Numerical Weather Prediction (NWP)

Raziel & tecer

Contact: tecer@hacknology.de - tecer@jabber.square-wave.de - DECT 4950

Intro

This course is...

...a very brief introduction to NWP
...an even shorter introduction to GIS
...all about running an NWP model, i.e.
...pre-processing the input data
...running the model
...post-processing the output data

This course is *not*...

...complete by almost all means
...about fancy weather visualizations
...providing a production-ready setup
...about data assimilation and other special topics
...too much about maths/meteorology/physics/...

Prelude

NWP model

Input data → pre-processor (data assimilation) → model
→ post-processor → output data

model := simulate processes in the atmosphere / boundary transitions

Evolve the current condition (several key parameters at many levels)
to the next time step using physical rules (ODEs/PDEs)

WRF

- Developed by NCAR @Boulder, CO
- 2 main components:
  -- WPS (preprocessor)
  -- WRF (main integration)
- For post-processing: UPP (or NCL)
- Mostly Fortran
- 300+ pages manual
- Impossible to use for the not-inaugurated
  → NCAR offers tutorials twice a year

Part 1

Preparing the software

Strongly recommended:

Use the VirtualBox image provided at https://www.hacknology.de/vortrag/2016/wrf/
as it contains all the required software and data sets.

Alternatively, set up the software manually, preferably in a Ubuntu 16.04 VM (username "wrf"),
the software and data package is also provided at
https://www.hacknology.de/vortrag/2016/wrf/

Compiling NetCDF

WRF uses NetCDF as the main data format, hence the NetCDF libraries and the Fortran bindings are required. Unfortunately, the distribution packages usually do *not* work, so NetCDF must be compiled manually before compiling WRF.

```
> cd /home/wrf/src
> tar -xf ../packages/netcdf-4.4.1.1.tar.gz ; cd netcdf-4.4.1.1
> ./configure --prefix=/home/wrf/netcdf --disable-dap \
--disable-netcdf-4 --disable-shared
> make && make install
```

Compiling NetCDF Fortran bindings

Most of WRF is Fortran code, so the NetCDF Fortran bindings are required as well.

```
> cd /home/wrf/src
> tar -xf ../packages/netcdf-fortran-4.4.4.tar.gz ; cd netcdf-fortran-4.4.4
> CPPFLAGS=-I/home/wrf/netcdf/include ./configure --prefix=/home/wrf/netcdf \
--disable-shared
> make && make install
```

Now, NetCDF and the NetCDF Fortran bindings are installed at `/home/wrf/netcdf`.

Compiling WRF

The compilation of WRF takes some time.

```
> cd /home/wrf/src
```

```
> tar -xf ../packages/wrf-3.8.1.tar.bz2 ; cd WRFV3
```

```
> export NETCDF=/home/wrf/netcdf
```

```
> export WRFIO_NCD_LARGE_FILE_SUPPORT=1
```

```
> ./configure
```

In the following dialogue, select GNU / dmpar (7) (requires OpenMPI, alternatively, you can select GNU / serial, but then the only 1 core will be used for the actual computation)

`> ./compile -j1 em_real >& compile.log` (for some reason, the OpenMPI version does not like to be compiled in parallel, hence the `-j1` option)

