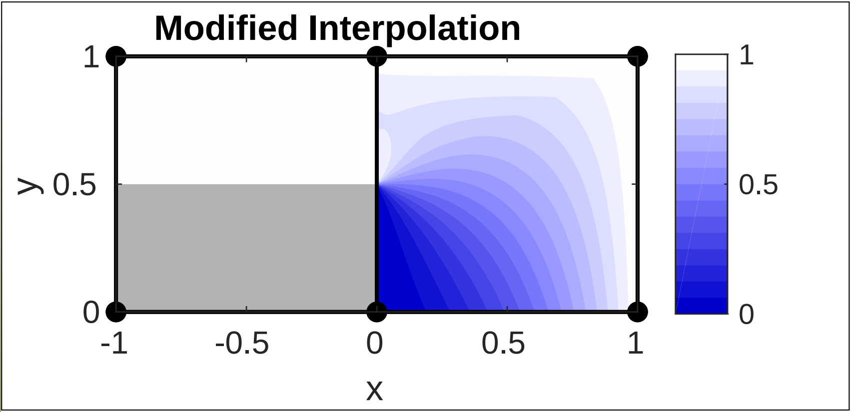
- Bilinear interpolation of velocities breaks at lateral boundary corners
- Product of grid staggering:
  - The full tracer-cell is either water or land
  - Velocity points are staggered by $\frac{1}{2}$ cell
  - From perspective of velocity points, boundary can be $\frac{1}{2}$ water, $\frac{1}{2}$ land.
- Bilinear interpolation breaks no normal-flow boundary condition, causes discontinuities at cell edges
  - Fictitious normal flow: trajectories try to converge inside boundary
  - Large cell-edge discontinuities: poor convergence of iteration
- Incorporate corner into interpolation with blended solution:
  - Bilinear interpolation: good away from the wall
  - Singularity solution (corner function): good near the wall, with angle dependence

A corner case



- Bilinear interpolation of velocity: boundary inconsistency and discontinuities at edges

A corner case



- Modified interpolation: boundary consistency and weaker discontinuities

Flow past a box

- Difficult to look at time-stepping stability in isolation
- Full ocean mixes different modes:
  - Surface wave modes
  - Baroclinic internal waves
  - Ice processes
  - Explicit lateral diffusion
  - *Advection* – the only change here
- Look at a simpler, theoretical test case: isothermal flow past a box
- Primarily test of stability for momentum advection; other influences negligible

