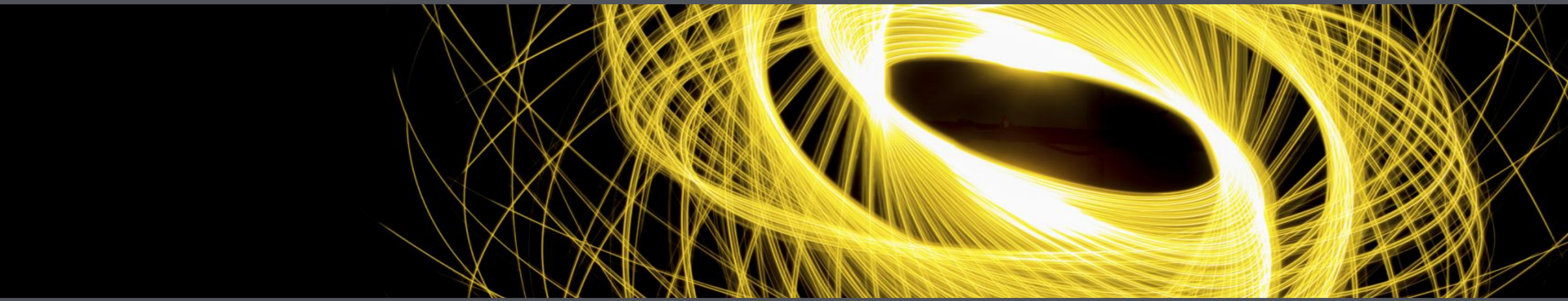
Department of Meteorology
Data assimilation research centre (DARC)

Data Assimilation Research Centre

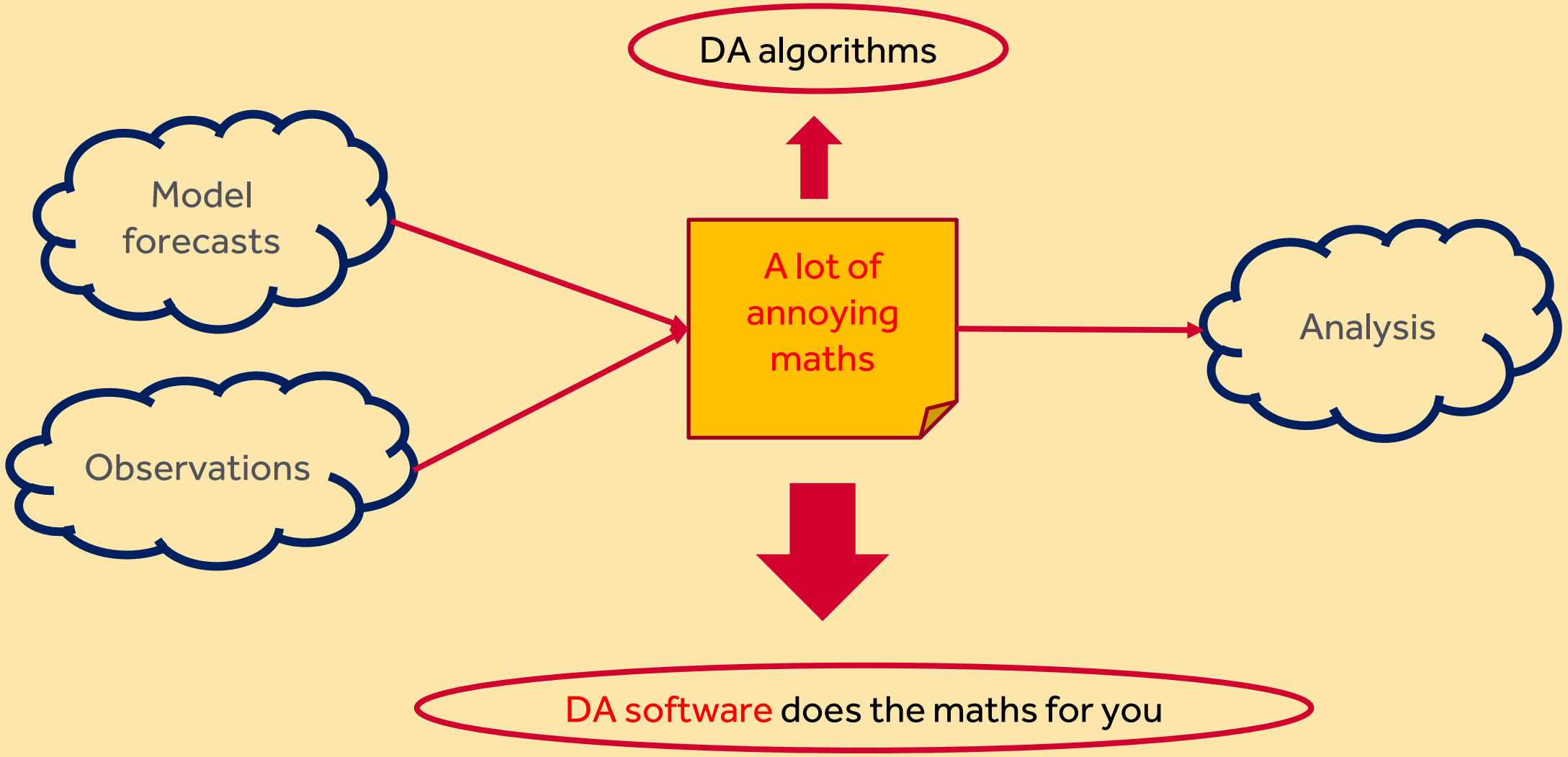
National Centre for Earth Observation
NATURAL ENVIRONMENT RESEARCH COUNCIL

