WAGENINGEN
UNIVERSITY & RESEARCH

TM5-MP, CTDAS, and EnKF at the WUR

Joram Hooghiem

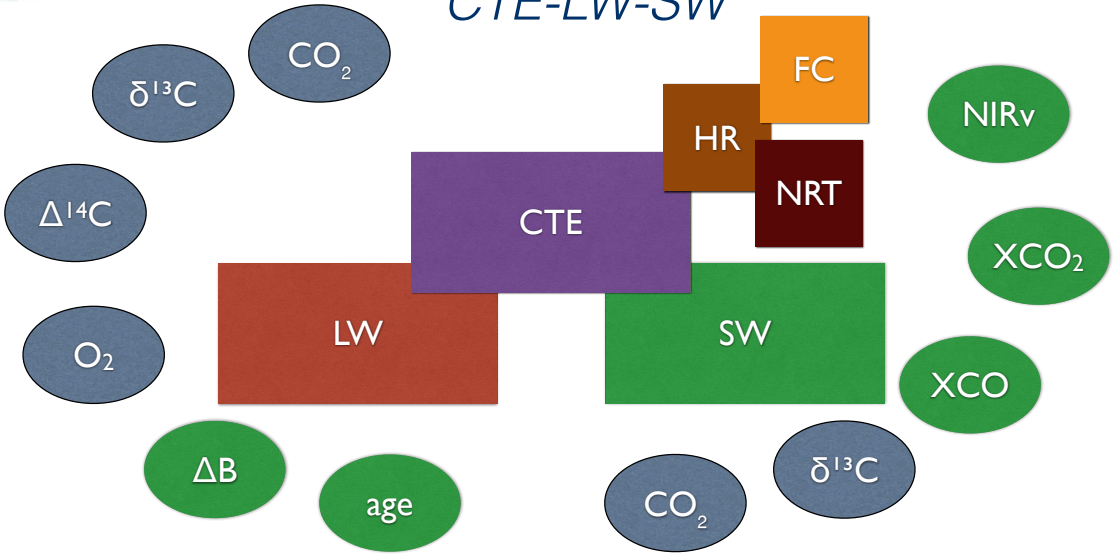
September 3, 2026

%

Co-authors

Carbon-Group (WUR)
CORSO (ECMWF)
13-Carbon project (NOAA)

CTE-LW-SW


CO2
δ13C
FC
HR
NRT
NIRv
Δ14C
CTE
XCO2
LW
SW
O2
XCO
ΔB
age
CO2
δ13C

WAGENINGEN UNIVERSITY & RESEARCH

An update on CTDAS and TM5-MP activities at the WUR

TM5 and CTDAS

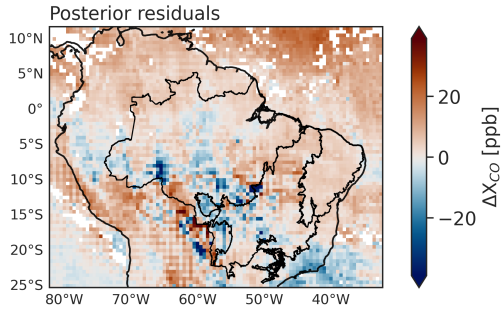
- Estimate global and regional carbon fluxes
- Land ocean sink partitioning
- Development of LW-SW and Multi-Tracer:
  1. GCB runs on 1x1
  2. CO to estimate fire emissions
  3. $\Delta(^{14}C)$ and $\delta(O_2/N_2)$ to estimate fossil fuel emissions
  4. Multi-tracer simulations
  5. H_2 see talk by Firmin

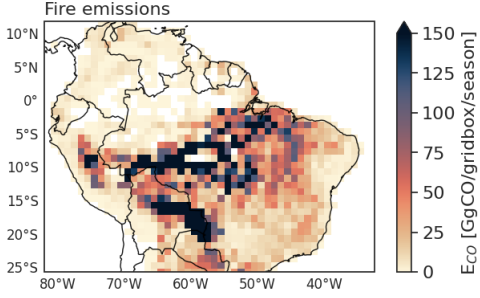
Report some TM5-CTDAS-Developments

Estimations of fire emissions in 2019

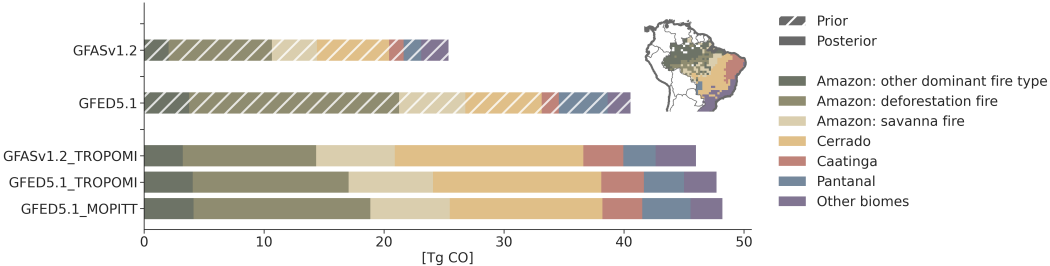
See preprint by Anne-Wil van den Berg et al.