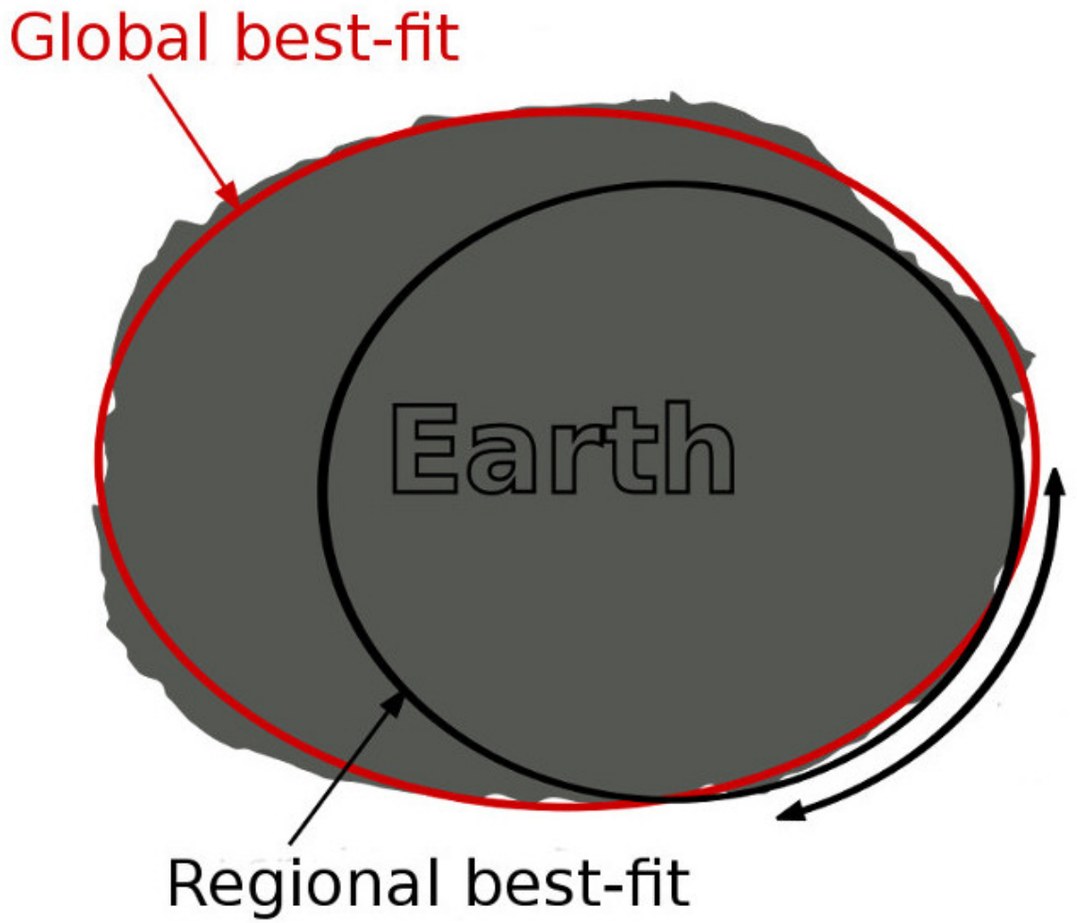
Sit back and enjoy the interlude...

Interlude 1

Map projections

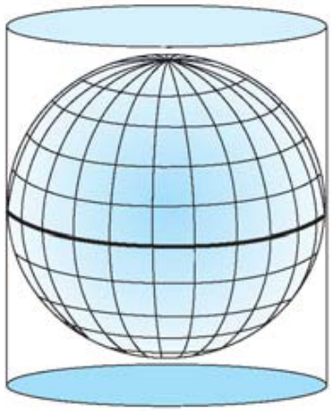
https://www.hacknology.de/vortrag/2016/wrf/

World model



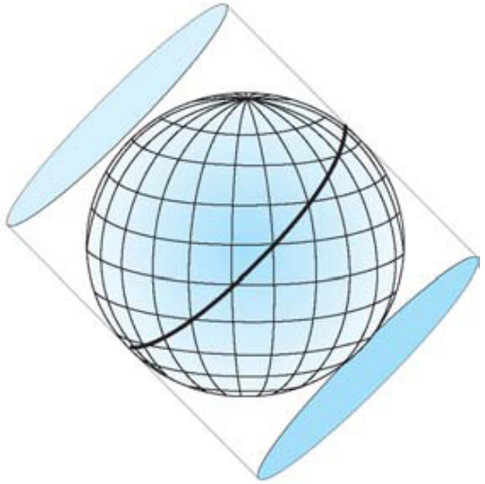
https://www.hacknology.de/vortrag/2016/wrf/

