Modularizing TM5 4DVAR

A pythonic approach

Sourish Basu

SRON

Netherlands Institute for Space Research

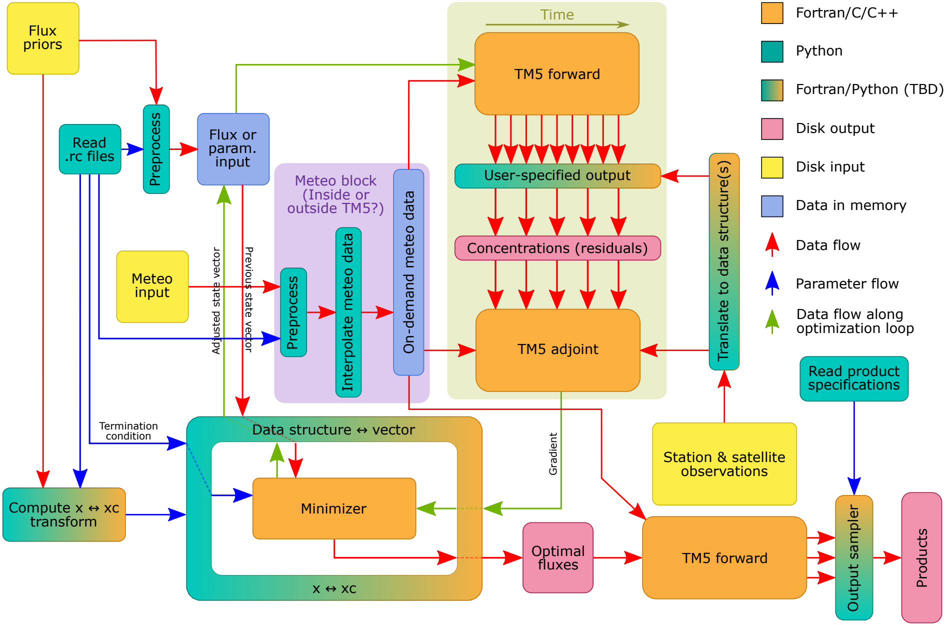
21 June 2010, Heraklion, Crete

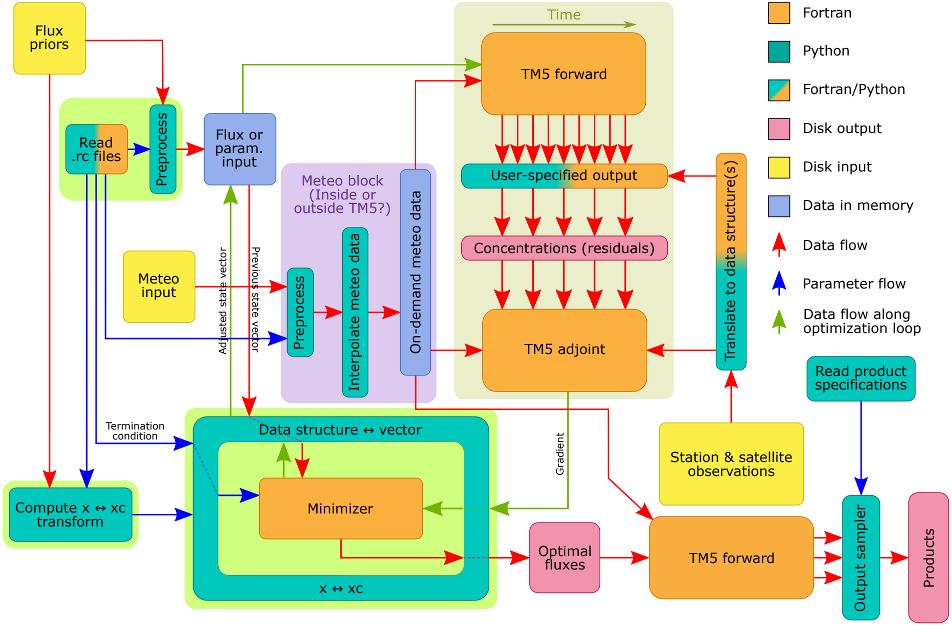
- Parts of 4D-VAR were hard-coded for specific purposes
- Structure is complicated to read for a new user
- Cumbersome to change something (optimization algorithm, meteo format, etc.)
- Difficult to develop code together by working on different parts separately