X_{CO} satellite $\rightarrow$ estimation of fire emissions





See preprint by Anne-Wil van den Berg et al.



1. Larger emissions then a-priori estimated
2. Cerrado vs Amazon deforestation

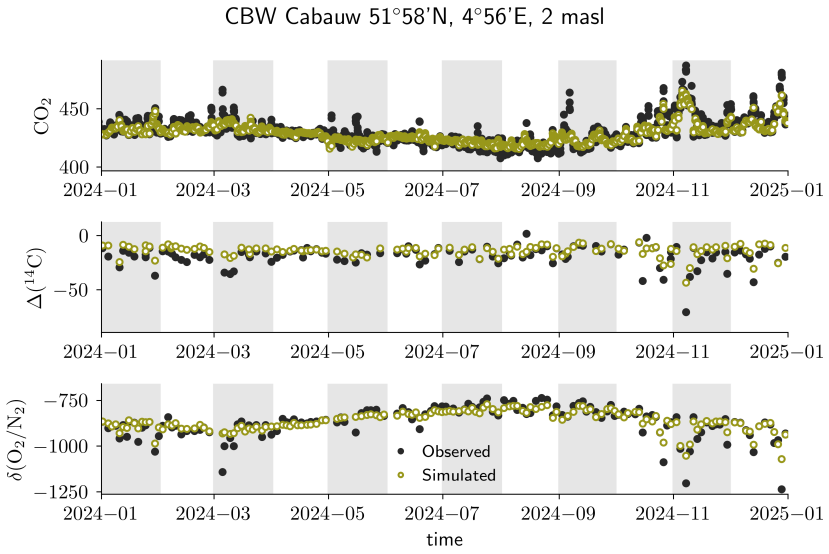
CORSO

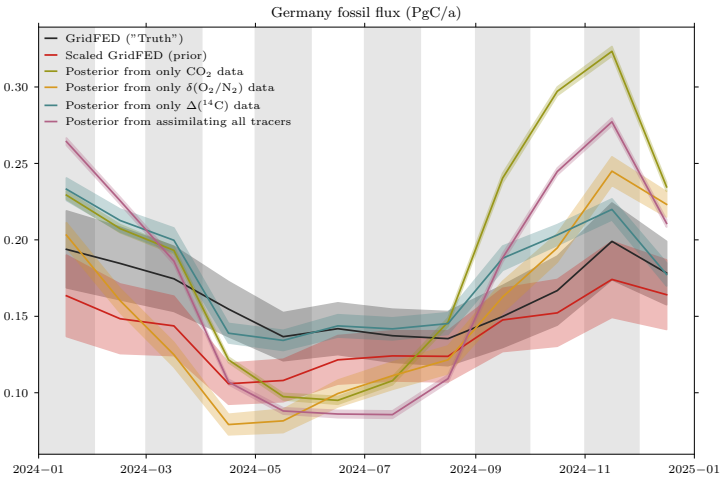
Monitoring and Verification of fossil fuel emissions using $\Delta(^{14}C)$ and $\delta(O_2/N_2)$

Fossil fuels emissions decreases atmospheric:

1. $\Delta(^{14}C)$. (No ^{14}C in ff emissions)
2. $\delta(O_2/N_2)$. (Combustion consumes oxygen)
3. can the inversion retrieve fossil flux

CORSO final deliverables


CBW Cabauw 51°58'N, 4°56'E, 2 masl
CO_2
450
400
$\Delta(^{14}C)$
0
−50
$\delta(O_2/N_2)$
−750
−1000
−1250
Observed
Simulated
2024−01
2024−03
2024−05
2024−07
2024−09
2024−11
2025−01
time