Cylindrical



Normal

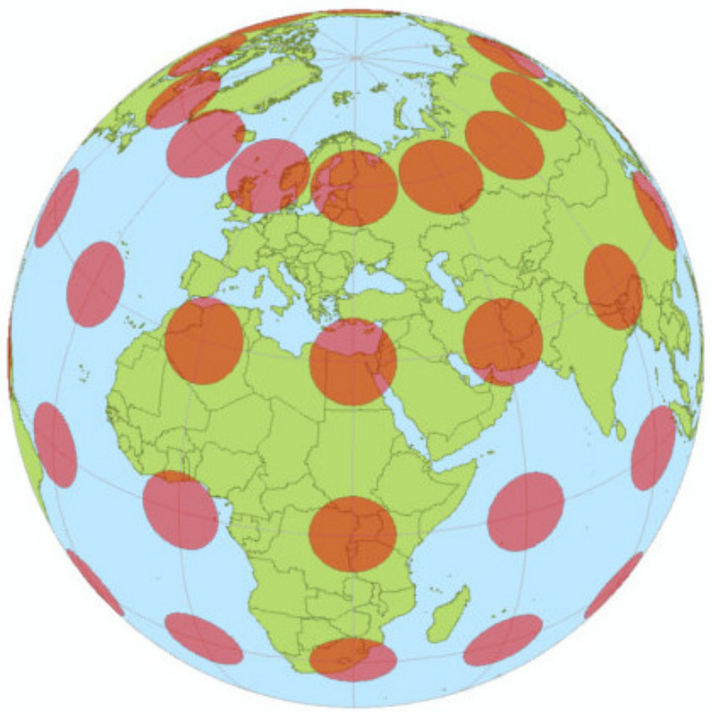
Transverse

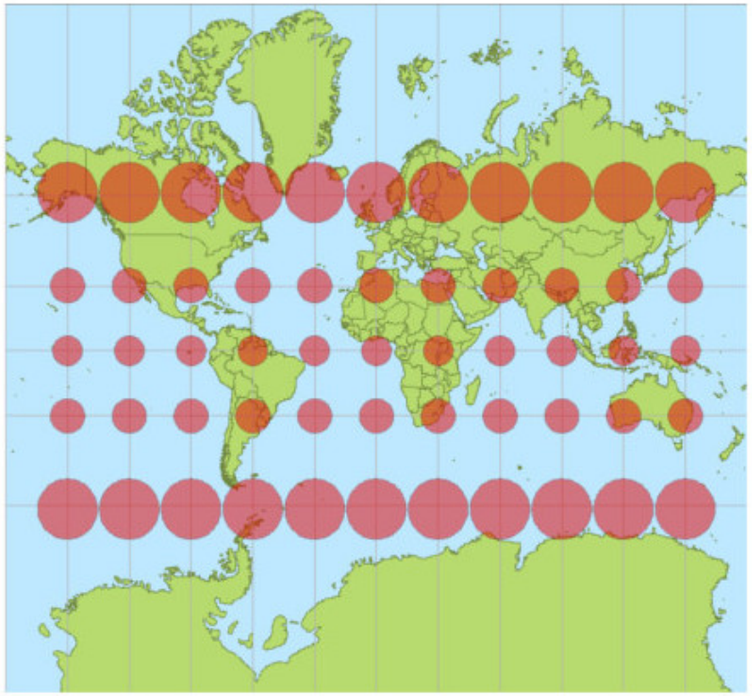


Oblique

https://www.hacknology.de/vortrag/2016/wrf/

