Imperial College London

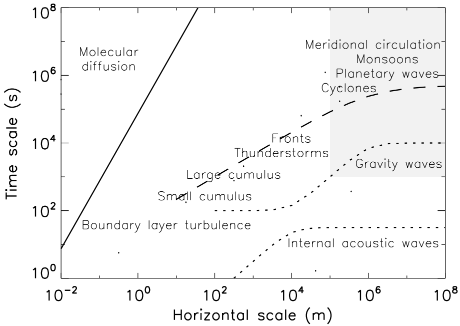
Time (integrator) parallel exponential integration and phase-averaging for geophysical fluid dynamics

Colin Cotter

September 28, 2019

Imperial College London

Timescales in atmospheric flows



Linear shallow water equations

$$\boldsymbol{u}_t + \underbrace{f\boldsymbol{u}^{\perp}}_{=f\boldsymbol{k}\times\boldsymbol{u}} + g\nabla\eta = 0,$$

$$\eta_t + H\nabla\cdot\boldsymbol{u} = 0. \qquad [D = H + \eta]$$

For constant f, H, g,

$$f\boldsymbol{u}^{\perp} = -g\nabla\eta \implies \nabla\cdot\boldsymbol{u} = 0.$$

Eliminating $\boldsymbol{u}$,

$$\underbrace{\frac{\partial}{\partial t}}_{\text{SLOW}} \underbrace{\left(\frac{\partial^2}{\partial t^2}h + \left(f - gH\nabla^2\right)h\right)}_{\text{FAST}} = 0.$$

Imperial College London

Quasigeostrophic shallow water equations

$$\boldsymbol{u}_t + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + f\boldsymbol{k}\times\boldsymbol{u} = -g\nabla\eta,$$
$$\eta_t + \nabla\cdot(\boldsymbol{u}(\eta+b)) = 0. \qquad [D = \eta + H + b].$$

For Ro $= U/fL$, assume $f\boldsymbol{k}\times\boldsymbol{u} - g\nabla\eta = \mathcal{O}(\text{Ro})$, $\eta/H = \mathcal{O}(\text{Ro})$, $b/H = \mathcal{O}(\text{Ro})$, $(f-f_0)/f_0 = \mathcal{O}(\text{Ro})$.

Then, to $\mathcal{O}(\text{Ro}^2)$, we have

$$\frac{\partial q}{\partial t} + \boldsymbol{u}\cdot\nabla q = 0, \quad \boldsymbol{u} = \nabla^\perp\psi, \quad \nabla^2\psi - \frac{gH}{f_0}\psi = qH + f + \frac{b}{H}.$$

Phase transformation

$$\boldsymbol{u}_t = -f\boldsymbol{k}\times\boldsymbol{u} - g\nabla\eta - (\boldsymbol{u}\cdot\nabla)\boldsymbol{u},$$
$$\eta_t = -H\nabla\cdot\boldsymbol{u} - \nabla\cdot(\boldsymbol{u}(\eta + b - H)).$$

Abstractly,

$$U_t = LU + N(U).$$

Rewrite

$$V_t = \exp(-Lt)N(\exp(Lt)V), \qquad [V = \exp(-Lt)U],$$

where $\exp(Lt)W$ is solution at time t to

$$\frac{\partial U}{\partial t} = LU, \quad U(0) = W.$$

Schochet, Embid and Majda

$$V_t = \exp(-Lt)N(\exp(Lt)V), \qquad [V = \exp(-Lt)U],$$

Phase averaging approximation,

$$V_t = \lim_{T\to\infty}\frac{1}{2T}\int_{-T}^{T}\exp(-Ls)N(\exp(Ls)V(t))\,\mathrm{d}s.$$

Embid and Majda (1998) showed (using work of Schochet) that taking the limit $\mathrm{Ro} \to 0$ recovers the quasi-geostrophic approximation of the shallow water equations (even with unprepared initial data).

Imperial College London

Balance models for NWP?

Why aren't balanced models used for NWP?

1. The approximations aren't uniformly valid.
2. At finite Ro, fast motions do couple back to the slow dynamics through near-resonances.

Alternative routes for NWP

NWP needs large efficient timesteps to get the forecast out on time.

Current alternatives to balanced models:

- Semi-implicit timestepping
- Split-explicit timestepping
- Vertically-implicit timestepping.

Another alternative

Haut and Wingate (2014) proposed to use a finite scale version of the phase average, implemented in parallel.

Finite scale phase averaging

$$V_t = \frac{1}{2T}\int_{-T}^{T} \rho(s/T)\exp(-Ls)N(\exp(Ls)V(t))\,\mathrm{d}s.$$

Replace integral by sum.

$$V_t = \sum_{m=-M/2}^{M/2} w_m \exp(-Ls_m)N(\exp(Ls_m)V(t)), \quad s_m = \frac{mT}{M}. \quad (1)$$

The terms in this sum can be evaluated independently in parallel.