- More modular structure:
  - Forward model, adjoint model *specific to atmospheric dynamics*
  - Process observations, sample output *specific to instrument*
  - Process meteo *specific to meteo source*
  - Optimization algorithm *specific to application (4DVAR)*

- Computationally intensive parts : Fortran

- Organizational parts (the glue) : Python


Flux priors
Read .rc files
Preprocess
Flux or param. input
Meteo input
Adjusted state vector
Previous state vector
Meteo block (Inside or outside TM5?)
Preprocess
Interpolate meteo data
On-demand meteo data
Time
TM5 forward
User-specified output
Concentrations (residuals)
TM5 adjoint
Translate to data structure(s)
Termination condition
Data structure ↔ vector
Minimizer
x ↔ xc
Compute x ↔ xc transform
Gradient
Optimal fluxes
TM5 forward
Station & satellite observations
Read product specifications
Output sampler
Products
Fortran/C/C++
Python
Fortran/Python (TBD)
Disk output
Disk input
Data in memory
Data flow
Parameter flow
Data flow along optimization loop



Flux priors

Read .rc files

Preprocess

Flux or param. input

Meteo input

Adjusted state vector

Previous state vector

Meteo block (Inside or outside TM5?)

Preprocess

Interpolate meteo data

On-demand meteo data

Time

TM5 forward

User-specified output

Concentrations (residuals)

TM5 adjoint

Translate to data structure(s)

Termination condition

Compute x ↔ xc transform

Data structure ↔ vector

Minimizer

x ↔ xc

Gradient

Optimal fluxes

TM5 forward

Station & satellite observations

Read product specifications

Output sampler

Products

- 🟧 Fortran
- 🟩 Python
- ◩ Fortran/Python
- 🟥 Disk output
- 🟨 Disk input
- 🟦 Data in memory
- ↑ Data flow
- ↑ Parameter flow
- ↑ Data flow along optimization loop

The class hierarchy

```
class ParseConfig                                          # to parse and modify config files
    def SetStartTime()
⋮
class Emissions                                 # general routines for writing emission files
    def WriteEmissions()
    class CO_Emissions                                             # tracer-specific routines
        def Anthropogenic()
    class CH4_Emissions
⋮
class Observations                                       # create OBS files from observations
    def WriteOBSFiles()
    class Aircraft
    class Satellite                                            # instrument-specific routines
        def ParseGOSAT()
⋮
class Precon                                               # pre-conditioner for optimization
    def ReadHorizontalCorrelation()
⋮
class RunTM5                                   # routines for running forward, backward, etc.
    def GradientTest()
```

CO emissions: Fortran

```
select case (emission_dims)
    case (SURFACE)
        call readhdf(‘emissions.hdf’, emis(region)%surf, region)
! not exact F90 code, just an example
! ‘emissions.hdf’ contains monthly emissions in ‘n_region’ arrays:
!     categories × months × latitude divisions × longitude divisions
! kg/sec/gridbox at model resolution
```

CO emissions: Python

```
class Emissions                                   # general routines for processing emissions
    def LoopThroughMonths()
    def AssembleEmissions()
    def WriteEmissions()                                             # writes ‘emissions.hdf’
    class CO_Emissions                                                 # CO-specific routines
        def AssembleEmissions()                # replaces AssembleEmissions() of parent class
        def Anthropogenic()
        def Natural()
        def BiomassBurning()
```

A forward run

```
[1]: %run ParseInit.py
[2]: r = RunTM5((2004,2,27,0),(2004,3,3,0))
[3]: r.SetupEmissions(‘CO’, emisFile=‘input/emissions_CO.hdf’)
[4]: r.RunForward(‘tm5.rc’)
```