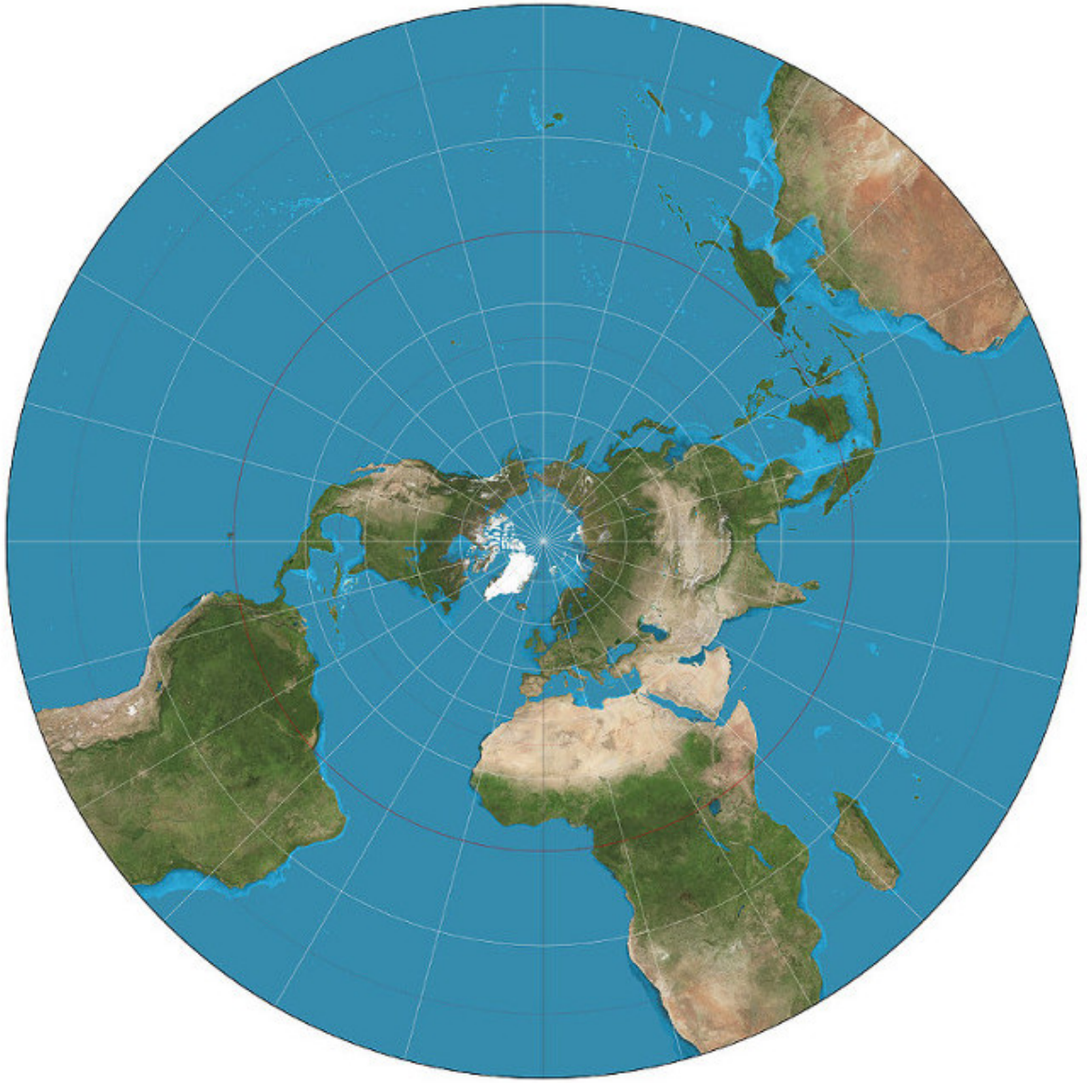
Mercator





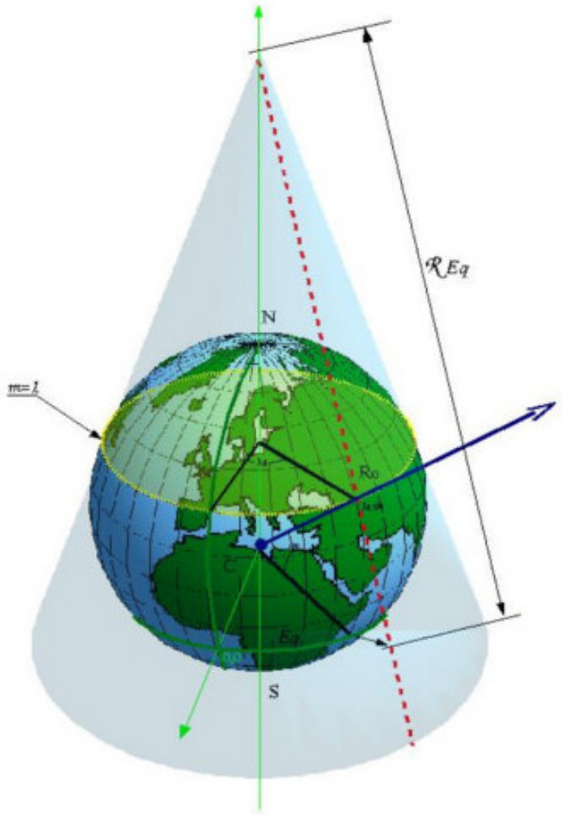
https://www.hacknology.de/vortrag/2016/wrf/