University of Reading

Data assimilation software (general)



Yumeng Chen
yumeng.chen@reading.ac.uk


DA algorithms
Model forecasts
Observations
A lot of annoying maths
Analysis
DA software does the maths for you

When do we want to write our own code?

- Not very complicated algorithms
- Learning details of an algorithm
- Sense of control
- Easier modifications
- Tailored to our own applications
- Fewer dependent libraries

...

Why bother with DA software?

$$\mathcal{J}(\mathbf{x}_0) = \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}^b)^{\mathrm{T}}\mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}^b) + \frac{1}{2}\sum_{i=0}^{N}(\mathcal{H}_i(\mathbf{x}_i) - \mathbf{y}_i)^{\mathrm{T}}\mathbf{R}_i^{-1}(\mathcal{H}_i(\mathbf{x}_i) - \mathbf{y}_i)$$

Algorithm 1.2 4D-Var in its basic form

$j = 0$, $\mathbf{x} = \mathbf{x}_0$
while $||\nabla J|| > \epsilon$ **or** $j \leq j_{\max}$
  (1) compute J with the direct model M and H
  (2) compute ∇J with adjoint model $\mathbf{M}^{\mathrm{T}}$ and $\mathbf{H}^{\mathrm{T}}$ (reverse mode)
  gradient descent and update of $\mathbf{x}_{j+1}$
  $j = j + 1$
end

Source: Data Assimilation: Methods, Algorithms, and Applications by Maëlle Nodet, Marc Bocquet, Mark Asch

- Convenience
  - Prefer np.mean to writing loops
- Optimised and efficient
- Well-maintained and reliable
- Focus on the scientific questions
- Ensures reproducible and consistent scientific research

$$\mathcal{J}(\mathbf{x}_0) = \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}^b)^{\mathrm{T}}\mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}^b) + \frac{1}{2}\sum_{i=0}^{N}(\mathcal{H}_i(\mathbf{x}_i) - \mathbf{y}_i)^{\mathrm{T}}\mathbf{R}_i^{-1}(\mathcal{H}_i(\mathbf{x}_i) - \mathbf{y}_i)$$

Algorithm 1.2 4D-Var in its basic form

$j = 0$, $\mathbf{x} = \mathbf{x}_0$
while $\|\nabla J\| > \epsilon$ **or** $j \leq j_{\max}$
  (1) compute J with the direct model M and H
  (2) compute ∇J with adjoint model $\mathbf{M}^{\mathrm{T}}$ and $\mathbf{H}^{\mathrm{T}}$ (reverse mode)
  gradient descent and update of $\mathbf{x}_{j+1}$
  $j = j + 1$
end

Source: Data Assimilation: Methods, Algorithms, and Applications by Maëlle Nodet, Marc Bocquet, Mark Asch

Why bother with DA software?

One may even argue that we can build a DA system without a team using good software.