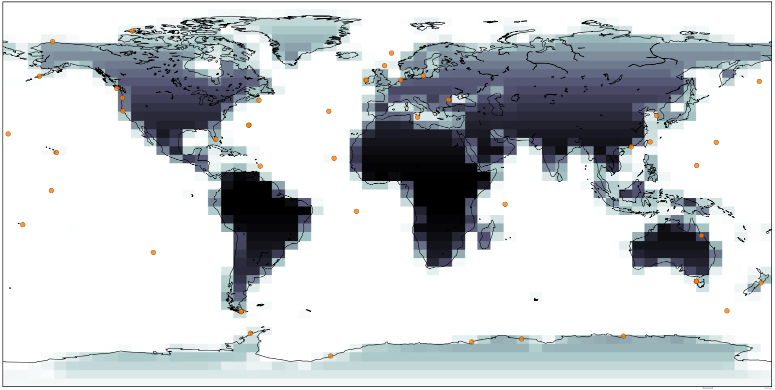
A forward run

```
[1]: %run ParseInit.py
[2]: r = RunTM5((2004,2,27,0),(2004,3,3,0))
[3]: r.SetupEmissions(‘CO’, emisFile=‘input/emissions_CO.hdf’)
[4]: r.RunForward(‘tm5.rc’)
```

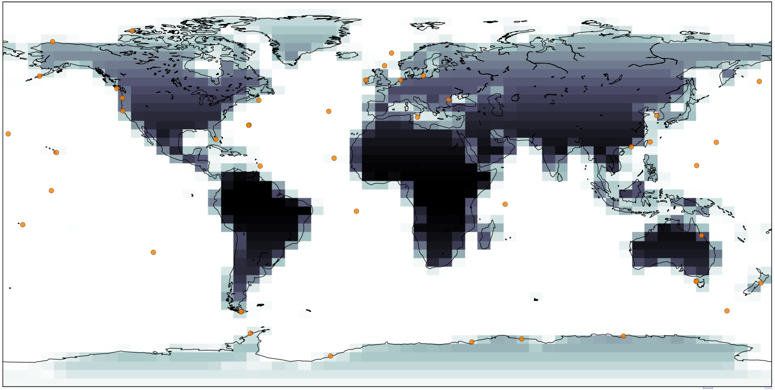
A 4D-VAR run

```
r = RunTM5((2004,2,27,0),(2004,3,3,0))
r.CleanUp()
r.SetupEmissions(‘CO’, emisFile=‘input/emissions_CO.hdf’)
r.RunForward(‘tm5.rc’)
r.StoreModelReprErrors()
r.preco = Precon(r.StartTime, r.EndTime)
for i in range(r.max_iter):
    r.RunForward(‘tm5.rc’)
    r.RestoreModelReprErrors()
    r.RunBackward(‘tm5.rc’)
    r.OptimizerLoop()
```

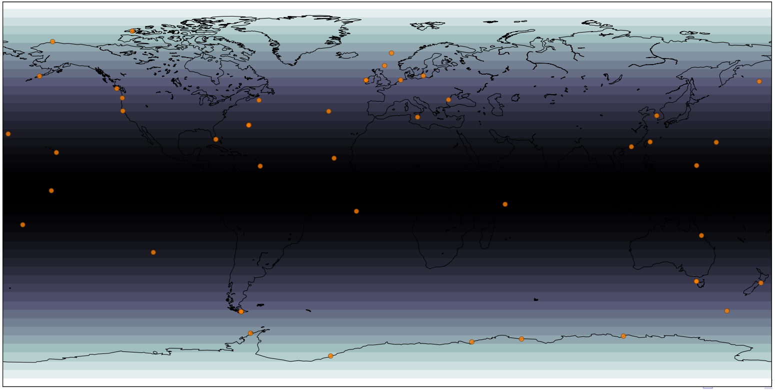
- Emission in a grid-box $\propto$ land area
- "Measure" at 47 NOAA stations
- 27-2-2004 to 3-3-2004 (5 days)
- Prior emission in a grid-box $\propto$ surface area
- Prior total = "True" total
- Emissions close to stations constrained



