DIVA developments and next training: suggestions for harmonization and improvements of results

C. Troupin, A. Barth & J.-M. Beckers (GHER–ULiège)

3rd Product Meeting, Plouzané (France), 15 October, 2019
Thanks for using DIVAnd
Thanks for using DIVAnd

DIVAnd

- build: passing
- coverage: 82%
- codecov: 84%
- docs: latest
- DOI: 10.5281/zenodo.3256600

DIVAnd (Data-Interpolating Variational Analysis in n dimensions) performs an n-dimensional variational analysis/gridding of arbitrarily located observations. Observations will be interpolated/analyzed on a curvilinear grid in 2, 3 or more dimensions. In this sense it is a generalization of the original two-dimensional DIVA version (still available here https://github.com/gher-ulg/DIVA but not further developed anymore).

1. Well tested
2. Always improvable
Why interpolation is complex?
Why interpolation is complex?

100 000’s of data points
Why interpolation is complex?

100,000's of data points

Regions without observations
Why interpolation is complex?

- Regions without observations
- 100,000’s of data points
- Physical boundaries
Developing cutting-edge interpolation products

Back in 2009 at EGU...
Developing cutting-edge interpolation products

Back in 2009 at EGU...

Comparison with WOA

- Generally: same features
- Resolution of coastline
- Artificial mixing

Table: Comparison of computational cost (Rixen et al., 2001).

<table>
<thead>
<tr>
<th></th>
<th>OA</th>
<th>VIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix inversion</td>
<td>$5 \times 10^{15}$</td>
<td>$5 \times 10^{13}$</td>
</tr>
<tr>
<td>Analysis (no inv.)</td>
<td>$9 \times 10^{10}$</td>
<td>$2.5 \times 10^{10}$</td>
</tr>
<tr>
<td>Error (no inv.)</td>
<td>$3 \times 10^{16}$</td>
<td>$2 \times 10^{15}$</td>
</tr>
</tbody>
</table>

$N_d = 300000$, $N_{\phi} = 500 \times 600$, $N_{\ell} = 80000$
Harmonisation
Harmonisation

Ensure consistency... taking into account the regional specificities
3 axes of harmonization

Data sources

Interpolation tool

Procedure & parameters
Data sources

- SeaDataNet
- World Ocean Database
- Coriolis Ocean database for ReAnalysis

Which source(s)?
Which version?
How to eliminate duplicates?
Data sources

1. Which source(s)?
2. Which version?
3. How to eliminate duplicates?

SeaDataNet
World Ocean Database
Coriolis Ocean database for ReAnalysis
Taking care of the domains...
Taking care of the domains...
Julia version:  ≥ 1.0  
DIVAnd version:  ≥ v2.1.0

(now at 1.2)  
(now at v2.4.0)
Domain

Spatial resolution: depending on region
Total time coverage: depending on region
Decade definitions: consistency (merging)
Vertical levels: consistency across regions (merging)
Procedure

Basic parameters

Bathymetry: GEBCO or EMODnet Bathymetry, resolution depending on domain
Correlation length: optimized (if coverage allows) + good
Noise-to-signal ratio: optimized (if coverage allows)
Data weights: optional (check sensitivity on a few levels)
Background field: to discuss (case by case)
Procedure

Other parameters

surfextend = true  
(coeff_derivative2 = [0., 0., 10^{-8}]  
(vertical extension at surface)  
(sensitivity test)
Improvement of DIVAnd
Continuous improvement using user feedback

<table>
<thead>
<tr>
<th>Event</th>
<th>Number of commits</th>
<th>DIVAnd code</th>
<th>Notebooks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st workshop (6 April 2018)</td>
<td>773</td>
<td>265</td>
<td></td>
</tr>
<tr>
<td>2nd training course (26 June 2019)</td>
<td>73</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>
Examples of what we’ve changed

V2.0.2 (Aug 21, 2018): matrix allocation in ODV reading
Examples of what we’ve changed