Example DA software/libraries/framework

Name	Developers	Primary Language
JEDI	JCSDA + collaborators	C++
PDAF	AWI	Fortran
DART	NCAR	Fortran
NEDAS	NERSC	Python
MIDAS	ECCC	Fortran
…		

Name	Developers	Primary Language
DAPPER	NERSC	Python
EnKF-C	Sakov	C
TorchDA	Cheng	Python
…		

There is also software for specific purposes, e.g. land DA, snow DA, weather DA, etc.

Data Assimilation with Python: a Package for Experimental Research (DAPPER)

- DAPPER is a set of templates to provide a benchmark for different DA methods
- The typical set-up is a synthetic (twin) experiment
- Ease of adding new DA methods and models
- Pure Python implementation with multiprocessing support
- It is not suited for very big models (>60k unknowns)

Method	Literature reproduced
EnKF [1]	Sakov08, Hoteit15, Grudzien2020
EnKF-N	Bocquet12, Bocquet15
EnKS, EnRTS	Raanes2016
iEnKS / iEnKF / EnRML / ES-MDA [2]	Sakov12, Bocquet12, Bocquet14
LETKF, local & serial EAKF	Bocquet11
Sqrt. model noise methods	Raanes2014
Particle filter (bootstrap) [3]	Bocquet10
Optimal/implicit Particle filter [3]	Bocquet10
NETF	Tödter15, Wiljes16
Rank histogram filter (RHF)	Anderson10
4D-Var	
3D-Var	
Extended KF	
Optimal interpolation	
Climatology	

https://github.com/nansencenter/DAPPER

Data Assimilation with Python: a Package for Experimental Research (DAPPER)

Model setup

Obs. setup

Time steps, observation frequency, etc.

```
import dapper.mods as modelling
import dapper.da_methods as da

HMM = modelling.HiddenMarkovModel(dyn, obs, tseq, x0)
xx, yy = HMM.simulate()

xp = da.EnKF("Sqrt", N=10, infl=1.02)
xp.assimilate(HMM, xx, yy)
```

Initial condition

Truth (xx) and synthetic observations (yy)

DA set up and run DA

Check model-specific configuration files for setting up dyn, obs, tseq, x0

Using DA for large models: JEDI or PDAF

University of Reading

Functionalities	PDAF	JEDI
Methods	Focuses on EnKF; also supports (Hybrid) 3DVar and nonlinear filters	EnKF + Var + Hybrid
Language	Fortran/Python	Primarily C++
Model Coupling	Adjustments in user-supplied functions. Non-intrusive to the model.	C++ interface class which needs to wrap the model components + YAML files manipulation.
Observation operator	User-defined by user-supplied code (can utilise existing Fortran code)	Various built-in operators and QC processes
Error covariance modelling	Ensemble generation method	Ensemble and parametric B-matrix modelling
Software installation	Minimal dependencies	Requires specific machine or correct container configuration
Communities	Research-driven	Operational centres but also used by the research community

PDAF Parallel Data Assimilation Framework

University of Reading

Parallel Data Assimilation Framework (PDAF)

- Suitable for weather and climate models, e.g. AWI-CM, MITgcm, MPI-ESM, NEMO, etc.
- Implementation is in general efficient and reliable
- Good protocols and well-documented to be used with any models and observations
- Recently PDAF version 3.0 is released

	Global filter	Local filter	Smoother
EnKF	ETKF	✓	✓
	ESTKF	✓	✓
	EnKF	✓	✓
	SEIK	✓	✓
	SEEK		
Nonlinear filtering	NETKF	✓	✓
	Particle filter		
3DVar	3DVar		
	3DEnVar		
	Hyb3DVar		

https://pdaf.awi.de/trac/wiki

PDAF Parallel Data Assimilation Framework

University of Reading

Parallel Data Assimilation Framework (PDAF)

- A typical workflow of PDAF:

Initialise PDAF

```
CALL PDAF_init(filtertype, subtype, 0,  filter_param_i, 2, filter_param_r, 1,
COMM_model, COMM_filter, COMM_couple,  task_id, n_modeltasks,
filterpe, init_ens,  screen, status_pdaf)
CALL PDAF_init_forecast(next_observation, distribute_state,  prepoststep,
status_pdaf)
```