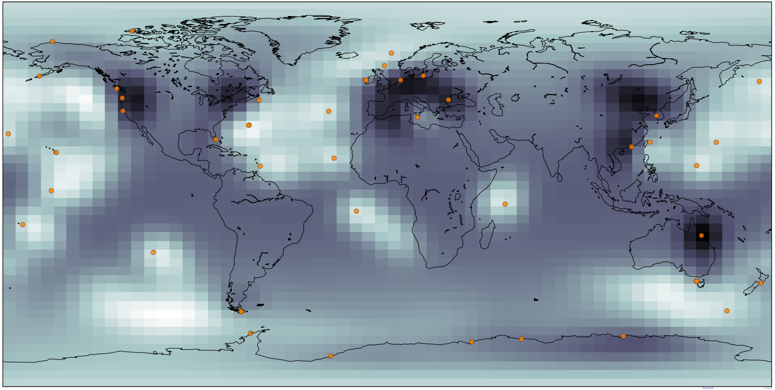
- Emission in a grid-box ∝ land area
- “Measure” at 47 NOAA stations
- 27-2-2004 to 3-3-2004 (5 days)
- Prior emission in a grid-box ∝ surface area
- Prior total = “True” total
- Emissions close to stations constrained

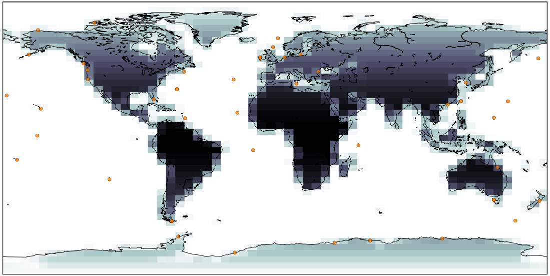


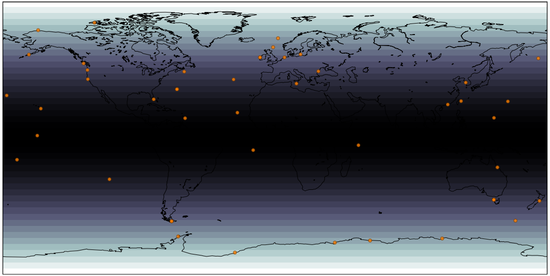
- Emission in a grid-box $\propto$ land area
- “Measure” at 47 NOAA stations
- 27-2-2004 to 3-3-2004 (5 days)
- Prior emission in a grid-box $\propto$ surface area
- Prior total = “True” total
- Emissions close to stations constrained

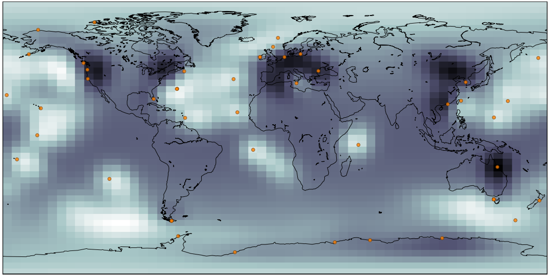


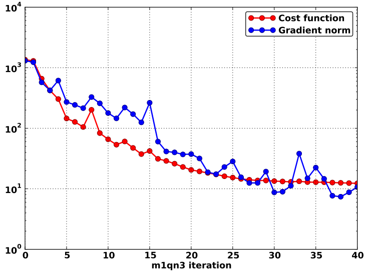
- Emission in a grid-box ∝ land area
- “Measure” at 47 NOAA stations
- 27-2-2004 to 3-3-2004 (5 days)
- Prior emission in a grid-box ∝ surface area
- Prior total = “True” total
- Emissions close to stations constrained