Polar stereographic



https://www.hacknology.de/vortrag/2016/wrf/

Lambert



https://www.hacknology.de/vortrag/2016/wrf/

Map projection recommendations

- Lambert: mid latitudes
- Mercator: low latitudes
- Polar-stereographic: high latitudes
- Lon-Lat: global
- Generally: minimize the distortion!
- Conformity: Locally, angles are preserved
- Areas are *not* preserved

https://www.hacknology.de/vortrag/2016/wrf/

Back to the preparation...

https://www.hacknology.de/vortrag/2016/wrf/

Compiling WPS

WPS is the WRF preprocessing system.

```
> cd /home/wrf/src
```

```
> tar -xf ../packages/wps-3.8.1.tar.bz2 ; cd WPS
```

```
> export NETCDF=/home/wrf/netcdf
```

```
> ./configure
```

In the dialogue, select Linux / gfortran / serial (OpenMPI does not improve anything here).

```
> ./compile >& compile.log
```

Compiling UPP

UPP is the only actually working and usable post-processing system.

```
> cd /home/wrf/src
```

```
> tar -xf ../packages/DTC_upp_v3.1.tar.gz ; cd UPPV3.1
```

```
> ./configure
```

In the dialogue, select Linux / gfortran / serial (dmpar won't work - and doesn't improve much anyway!)

```
> ./compile >& compile.log
```

Now, the software is set up. Next...

Part 2

Preprocessing

WPS

WPS is the pre-processing package for WRF.

Main settings:

- timespan
- area of interest (map projection + extents)

It prepares

- static data
- initial data

by extracting the relevant parts and interpolating the data to the area of interest (horizontally).

WPS configuration

Configuration file: `WPS/namelist.wps`

Important settings:

`start_date` and `end_date`: forecast time span

`interval_seconds`: output time step duration

`e_sn`, `e_we`: number of grid cells south-north/west-east

`geog_data_res`: resolution of static data

`dx`, `dy`: grid cell dimension in map units (f.e. meters)

`map_proj`: map projection ("lambert", "mercator", "polar", "lat-lon")

Projection parameters:

`ref_lat`, `ref_lon`: grid center

`true_lat{1,2}`, `stand_lon`: Lambert-parameters

`geog_data_path`: absolute path to the static data (`/home/wrf/data/WPS_GEOG`)

Pre-processing static data

Static data := Digital elevation model (DEM), land-use, land-sea-mask, (SST),...

```
> cd /home/wrf/src/WPS
> ./geogrid.exe
```

Result: `geo_em` file(s) containing static data horizontally interpolated to AOI.