Assimilation in model time stepping

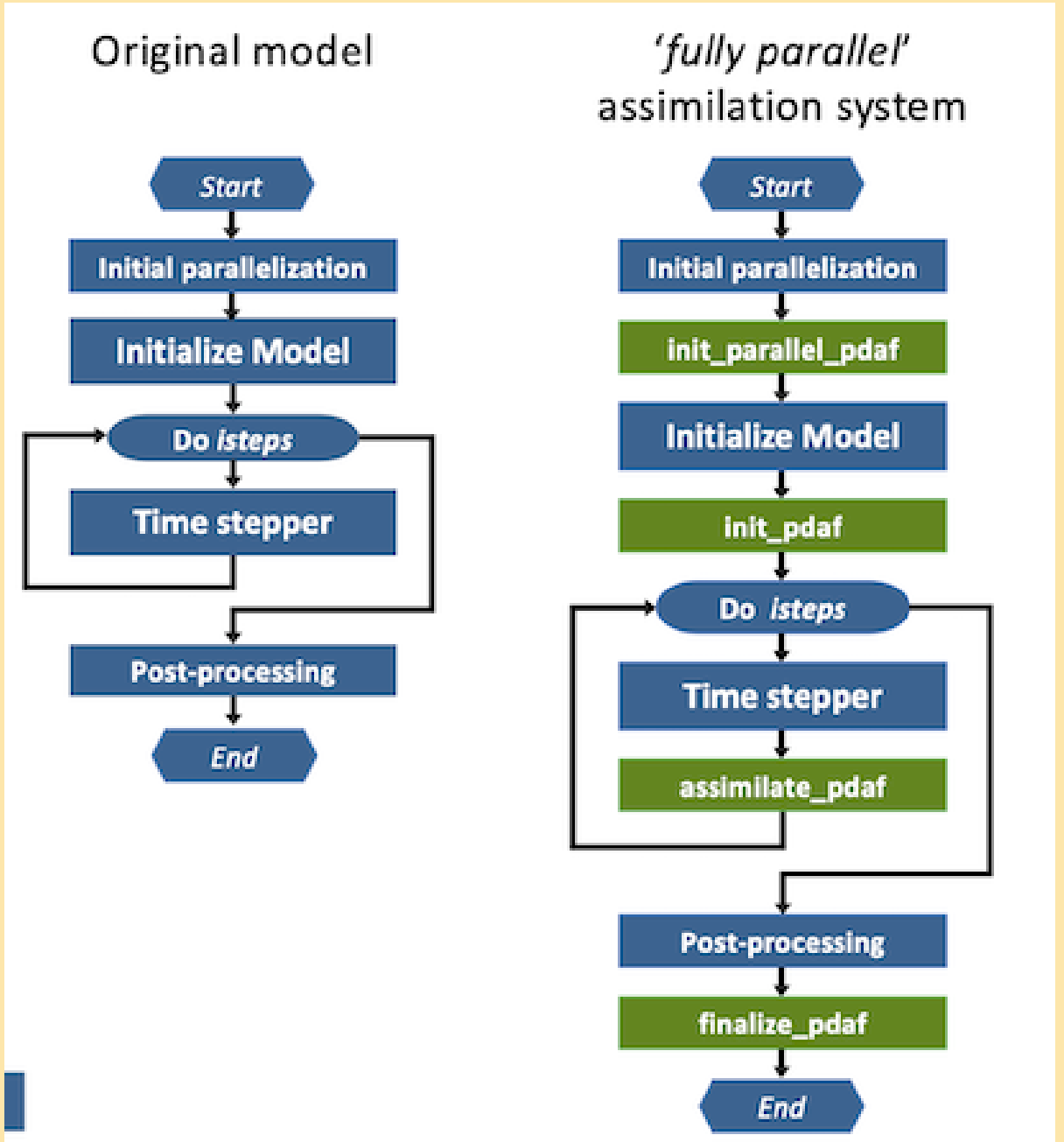
```
CALL PDAF3_assimilate(collect_state, distribute_state, init_dim_obs,
obs_op,  init_n_domains, init_dim_l, init_dim_obs_l,  prepoststep,
next_observation, status_pdaf)
```

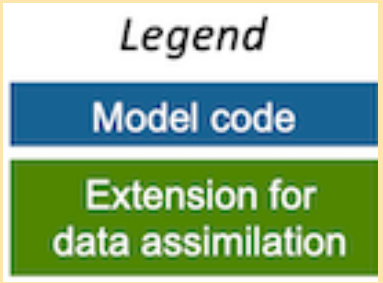
Finalise PDAF

```
CALL finalize_pdaf()
```

