Speed Up Macroeconomic Modeling with MATLAB and Parallel Computing

Eduard Benet Cerdà

01 - Oct - 2025

“we have a BVAR model that we would need to speed up”

“we have a DSGE model that we would need to speed up”

“we have a SSM model that we would need to speed up”

“we have a Regime Switching model that we would need to speed up”

…

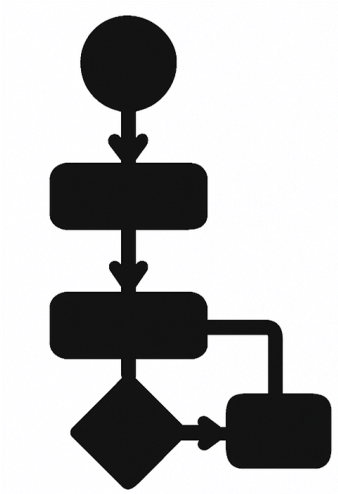
Why is speed important?

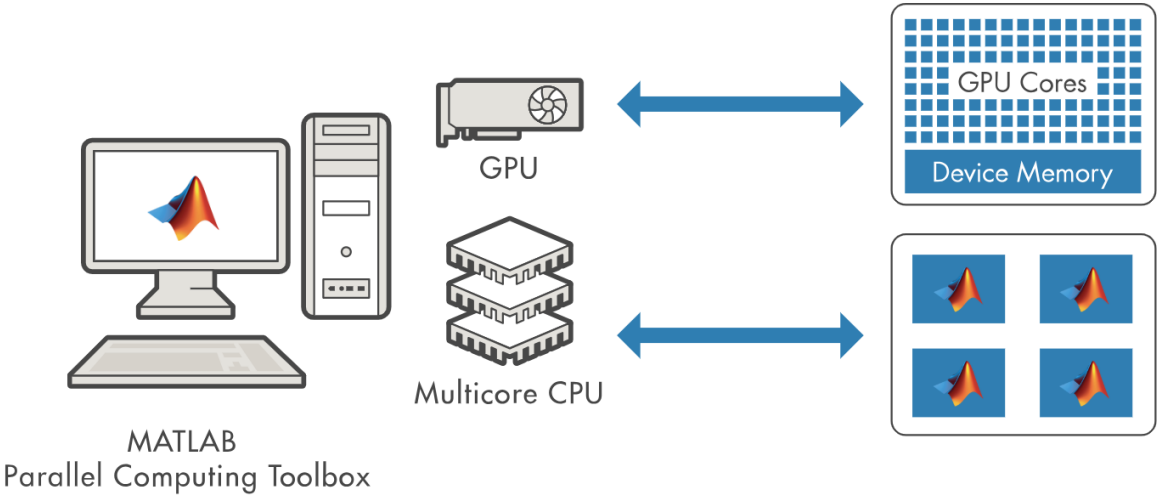
- **Long Computation times**
  - Many macroeconomic models can take 20, 40, or even 100+ hours to run a single estimation.
- **Hardware Utilization**
  - There is often uncertainty about whether current computational resources are being fully leveraged.
- **Scalability needs**
  - Access to more powerful computers is now easier
  - There is a need to distribute calculations across multiple computers
- **Repeated Evaluations**
  - Tasks such as parameter sweeps and Monte Carlo simulations require running many independent model evaluations, which multiplies the total computation time.
  - It also requires a good documentation and coordination of inputs and outputs

Approaches to Speeding Up Modeling

Refactor and optimize existing code for efficiency.