V2.0.2  (Aug 21, 2018): matrix allocation in ODV reading

V2.1.0  (Sep 3, 2018):
- Support Julia 1.0
- Optimization in interpolation routines
- Improved type-stability
Examples of what we’ve changed

**V2.0.2** (Aug 21, 2018): matrix allocation in ODV reading

**V2.1.0** (Sep 3, 2018):
- Support Julia 1.0
- Optimization in interpolation routines
- Improved type-stability

**V2.1.1** (Oct 19, 2018): allow one to force the direct solver
Examples of what we’ve changed

V2.0.2 (Aug 21, 2018): matrix allocation in ODV reading

V2.1.0 (Sep 3, 2018):
- Support Julia 1.0
- Optimization in interpolation routines
- Improved type-stability

V2.1.1 (Oct 19, 2018): allow one to force the direct solver

V2.2.0 (Dec 20, 2018): NetCDF import for ODV
Examples of what we’ve changed

**V2.3.0** (Jan 24, 2019):

- time varying background
- edit mask with flood-fill
- optimization
Examples of what we’ve changed

V2.3.0 (Jan 24, 2019):
- time varying background
- edit mask with flood-fill
- optimization

V2.3.1 (Feb 11, 2019): fix interpolation issue for time series near masked grid points (coast line)
Examples of what we’ve changed

V2.3.0  (Jan 24, 2019):
  ▶  time varying background
  ▶  edit mask with flood-fill
  ▶  optimization

V2.3.1  (Feb 11, 2019): fix interpolation issue for time series near masked grid points (coast line)

V2.4.0  (Jun 25, 2019):
  ▶  Fixed issue on DIVAnd.fit
  ▶  diva3d correlation length fitting using an empty tuple for len
  ▶  fit_isotropic arguments
Speeding up things

Julia is fast!

Julia was designed from the beginning for **high performance**. Julia programs compile to efficient native code for multiple platforms via LLVM.

Don’t forget:
- Packages are pre-compiled when a kernel is started
- Functions gets compiled during the 1st execution
- Some operations can be executed once and their results stored
Data reading

1. Read the original netCDF ODV file...30"
   
   \[
   \text{obsval, obslon, obslat, obsdepth, obstime, obsid} = \text{NCODV.load(Float64, ODVfile1, "Water body salinity");}
   \]

2. Re-write the data 15"
   
   \[
   \text{DIVAnd.save(obsfile, "Water body salinity", obsval, (obslon, obslat, obsdepth, obstime), obsid)}
   \]

3. Use the newly written files for the climatologies 10"
   
   \[
   \text{DIVAnd.save(obsfile, "Water body salinity", obsval, (obslon, obslat, obsdepth, obstime), obsid)}
   \]
Computing weights "offline"

```julia
using DIVAnd
using JLD

datadir = "data/SeaDataCloud/NorthSea/"
varname = "Salinity"
obsfile = joinpath(datadir, "NorthSea_obs.nc")
netcdfODV = joinpath(datadir, "data_from_SDC_NS_DATA_DISCRETE_TS_V1b.nc")
isfile(netcdfODV)
@info("Reading data from the observation file")
@time obsval, obslon, obslat, obsdepth, obstime, obsid = DIVAnd.loadobs(Float64, obsfile, varname)
@info("Total number of data points: \$(length(obsval))")

@time rdiag = 1.0 / DIVAnd.weight_RtimesOne((obslon, obslat), (0.03, 0.03));
@show maximum(rdiag), mean(rdiag)
save("northsea_weights.jld", "rdiag", rdiag);
```
Export the notebooks as .jl files
Julia is becoming more famous!

TOOLBOX  ·  30 JULY 2019

Julia: come for the syntax, stay for the speed

Researchers often find themselves coding algorithms in one programming language, only to have to rewrite them in a faster one. An up-and-coming language could be the answer.

Jeffrey M. Perkel

https://www.nature.com/articles/d41586-019-02310-3
Thanks for your attention

1. Shall we upload the products to OceanBrowser?
2. Would you agree to publish the notebooks used for the products? (FAIR principles)