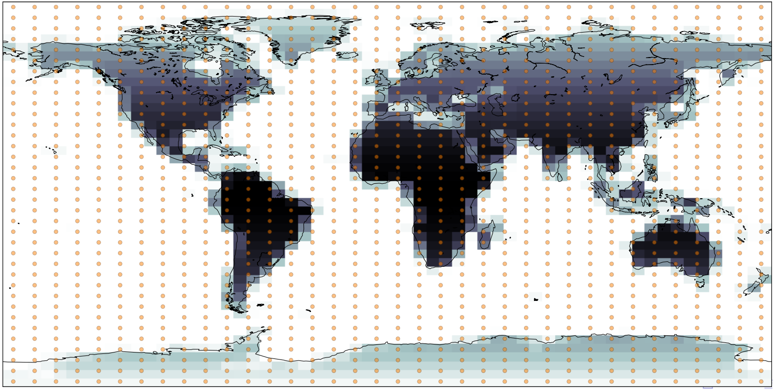




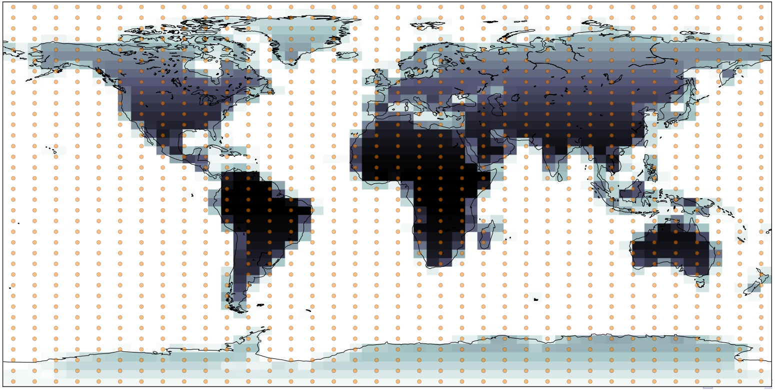


Let’s put stations everywhere!

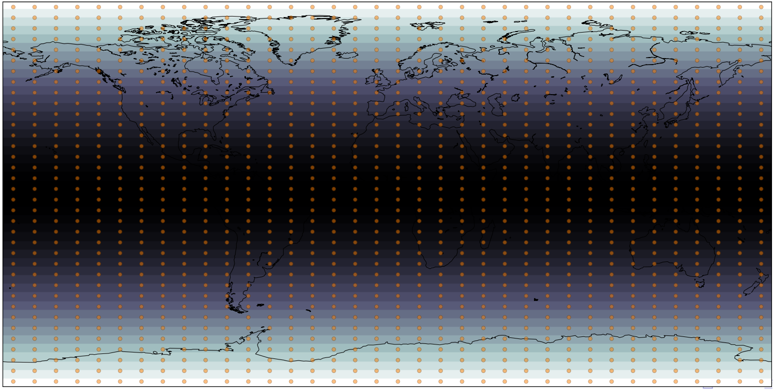
- Emission in a grid-box ∝ land area
- “Measure” at 1296 stations
- 27-2-2004 to 3-3-2004 (5 days)
- Prior emission in a grid-box ∝ surface area
- Prior total = “True” total
- Enough observations to constrain emissions



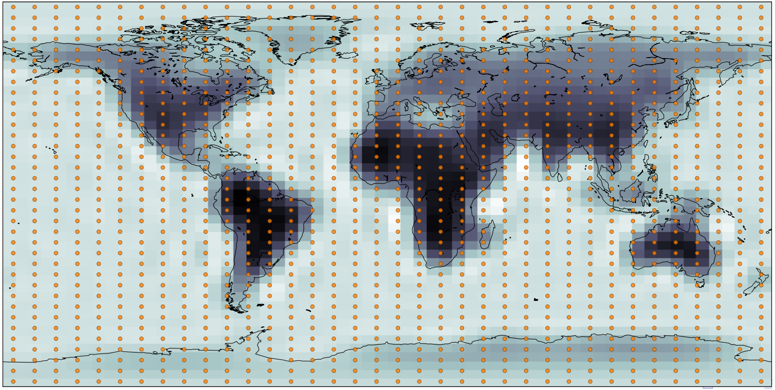
- Emission in a grid-box ∝ land area
- “Measure” at 1296 stations
- 27-2-2004 to 3-3-2004 (5 days)
- Prior emission in a grid-box ∝ surface area
- Prior total = “True” total
- Enough observations to constrain emissions

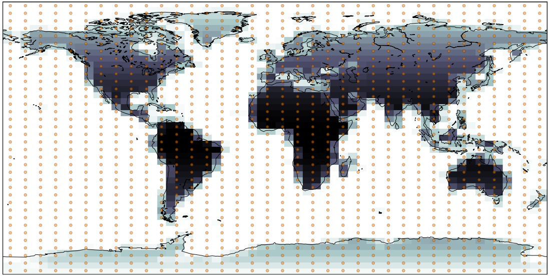


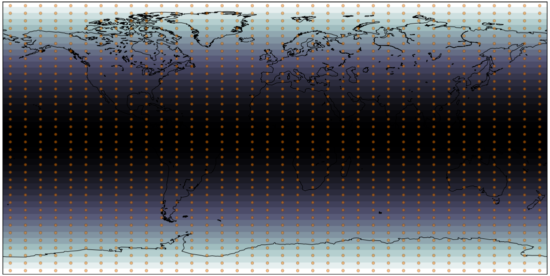
- Emission in a grid-box ∝ land area
- "Measure" at 1296 stations
- 27-2-2004 to 3-3-2004 (5 days)
- Prior emission in a grid-box ∝ surface area
- Prior total = "True" total
- Enough observations to constrain emissions