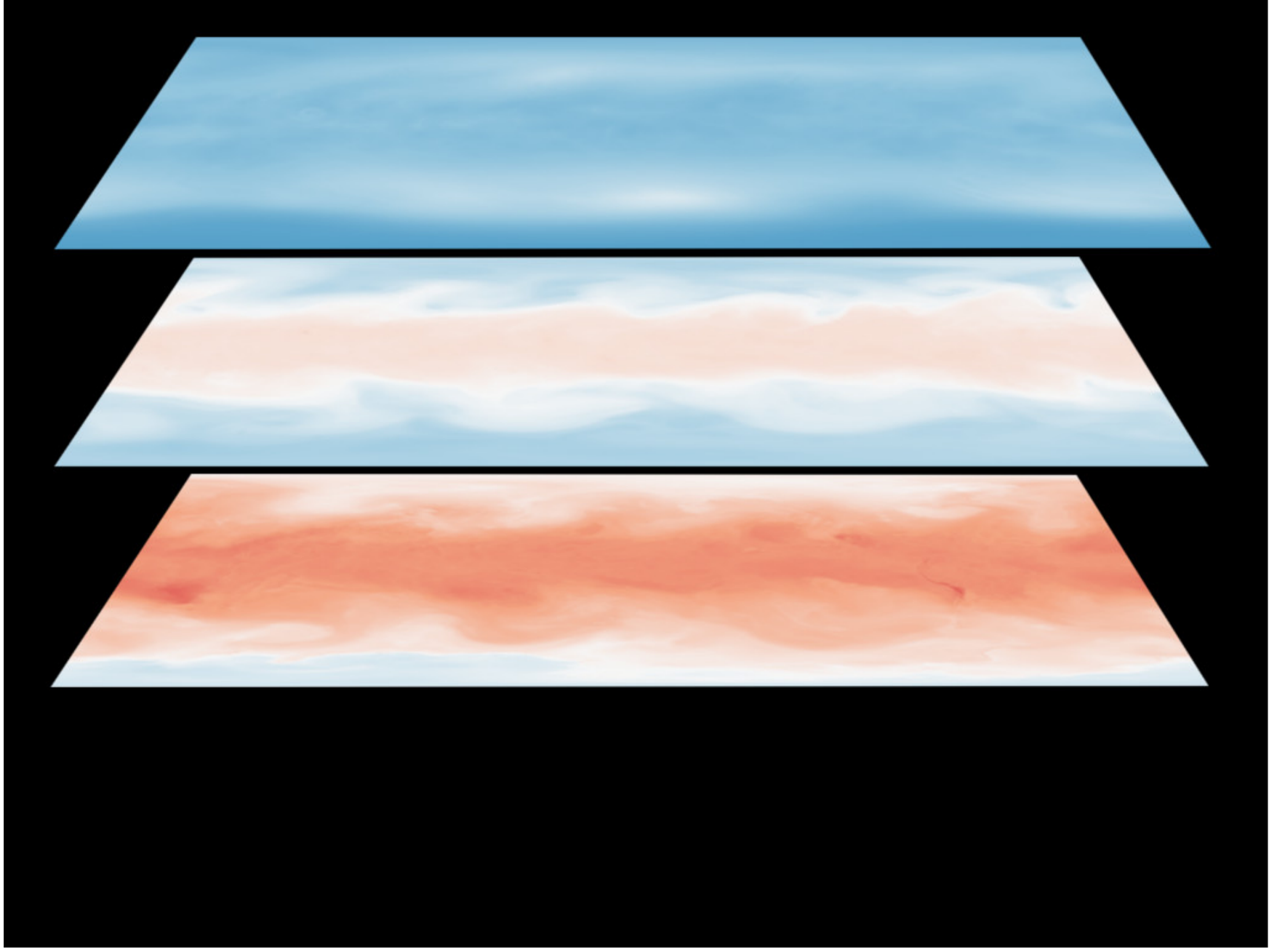
Initial data / BC

Several sources, here: GFS (NOAA/NCEP global model), 4 runs per day.
Temporal resolution: hourly forecast up to 120 hours, 3-hourly 120-240 hours, 12-hourly to 384 hours
Spatial resolution: 0.5 degrees or 0.25 degrees (lon-lat)

Download (freely available) from

```
http://www.ftp.ncep.noaa.gov/data/nccf/com/gfs/prod/
```

File format: Grib2, one file per time step, containing several hundred bands
(=many parameters at many levels)