Blue: user-supplied functions

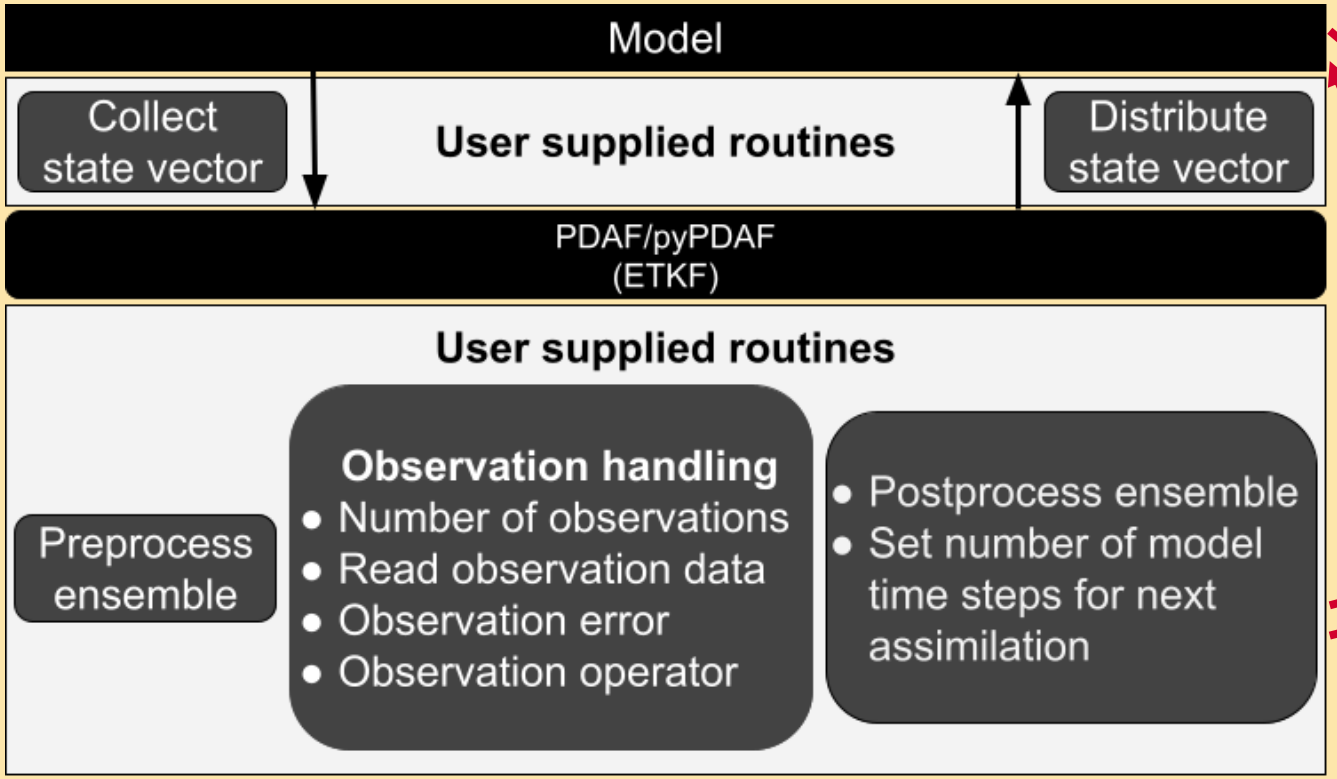
- Adding addition subroutines with limited subroutine calls in existing model





Parallel Data Assimilation Framework (PDAF)

- Flexibility of PDAF relies on user-supplied routines (grey boxes)



$$\mathbf{x}_{\mathrm{a}}^{i} = \mathbf{x}_{\mathrm{b}}^{i} + \mathbf{K}^{i}\left(\mathbf{y}^{i} - h\left(\mathbf{x}_{\mathrm{b}}^{i}\right)\right)$$

- Types of user-supplied routines vary for different filters; check the PDAF website for references.

PDAF Parallel Data Assimilation Framework

University of Reading

Parallel Data Assimilation Framework (PDAF)

- PDAF also has a Python version, pyPDAF:
  - https://github.com/yumengch/pypdaf
- Build a simple DA system using pyPDAF:
  - https://colab.research.google.com/github/yumengch/pyPDAF/
  - If you run the notebook on your local computer:
    - conda create -n pypdaf -c conda-forge yumengch::pypdaf jupyter matplotlib
  - Any feedback on pyPDAF and the tutorials are welcome!