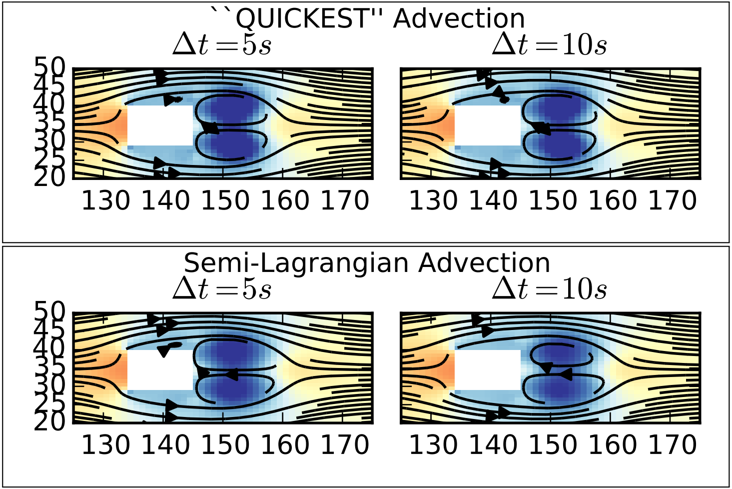
Flow past a box

Problem setup

- Domain:
  - $280 \times 70 \times 3$ grid, nearly two-dimensional
  - $\Delta x = \Delta y = 5\text{m}$, 30m “ocean” depth
  - 10×10 box ($50 \times 50\text{m}$) masked in center of domain
  - Initial and far-field flow of $\|\vec{v}\| = 3\text{cm}/\text{s}$
  - Run to final time of 8000s
- Control: traditional advection of momentum
  - Flux-form advection operator with QUICKEST scheme (best-behaving of NEMO advection schemes)
  - Slope limited, so no explicit diffusion of momentum
  - Implicit, linear free surface
- Semi-Lagrangian advection:
  - Semi-Lagrangian advection of momentum to u and v points
  - w unmodified, computed via hydrostatic approximation
- Flow sets up recirculation cells behind the box, over long time would develop a Von Karman vortex street

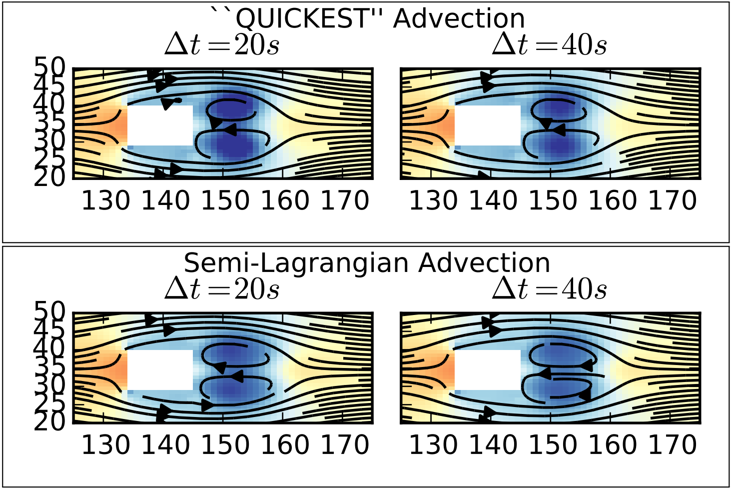
Flow past a box

Results



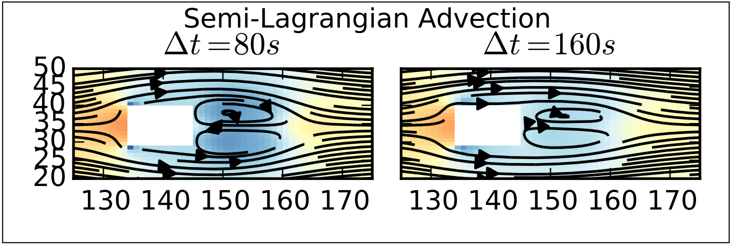
Flow past a box

Results



Flow past a box

Long-timestep results



- Control run unstable with $\Delta t = 80$s, semi-Lagrangian stable to $\Delta t = 160$s
- Steady-state Courant number > 1, higher with initial transients
- Flow strongly accelerated near leading edge of box, handled sensibly (if diffusively) with semi-Lagrangian method

NEMO 3.1 runs

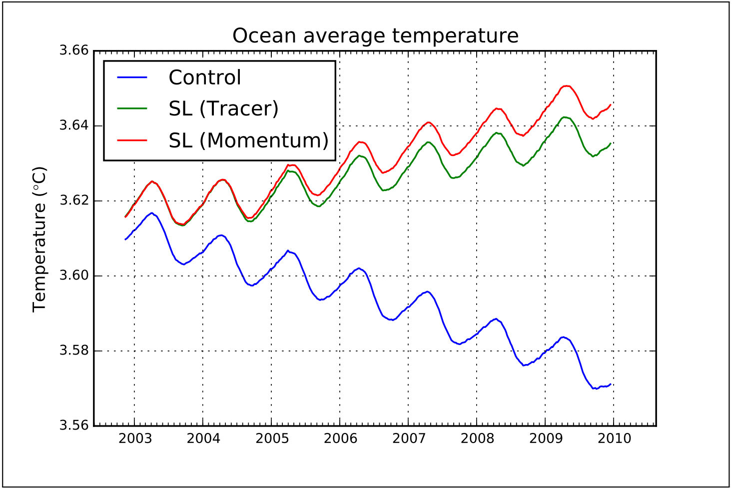
- Method initially implemented in NEMO 3.1 (based on CMC coupled forecast configuration)
- 10-year free runs, initialized October 1, 2001 with ocean at rest
- Atmospheric forcing given by 0.25° global reforecast (uncoupled)
- ORCA025 grid, CICEv4 ice modeling, 50 vertical levels
  - Implicit, linear free surface
- Objectives – proof of concept
  - Test for any conservation issues, especially for tracers
  - Start performance measurements
  - Find bugs in specification or implementation
  - Examine any qualitative differences in output