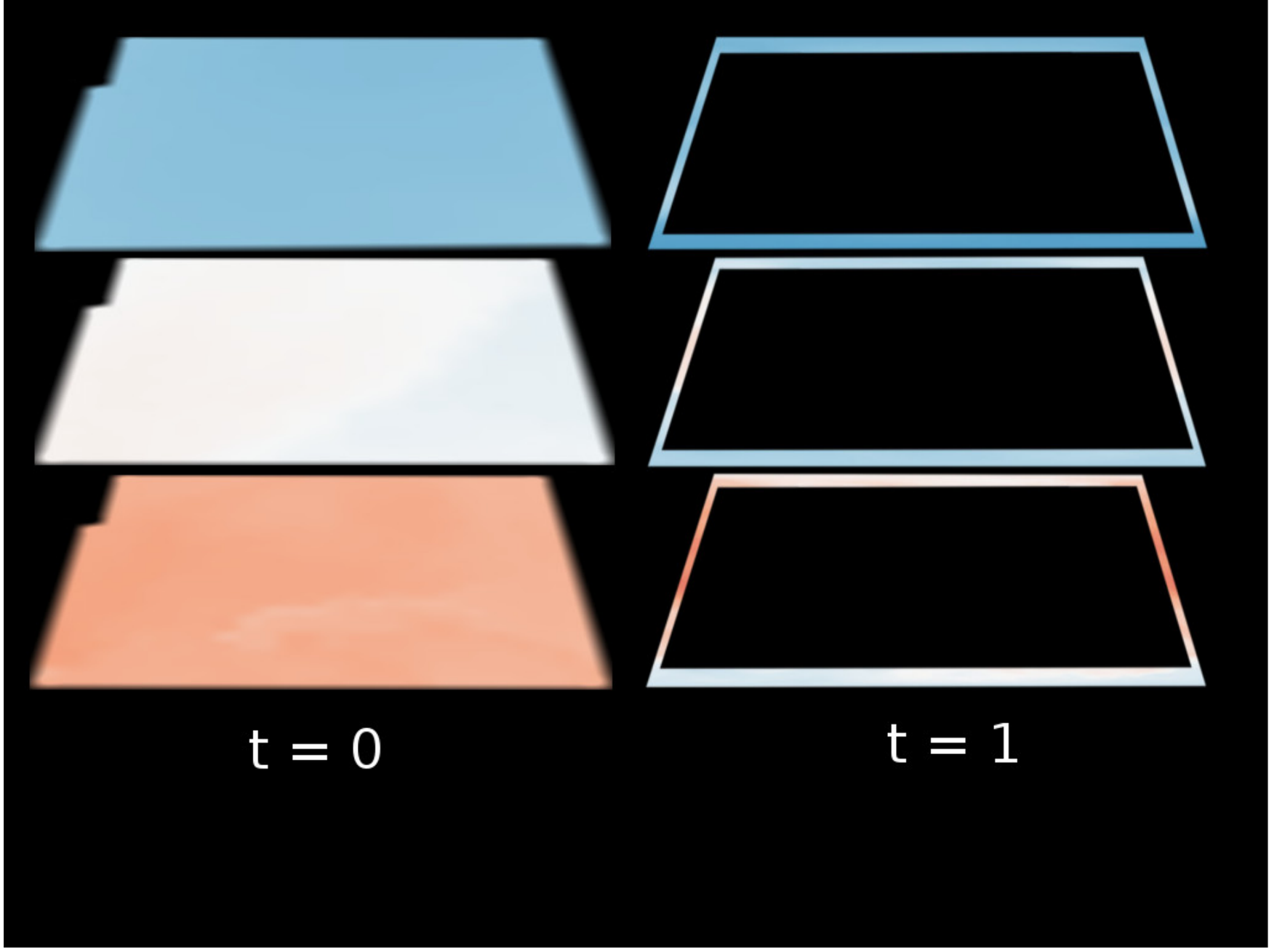


https://www.hacknology.de/vortrag/2016/wrf/

Pre-processing initial/BC data

```
> cd /home/wrf/src/WPS
> ./link_grib.csh /home/wrf/data/gfs.2016121700/gfs.*.gb2
> ln -sf ungrib/Variable_Tables/Vtable.GFS Vtable
> ./ungrib.exe
> ./metgrid.exe
```

Result: `met_em` file(s) containing all the static and initial condition/boundary condition data for all time steps, horizontally interpolated to the AOI.



https://www.hacknology.de/vortrag/2016/wrf/

Part 3

Model run (finally!)