Adopt more efficient techniques (e.g. auto-diff) or develop new algorithms



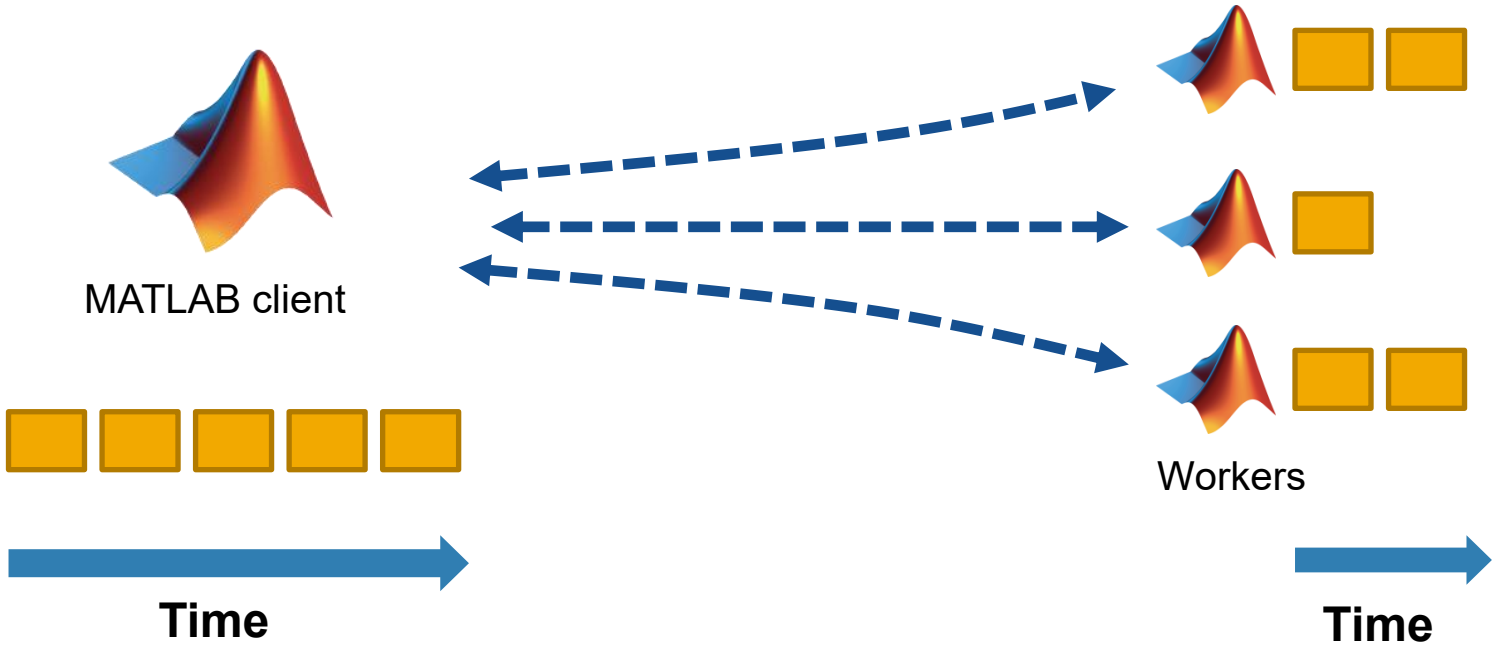


Harness the power of modern multi-core and cloud computing to distribute workloads.

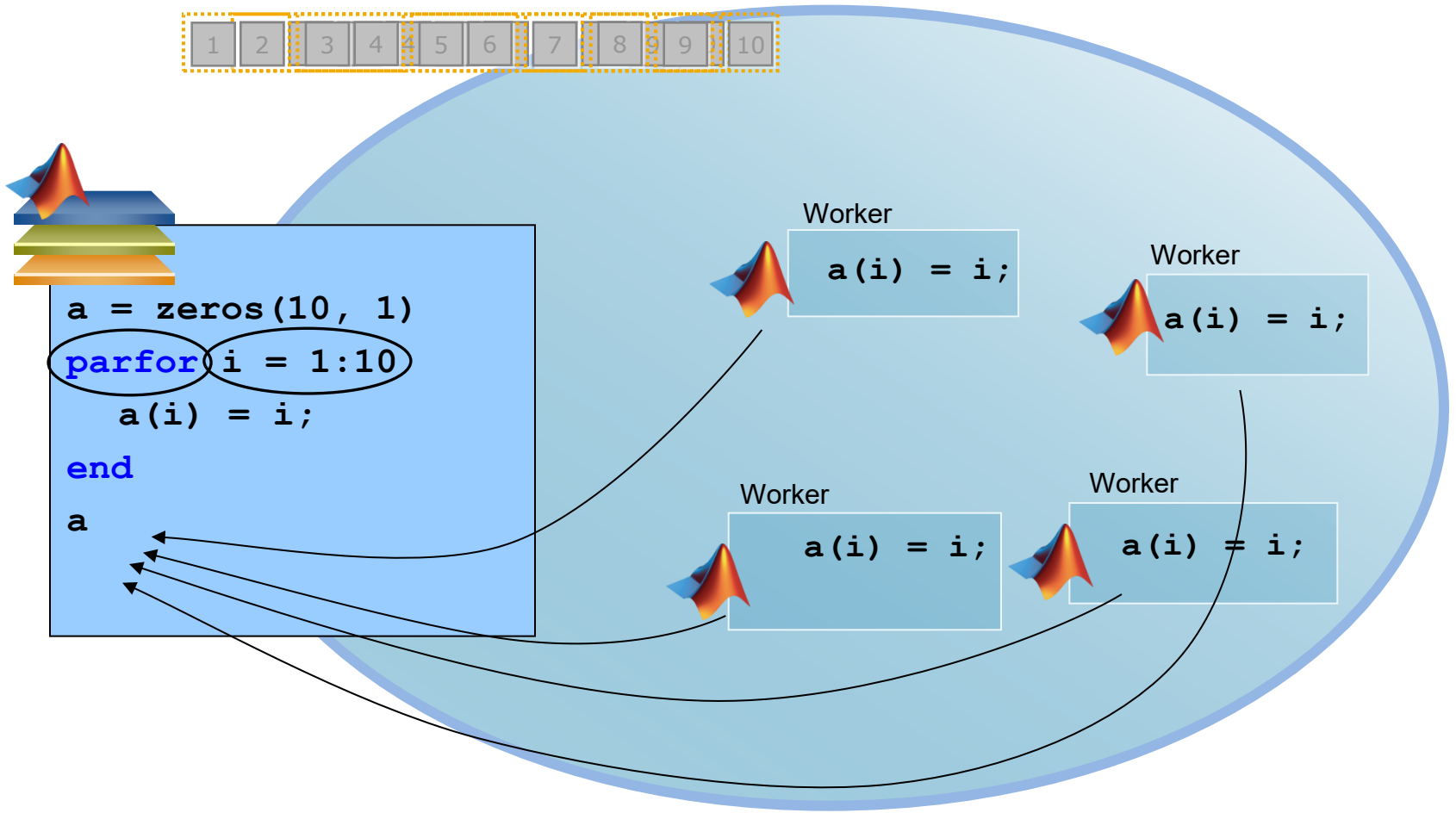
What is parallel computing?

```
for i = 1:5
      y(i) = myFunc(myVar(i));
end
```