1. Large T: fast oscillations due to L are filtered and we can take a large timestep in the corresponding ODE integrator for (1).
2. $\epsilon \to 0$ at fixed T: recover the quasigeostrophic approximation.
3. $T \to 0$: recover original equations.

Averaging the time-integrator

Strang splitting:

$$U^{n+1} = \Phi\left(\exp(L\Delta t)U^n\right),$$

where Φ is a timestepper for $U_t = N(U)$. Writing $\Phi = \mathrm{Id} + \Delta\Phi$,

$$U^{n+1} = \exp(L\Delta t)\left(U^n + \exp(-L\Delta t)\Delta\Phi\left(\exp(L\Delta t)U^n\right)\right).$$

Always average the equation, not the solution.

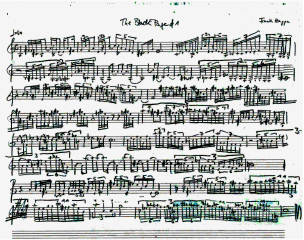
Phase-averaging:

$$U^{n+1} = \exp(L\Delta t)\left(U^n + \sum_{m=-M/2}^{M/2} w_m \exp(-Ls_i)\Delta\Phi\left(\exp(Ls_i)U^n\right)\right).$$

How to implement $\exp(Lt)U$?

L skew-adjoint, $L = UDU^T \implies \exp(Lt) = U\exp(Dt)U^T$.

- Rational approximations - see Dave's talk.
- (skew-)Krylov subspace methods - see work of Chad Sockwell.
- Chebyshev polynomials - I'm using these.



Chebyshev exponentiation

- Chebyshev approximation[1]: $\exp(is) \approx \sum_{k=0}^{N} a_k T_k(s)$, where T_k are Chebyshev polynomials transformed to create approximation on interval $[-iS, S]$. $(S > |\lambda_{\max}|T)$.
- Recurrence relation: $T_0(s) = 1$, $T_1(s) = -is/S$, $T_n(s) = 2sT_{n-1}(s)/(iS) - T_{n-2}$.
- Action of matrix exponential $\exp(tL)U \approx \sum_{k=0}^{N} a_k T_k(tL)U$.
- Build $T_k(tl)U$ recursively using $T_0(tL)U = U$, $T_1(tL)U = -itLU/S$, $T_n(tL)U = 2tT_{n-1}(tL)LU/(iS) - T_{n-2}(tL)U$.
- Application of L requires solution of mass matrices.
- Larger S (higher resolution or bigger T) requires more terms.

[1] Can do this with any matrix function, not just exp

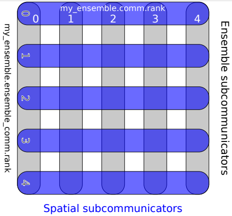
```
for i in range(2, ncheb+1):
    Tm2_r.assign(Tm1_r)
    Tm2_i.assign(Tm1_i)
    Tm1_r.assign(T_r)
    Tm1_i.assign(T_i)

    #Tn  = 2*t*A*Tnm1/(L*1j) - Tnm2
    operator_in.assign(Tm1_r)
    operator_solver.solve()
    T_i.assign(operator_out)
    T_i *= -2*t/L
    operator_in.assign(Tm1_i)
    operator_solver.solve()
    T_r.assign(operator_out)
    T_r *= 2*t/L
```

```
T_i -= Tm2_i
T_r -= Tm2_r

dy.assign(T_r)
Coeff.assign(real(ChebCoeffs[i]))
dy *= Coeff
y += dy

dy.assign(T_i)
Coeff.assign(imag(ChebCoeffs[i]))
dy *= -Coeff
y += dy
```

Parallel averaging

$$U^{n+1} = \exp(L\Delta t)\left(U^n + \sum_{m=-M/2}^{M/2} w_m \exp(-Ls_i)\Delta\Phi\left(\exp(Ls_i)U^n\right)\right).$$

Firedrake now has the Ensemble communicator class for ensembles of functions with spatial domain decomposition.



Imperial College London

```
ensemble = Ensemble(COMM_WORLD, 1)
mesh = IcosahedralSphereMesh(radius=R0, \
    refinement_level=ref_level, degree=3, \
                          comm = ensemble.comm)

...

while t < tmax + 0.5*dt:
    t += dt

    cheby.apply(U, V, expt)

    for i in range(ncycles):
        USlow_in.assign(V)
        SlowSolver.solve()
```