WRF

Main settings:

- timespan
- area of interest
- physics options

Initial condition is evolved using physical rules while respecting the boundary conditions.

https://www.hacknology.de/vortrag/2016/wrf/

WRF configuration

Configuration file: `WRFV3/test/em_real/namelist.input`

Important settings:

Start and end dates, time interval and domain definition as in WPS configuration

`history_interval` : output data every N **minutes**

Settings for physics schemes

Vertical interpolation

```
> cd /home/wrf/src/WRFV3/test/em_real
> ln -sf ../../../WPS/met_em.* .
> ./real.exe
```

Result: `wrfinput` and `wrfbdy` files.

Strictly speaking: another pre-processing step.

WRF Model run

```
> ./wrf.exe
```

if WRF was compiled with OpenMPI support (strongly recommended), it can use multiple cores via

```
> mpirun -n 2 wrf.exe
```

Sit back and have a drink...

Interlude 2

https://www.hacknology.de/vortrag/2016/wrf/

Equations and how to solve them

Derivatives? Differences? PDEs? ODEs? Why large matrices?

ODE example

$\ddot{x} = 2, t \in [0, 1]$
$x(0) = x(1) = 3$

Approximate the 2nd derivative by
$\ddot{x}(t) \approx \frac{x(t-h)-2x(t)+x(t+h)}{h^2}$

("Finite difference method")

https://www.hacknology.de/vortrag/2016/wrf/

Numerical approach

$\frac{1}{h^2}(x_{i-1} - 2x_i + x_{i+1}) = 2, i = 1, ..., N$
for the N inner grid points $x_i := x(t + ih), h := \frac{1}{N+1}$.

Solving the equation at given points in time

We solve this equation f.e. for $N = 3$ time steps, thus

$i = 1 : \frac{1}{h}(2x_1 + x_2) = 2 - \frac{1}{h}x_0$

$i = 2 : \frac{1}{h}(x_1 - 2x_2 + x_3) = 2$

$i = 3 : \frac{1}{h}(x_2 - 2x_3) = 2 - \frac{1}{h}x_4$

Linear equations $\rightarrow Ax = b$ with a 3x3 matrix A and $x = (x_1, x_2, x_3)$.

Of course, we are considering large N (plus more complicated and higher dimensional equations), hence *large* systems of linear equations / (typically sparse) matrices.

Post-processing

UPP to the rescue!

https://www.hacknology.de/vortrag/2016/wrf/

Running UPP

UPP is the unified Post Processing system, to my knowledge the only feasible way of producing usable output.

```
> cd /home/wrf/src/WRFV3/test/em_real/postprd
> ln -s ../wrfout_d01_* .
> cd /home/wrf/src/UPPV3.1/scripts
> ./run_unipost_xmas
```

Transforming the output

The final output files are called `WRFPRS_d01.00` .. `WRFPRS_d01.180`, they are in the *Grib1* format.

To see what is inside

```
> gdalinfo WRFPRS_d01.00 | less
```

You'll see there are many parameters in many different levels (>300 bands).

To transform the grib files into something more useful

```
> gdal_translate -b 294 WRFPRS_d01.00 snowc.000.tiff
```

(This will take band number 294, which is the snow cover, and convert the file into a GeoTIFF, which is much easier to handle than a grib file.

Finally, reproject the data from the original (Lambert-)projection to a Lon-Lat grid:

```
> gdalwarp -t_srs epsg:4326 -dstnodata "-1" snowc.000.tiff snowc.4326.000.tiff
```

(you could also use `epsg:3857` for webmercator as used in most web map applications, or any other projection).

Visualisation

Start QGis.

Add vector layer: `Ctrl+Shift+V`, select `~/data/ne/ne_10m_admin_0_countries.shp`

Add raster layer: `Ctrl+Shift+R`, select `snowc.4326.tiff`

Congratulations!

You've completed a WRF model run, forecasting the weather from December 17, 2016 for 180 hours, and you visualized some of the output data!

Hope you enjoyed the tutorial!

Tasks

Some suggestions:

- change pre-processing for initial and boundary conditions to 1-hourly
- change the WRF output to 1-hourly
- change the spatial extent / resolution
- fetch new GFS data to change the temporal domain