NEMO 3.1 runs

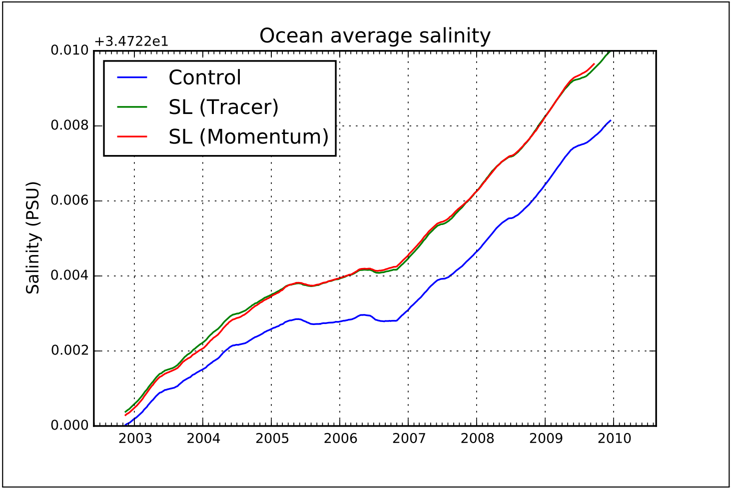
Three cases

- Control:
  - TVD advection of temperature and salinity
  - EEN (Energy and Enstrophy Conserving) vector-form advection of velocities
  - Lateral, iso-neutral Laplacian diffusion of $300\mathrm{m}^2/\mathrm{s}$ for tracers
  - $-3 \cdot 10^{11}\mathrm{m}^4/\mathrm{s}$ horizontal Bilaplacian diffusion of momentum
  - 600s timestep (limited by strong ice/ocean drag coupling)
- Semi-Lagrangian tracer:
  - Semi-Lagrangian advection of *only* tracers
  - Zero explicit diffusion of tracers
- Fully semi-Lagrangian:
  - Also semi-Lagrangian advection of momentum
  - 900s timestep (longer caused difficulties in ice dynamics)

