ESCAPE 2

Spectral Transform

Andreas Mueller
ESCAPE: Energy-efficient Scalable Algorithms for Weather Prediction at Exascale

Extract model dwarfs...

... explore alternative numerical algorithms...

... hardware adaptation...

... reassemble model
Overview

10 minutes
- Fourier transform
- Spectral transform

60 minutes
- hands-on exercises with Python
- coffee break and group photo in between

30 minutes
- aliasing
- parallelization
- performance
- Fast Legendre Transform
technology applied at ECMWF for the last 30 years

- spectral transform
- semi-Lagrangian
- semi-implicit

pie chart: % of runtime in 9km operational forecast

- spectral transform: 24%
- physics + radiation: 26%
- semi-implicit solver: 9%
- grid point dynamics: 9%
- wave model: 9%
- ocean model: 2%
IFS (Integrated Forecast System)

technology applied at ECMWF for the last 30 years

- spectral transform
- semi-Lagrangian
- semi-implicit

pie chart: % of runtime in 5km forecast (future operational)
IFS (Integrated Forecast System) technology applied at ECMWF for the last 30 years

- spectral transform
- semi-Lagrangian
- semi-implicit

pie chart: % of runtime in 1.25km forecast (experiment, no ocean)

- spectral transform: 41%
- grid point dynamics: 38%
- physics+radiation: 20%
- semi-implicit solver: 2%
- wave model: 2%
- ocean model: 41%
Fourier transform

Fourier transform = Spectral transform in 1D
Fourier transform = Spectral transform in 1D

location $x$  

wavenumber $n$

grid point space  Fourier space
Fourier transform

\[ f(x) = \sum_n f_n \cdot e^{-2\pi i nx} \]
Fourier transform

\[ f(x) = \sum_n f_n \cdot e^{-2\pi i n x} \]

\[ \frac{df(x)}{dx} = \sum_n (-2\pi i n f_n) \cdot e^{-2\pi i n x} \]
on the sphere: spectral transform

grid point space = spectral space

= + + + \ldots

spherical harmonics
on the sphere: spectral transform

\[ f(\phi, \lambda) = \Re \left( \sum_{m=0}^{M} \sum_{n=m}^{M} f_{m,n} Y_{m}^{n}(\phi, \lambda) \right) \]

grid point space = spherical harmonics + spectral space

m: zonal wavenumber
n: total wavenumber
M: truncation
on the sphere: spectral transform

\[ f(\phi, \lambda) = \mathbb{R} \left( \sum_{m=0}^{M} \sum_{n=m}^{M} f_{m,n} Y_{n}^{m}(\phi, \lambda) \right) \]

Grid point variable

Latitude
Longitude

Spectral coefficients
Spherical harmonics

m: zonal wavenumber
n: total wavenumber
M: truncation

Legendre polynomials

\[ f(\phi, \lambda) = \mathbb{R} \left( \sum_{m=0}^{M} e^{im\lambda} \sum_{n=m}^{M} f_{m,n} P_{n}^{m}(\phi) \right) \]

Legendre transform
Fourier transform
time step in IFS

- Grid-point space
  - semi-Lagrangian advection
  - physical parametrizations
  - products of terms

- Spectral space
  - horizontal gradients
  - semi-implicit calculations
  - horizontal diffusion

FFT: Fast Fourier Transform, LT: Legendre Transform

No grid-staggering of prognostic variables
hands-on session

on the classroom computers:
run in the terminal:
/home/ectrain/trx/NM_TC2019/copyspectral.sh

in the cloud (Microsoft):
https://notebooks.azure.com/anmrde/libraries/tnm2019
click on clone

files:
TCNM2019.ipynb: Python notebook with exercises
TCNM2019solution.ipynb: notebook including sample solutions
Issue: multiplication of two variables produces shorter waves than grid can handle
Alias: multiplication of two variables produces shorter waves than grid can handle.
aliasing example
500hPa adiabatic zonal wind tendencies (T159)

with aliasing

filtered
aliasing example
500hPa adiabatic meridional wind tendencies (T159)

with aliasing

filtered
aliasing example
kinetic energy spectra, 100 hPa

Horizontal kinetic energy spectra plots
Equivalent grid/km

with aliasing

filtered
alternatives to using a filter

Idea: use more grid points than spectral coefficients

Orszag, 1971:

2N+1 gridpoints to N waves: linear grid ~ 1-2 Δ
3N+1 gridpoints to N waves: quadratic grid ~ 2-3 Δ
4N+1 gridpoints to N waves: cubic grid ~ 3-4 Δ (Wedi, 2014)

Spatial filter range
effective resolution
of linear and cubic grids (Abdalla et al. 2013)

Figure 7: Locations of the octahedral reduced Gaussian grid nodes (left), and the edges of the primary mesh connecting the nodes as applied with the finite-volume discretisation in FVM (right). The dual mesh consists of general polygons and is not shown. A coarse octahedral grid with only 24 latitudes between pole and equator 'O24' is used for illustration.

Figure 8: Comparison of effective resolution in linear grid and cubic grid truncations compared to spectra derived from altimeter observations ((Abdalla et al. 2013), S. Abdallah personal communication). Effective resolution is increased with the cubic grid truncation but still notably coarser than the 3-4 of the truncation wavenumber-gridpoint ratio.
inverse spectral transform

spectral data: $D(f, i, n, m)$

fastest index left (column-major order like in Fortran)

fields (variables, height levels)

wave numbers
$m = 0, ..., N; \quad n = 0, ..., N-m$

(N: truncation)

real and imaginary part
inverse spectral transform

spectral data: \( D(f, i, n, m) \)

for each \( m \): 

- even \( n \)
- odd \( n \)

\[
S_m(f, i, \phi) = \sum_n D_{e,m}(f, i, n) \cdot P_{e,m}(n, \phi), \quad A_m(f, i, \phi) = \sum_n D_{o,m}(f, i, n) \cdot P_{o,m}(n, \phi)
\]

\( \phi > 0 \):
\[
F(i, m, \phi, f) = S_m(f, i, \phi) + A_m(f, i, \phi)
\]
\( \phi < 0 \):
\[
F(i, m, \phi, f) = S_m(f, i, -\phi) - A_m(f, i, -\phi)
\]

for each \( \phi, f \):
\[
G_{\phi, f}(\lambda) = \text{FFT}(F_{\phi, f}(i, m))
\]

grid point data: \( G(f, \lambda, \phi) \)

\( m=0,...,N; \quad n=0,...,N-m \)

\( P \): precomputed Legendre polynomials

\( D \): matrix multiplications

\( \text{FFT} \): Fast Fourier Transform
inverse spectral transform

spectral data: \( D(f, i, n, m) \)

for each \( m \):
- even \( n \):
  \[
  S_m(f, i, \phi) = \sum_n D_{e,m}(f, i, n) \cdot P_{e,m}(n, \phi),
  \]
  \[
  A_m(f, i, \phi) = \sum_n D_{o,m}(f, i, n) \cdot P_{o,m}(n, \phi)
  \]
  \[
  \phi > 0: \quad F(i, m, \phi, f) = S_m(f, i, \phi) + A_m(f, i, \phi)
  \]
  \[
  \phi < 0: \quad F(i, m, \phi, f) = S_m(f, i, -\phi) - A_m(f, i, -\phi)
  \]

for each \( \phi, f \):
  \[
  G_{\phi, f}(\lambda) = \text{FFT}(F_{\phi, f}(i, m))
  \]

grid point data: \( G(f, \lambda, \phi) \)
direct spectral transform

- same like inverse spectral transform
- reverse order
- multiply data with Gaussian quadrature weights before Legendre transform
performance comparison
of IFS with other models

IFS

13km Case: Speed Normalized to Operational Threshold (8.5 mins per day)

Fraction of Operational Threshold

Number of Edison Cores (CRAY XC-30)

(Fichalakes et al, NGGPS AVEC report, 2015)
scalability comparison of IFS with other models

(Michalakes et al, NGGPS AVEC report, 2015)
IFS scaling on Summit and PizDaint (CPU only)

- TCo7999 (1km) H PizDaint
- TCo7999 (1km) NH PizDaint
- TCo7999 (1km) H Summit

Forecast Days / Day vs Compute nodes (XC50 x12 / Summit x 42 == cores)