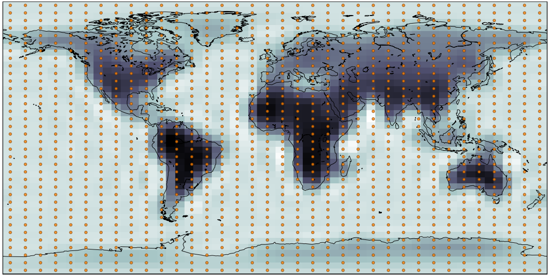


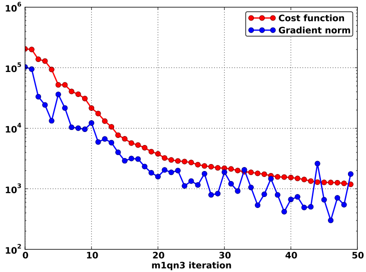
- Emission in a grid-box ∝ land area
- "Measure" at 1296 stations
- 27-2-2004 to 3-3-2004 (5 days)
- Prior emission in a grid-box ∝ surface area
- Prior total = "True" total
- Enough observations to constrain emissions





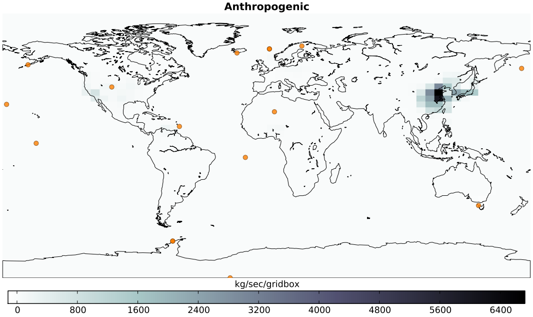


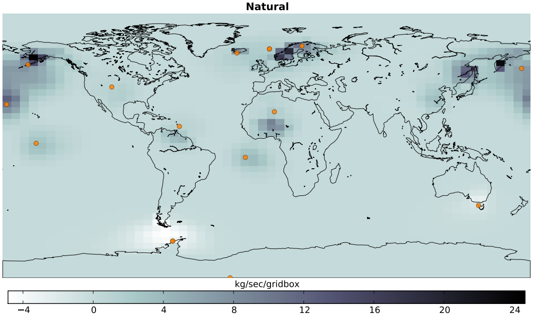


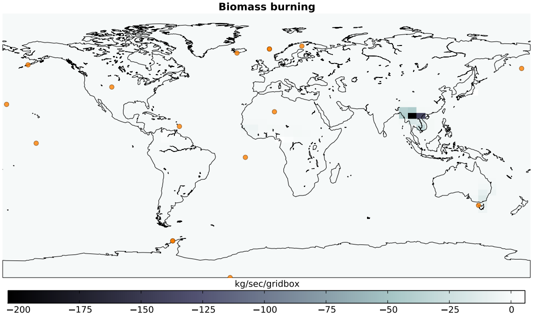


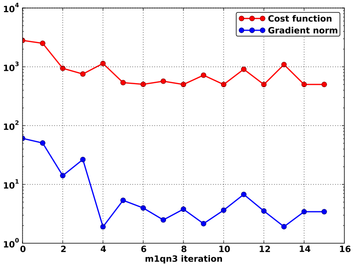
Proof of concept, not a realistic inversion

- Pim Hooghiemstra's inversion using NOAA station data
- 27-2-2004 to 3-3-2004 (5 days)
- Prior emissions from GFED, EDGAR
- Total of 16 observations









Optimized emission – prior

- Add vertical distribution of emissions (e.g., CO)
- Separate the meteo processing, settle on input meteo format for TM5
- Complete parser for config files
- Move observations outside TM5
  - Add satellite module (GOSAT)
  - Interpolation scheme for stations
- Add initial field and arbitrary parameters to preconditioner
- Settle on I/O format for optimizer
- Implement zoom regions