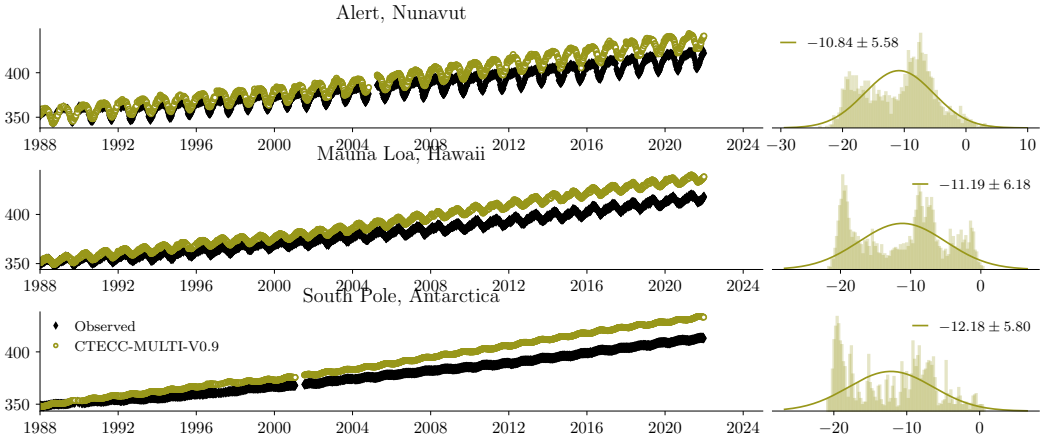
OSSE-Like experiment

1. GridFED (black) as truth
2. Natural global surface flux consistent with gridfed
3. Red flux simulates underreporting

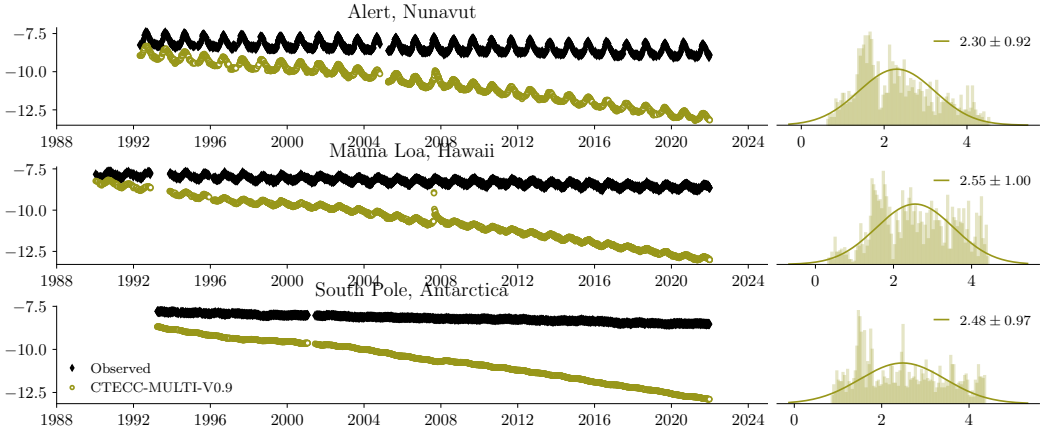
Multi-tracer inversions

30 + years of carbon fluxes estimates consistent with CO_2, $\delta(O_2/N_2)$, and $\delta(^{13}C)$, $\Delta(^{14}C)$ observations

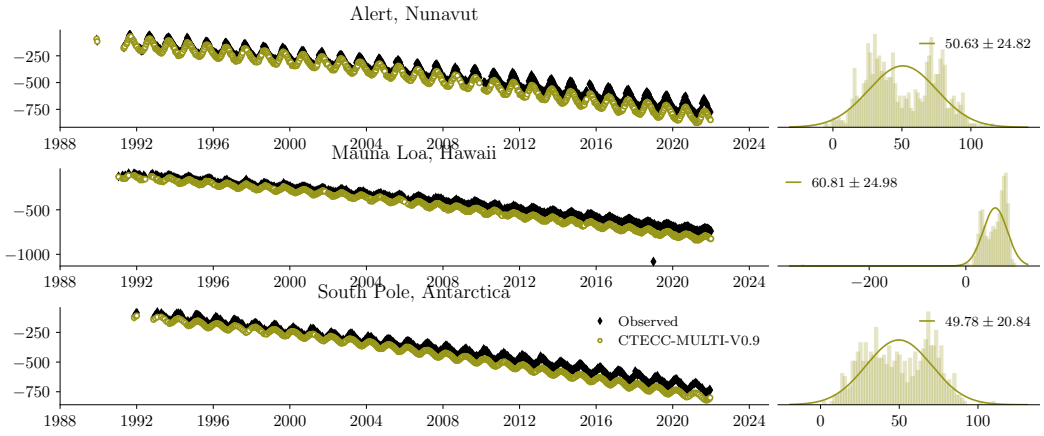
Time series: CO_2/ppm



Time series: $\delta(^{13}C)/‰$



Time series: $\delta(O_2/N_2)$ per meg



Looking ahead

Higher resolution transport

Speed up TM5

TM5 with 150 tracers 1x1 34 levels 3 day run

AMD Epyc genoa 180 cores (nx=4,ny=45)

base + ctdas

component	time (s)	relative to total
total run	519.23	100 %
mpi comm	164.33	31 %
set mass***	51.41	9.9 %
readrecord	44.38	8.6 %
advectyz*	58.7	11.3 %
advectx*,**	139	26.9%
vertical	36.33	6.9 %