Conservation – temperature

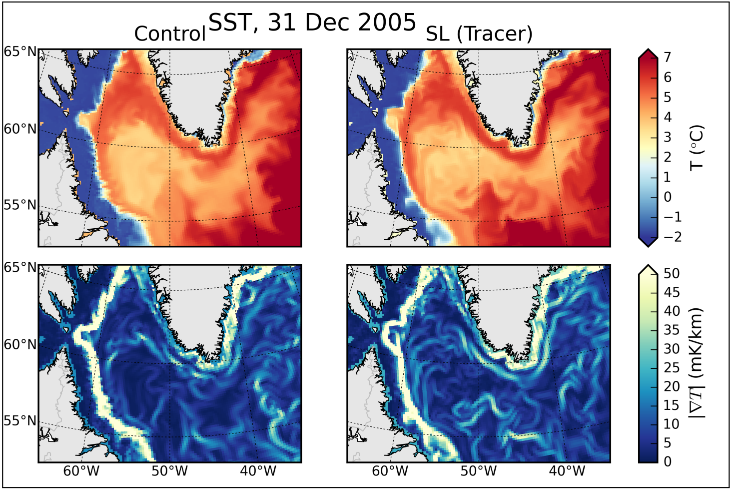


Conservation – salinity



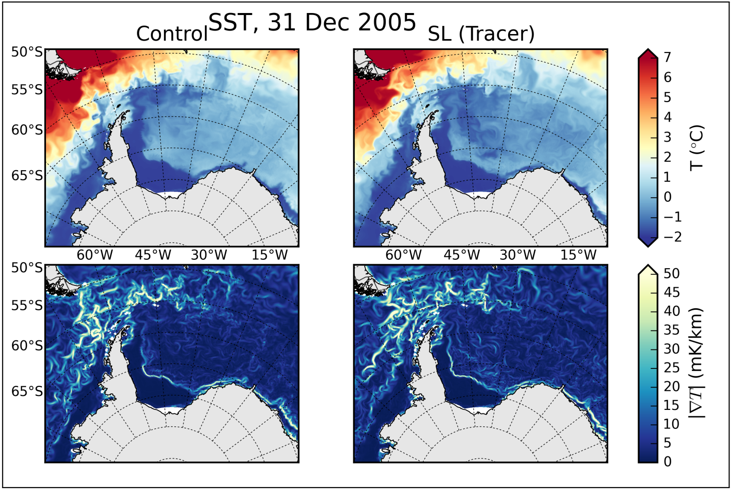
Qualitative results

Labrador Sea

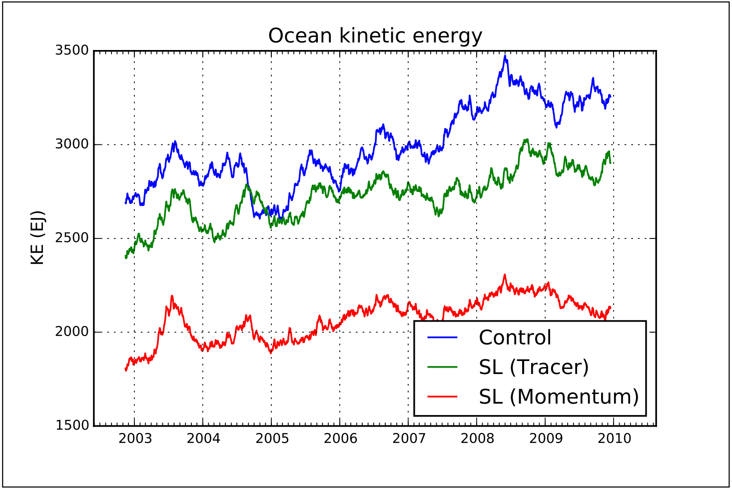


Qualitative results

Weddell Sea



Kinetic energy



Conclusions

- Semi-Lagrangian advection provides reasonable conservation of temperature and salinity, despite no explicit guarantee
  - Good conservation within layers, not just globally
  - Please be careful before trying this in very long-running climate simulations
- The method is stable without explicit diffusion for tracers
  - Potential for improvements in effective resolution (needs further analysis)
- Semi-Lagrangian advection of momentum has a surprisingly large effect on energy budget
  - Focus of ongoing work in NEMO 3.6
- *Still a significant performance penalty*, about 1hr/5d compared to 30min/5d – but room to optimize
- More detail recently published in GMD:
  - *Development of a semi-Lagrangian advection scheme for the NEMO ocean model (3.1)*

Conclusions & Future Work

- Semi-Lagrangian advection in NEMO is generally successful
  - Meets major goal of allowing a longer timestep
  - Does not cause large conservation errors
  - Optimistic signs for reducing salinity/temperature diffusion
- Goal: implementation in the forecast setting
  - Coupled global forecast – similar to this code-base; also coupled ensembles
  - Regional models – needs extension to allow for tides (variable vertical grid)
  - Comparison with ALE coordinates
- Contribution back to NEMO trunk
  - “Just another advection scheme” design
  - May need tweaks for RK3 timestepping