- 17.71 days at 480 nodes
- 14.19 days at 960 nodes
- 112.15 days at 5280 nodes
- 64.97 days at 2400 nodes
- 64.37 days at 3360 nodes
- 69.38 days at 5280 nodes
- 33.93 days at 1440 nodes
- 31.10 days at 4800 nodes
spectral transform vs discontinuous Galerkin projected for 5km 2-day forecast

DG, horizontally explicit => 4s time-step, almost no communication

IFS (spectral transform): 240s time-step, lots of communication

communication volume:

DG (like on the left)

time to solution:

34 TB on 2880 MPI procs

427 TB on 2880 MPI procs

689 TB on 57600 MPI procs

4 hours

12 minutes

12 minutes
performances in GFlops/s

peak intensity in Flops/Byte

peak bandwidth

goal peak performance

Spherical Harmonics Dwarf on NVIDIA Tesla P100

0.037s (10.5x)

0.389s (1x)

figure: courtesy of Alan Gray, Peter Messmer (NVIDIA)
optimisations by NVIDIA in ESCAPE

Spherical Harmonics Dwarf TCO639 Test Case
4 GPUs on DGX-1V

- Original MPI: 1.89s
- CUDA-aware MPI: 0.16s
- CUDA IPC + Streams: 0.12s

figure: courtesy of Alan Gray, Peter Messmer (NVIDIA)
optimisations by NVIDIA in ESCAPE

Spherical Harmonics Dwarf TCO639 Test Case
DGX-2 vs DGX-1V

DGX-1V uses MPI for >=8 GPUs (due to lack of AlltoAll links), all others use CUDA IPC. DGX-2 results use pre-production hardware.

figure: courtesy of Alan Gray, Peter Messmer (NVIDIA)
GPUs vs CPUs on Summit

The graph compares the runtime per timestep in seconds for NVIDIA V100 GPUs and IBM Power9 CPUs across different numbers of MPI tasks (which is equal to the number of GPUs). The graph shows that with increasing numbers of MPI tasks, the runtime decreases for both GPUs and CPUs. The blue line represents NVIDIA V100 GPUs, and the green line represents IBM Power9 CPUs. The black dashed line represents perfect scaling. The x-axis represents the number of MPI tasks, and the y-axis represents the runtime per timestep in seconds.
Optalysys: optical processor for spectral transform

Figures used with permission from Optalysys, 2017
Fast Legendre Transform

matrix of Legendre polynomials

truncation N=500, zonal wavenumber m=40

FLT:
step 1: split matrix into two rows
step 2: use interpolation to empty half of the columns
Fast Legendre Transform

matrix of Legendre polynomials

truncation N=500, zonal wavenumber m=40

FLT:
step 1: split matrix into two rows
step 2: use interpolation to empty half of the columns
step 3: reorder columns
Fast Legendre Transform

matrix of Legendre polynomials

truncation $N=500$, zonal wavenumber $m=40$

FLT:

step 1: split matrix into two rows

step 2: use interpolation to empty half of the columns

step 3: reorder columns

step 4: apply to each block recursively
Fast Legendre Transform

matrix of Legendre polynomials

truncation $N=500$, zonal wavenumber $m=40$

FLT:

1. **step 1**: split matrix into two rows
2. **step 2**: use interpolation to empty half of the columns
3. **step 3**: reorder columns
4. **step 4**: apply to each block recursively
Fast Legendre Transform

matrix of Legendre polynomials

truncation $N=500$, zonal wavenumber $m=40$

FLT:

step 1: split matrix into two rows

step 2: use interpolation to empty half of the columns

step 3: reorder columns

step 4: apply to each block recursively
Fast Legendre Transform

matrix of Legendre polynomials

truncation N=500, zonal wavenumber m=100

FLT:
step 1: split matrix into two rows
step 2: use interpolation to empty half of the columns
step 3: reorder columns
step 4: apply to each block recursively
Number of floating point operations for direct or inverse spectral transforms of a single field, scaled by $N^2 \log^3 N$. The bar chart shows the comparison between no FLT and FLT for different values of $N$: 799, 1279, 2047, 3999, and 7999.
Average wall-clock time compute cost of $10^7$ spectral transforms scaled by $N^2 \log^3 N$
• Images on slide 2 used under license from shutterstock.com