*Excluding MPI COMM

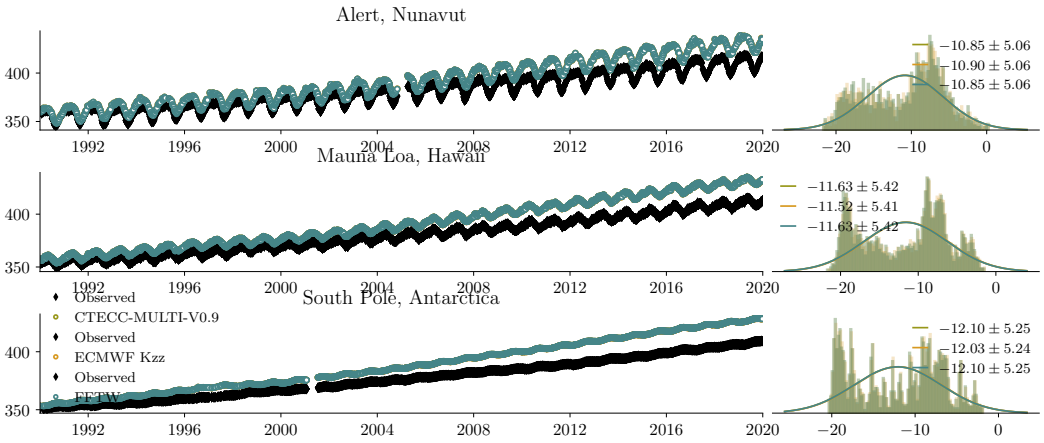
**With domain transform this is about 30 seconds

*** excluding mpi comm and readrecord

FFTW and Kzz

two experiments:

1. Implemented the FFTW library as drop-in replacement of the Singleton algorithm
    1. architecture dependent optimization (cache blocking, vectorization schemes)
2. Reading of ecmwf Kzz vs computing Kzz (see my presentation of the last meeting.)


Alert, Nunavut
Mauna Loa, Hawaii
South Pole, Antarctica
Observed
CTECC-MULTI-V0.9
Observed
ECMWF Kzz
Observed
EETW
−10.85 ± 5.06
−10.90 ± 5.06
−10.85 ± 5.06
−11.63 ± 5.42
−11.52 ± 5.41
−11.63 ± 5.42
−12.10 ± 5.25
−12.03 ± 5.24
−12.10 ± 5.25
350
400
1992
1996
2000
2004
2008
2012
2016
2020
−20
−10
0

ECMWF Kzz vs TM5:

1. Differences seem statistically insignificant for remote sites (for $\delta(^{13}C)$, $\Delta(^{14}C)$, $\delta(O_2/N_2)$ as well)
2. To analyze still: in-situ and tower and aircaft

FFTW speed: top down

3x2xtropo34 1 month

FFTW

```
step init set mass 60.64
  mpi comm         17.87 ( 29.5 %)
  tmm readfield 2D  1.89 (  3.1 %)
  tmm readfield 3D  4.91 (  8.1 %)
  other            35.97 ( 59.3 %)
```

Singleton implementation

```
step init set mass 104.49
  mpi comm          17.61 ( 16.9 %)
  tmm readfield 2D   2.82 (  2.7 %)
  tmm readfield 3D   6.25 (  6.0 %)
  other             77.81 ( 74.5 %)
```

fft is used in MassBalanceFluxes, here in "other"

fftw is about 30 seconds faster (50 percent speedup compared to base)

Hypothetical speedup from current developments

Not measured but estimated!

component	old	new	further speedup?
total run	519.23	384	
mpi comm	164.33		some
set mass***	51.41	25	MPI(fftw/distribute over layers)
readrecord	44.38		
advectyz*	58.7		architecture dep
advectx*,**	139	30	architecture dep
vertical	36.33		gemm

*Excluding MPI COMM

**With domain transform this is about 30 seconds

*** excluding mpi comm and readrecord

NOT TM5 but PyEnkf

Challenge:

1. Large amount of data (satellite data)
2. Large statevectors

Solution: Cache blocking + OpenMP

1. cache aware loop reduces memory fetches
2. enkf loop that scales well beyond (openmp)

1e6 statevector elements obs in under 7 minutes (vs 4 hours of the original loop)

https://github.com/JJDHooghiem/pyenkf

(Provocative? -) Food for thought on tm5 future speedup

1. OpenMP/OpenACC are deceptively simple
2. During porting application to gpu part of the speedup is simply clean up
    1. for example removing branching (if statements)
3. Going to higher resolution in tm5 computational speed is not our main concern