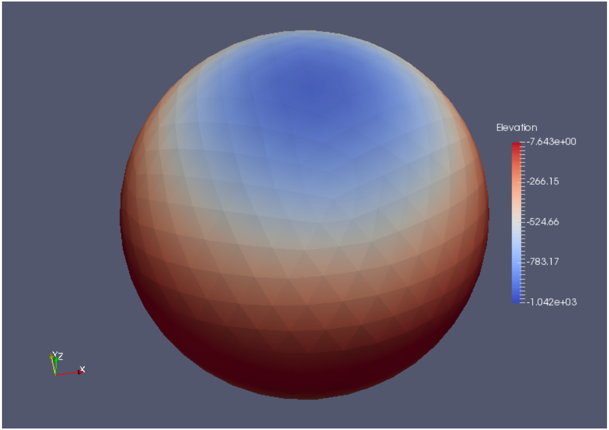
Imperial College London

```
        USlow_in.assign(USlow_out)
        SlowSolver.solve()
        V.assign(0.5*(V + USlow_out))
    V.assign(V-U)

    cheby.apply(V, DU, -expt)
    DU *= wt

    ensemble.allreduce(DU, V)
    U += V

    cheby.apply(V, U, dt)
```

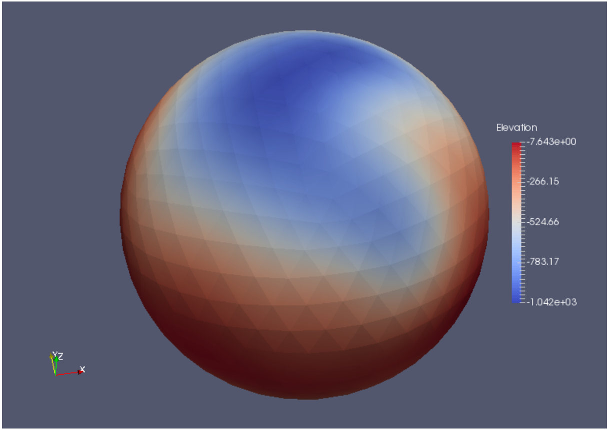
Averaged time integrator

- Icosehedral mesh refinement 3
- BDM2 for velocity, DG1 for height, both upwinded
- $\Delta t = 0.1$ hour, averaging window $= 2.5$ hours
- 150 terms in average (overkill)

$$U^{n+1} = \exp(L\Delta t)\left(U^n + \sum_{m=-M/2}^{M/2} w_m \exp(-Ls_i)\Delta\Phi\left(\exp(Ls_i)U^n\right)\right).$$



1



56

Imperial College London

:(

Unfortunately this scheme is unstable for larger Δt.

Time integrator of average

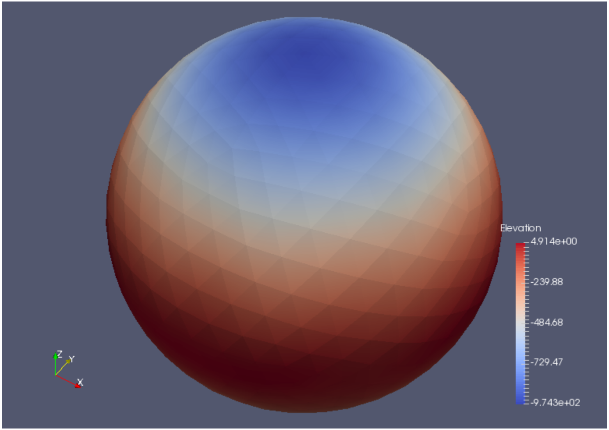
(That's the original Haut-Wingate (2014) method)

- Icosehedral mesh refinement 3
- BDM2 for velocity, DG1 for height, both upwinded
- $\Delta t = 1$ hour, averaging window = 2.5 hours
- 150 terms in average (overkill)

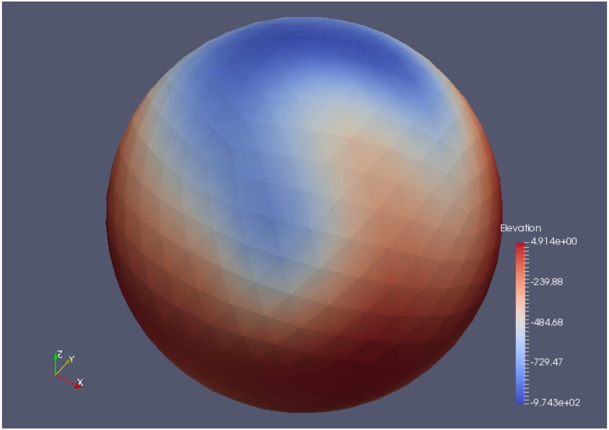
$$V^{n+1/2} = U^n + \frac{\Delta t}{2} \sum_{m=-M/2}^{M/2} w_m \exp(-Ls_i) N\left(\exp(Ls_i) U^n\right),$$

$$V^{n+1} = U^n + \Delta t \sum_{m=-M/2}^{M/2} w_m \exp(-Ls_i) N\left(\exp(Ls_i) V^{n+1/2}\right),$$

$$U^{n+1} = \exp(L\Delta t) V^{n+1}.$$



1



56

What's next?

- Some more rigorous checking of results and higher resolution (these results are from yesterday!)
- Benchmarking of cost of allreduce
- Try to understand the instability in the averaged time-integrator
- Incorporation into predictor-corrector schemes (SDC, Parareal, PFASST)
- Use in data assimilation