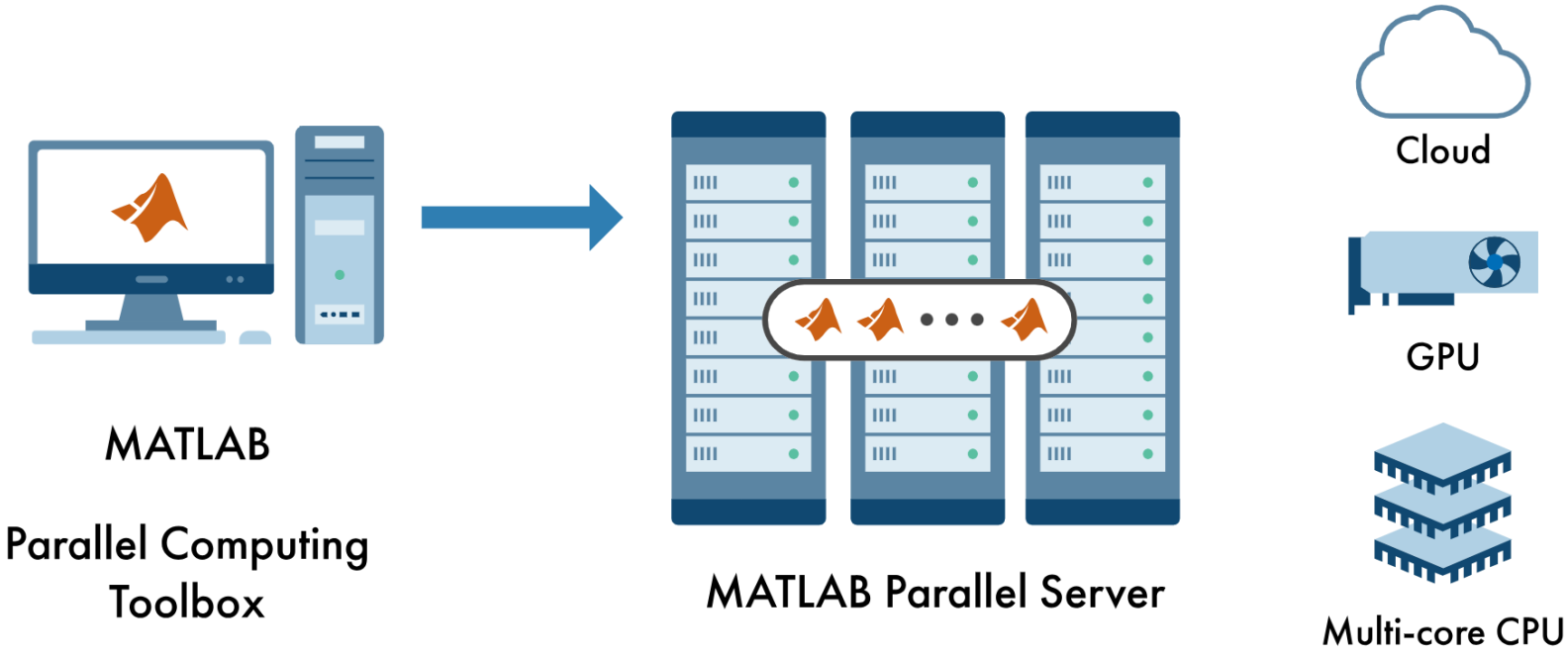
```
parfor i = 1:5
      y(i) = myFunc(myVar(i));
end
```



Mechanics of `parfor` Loops



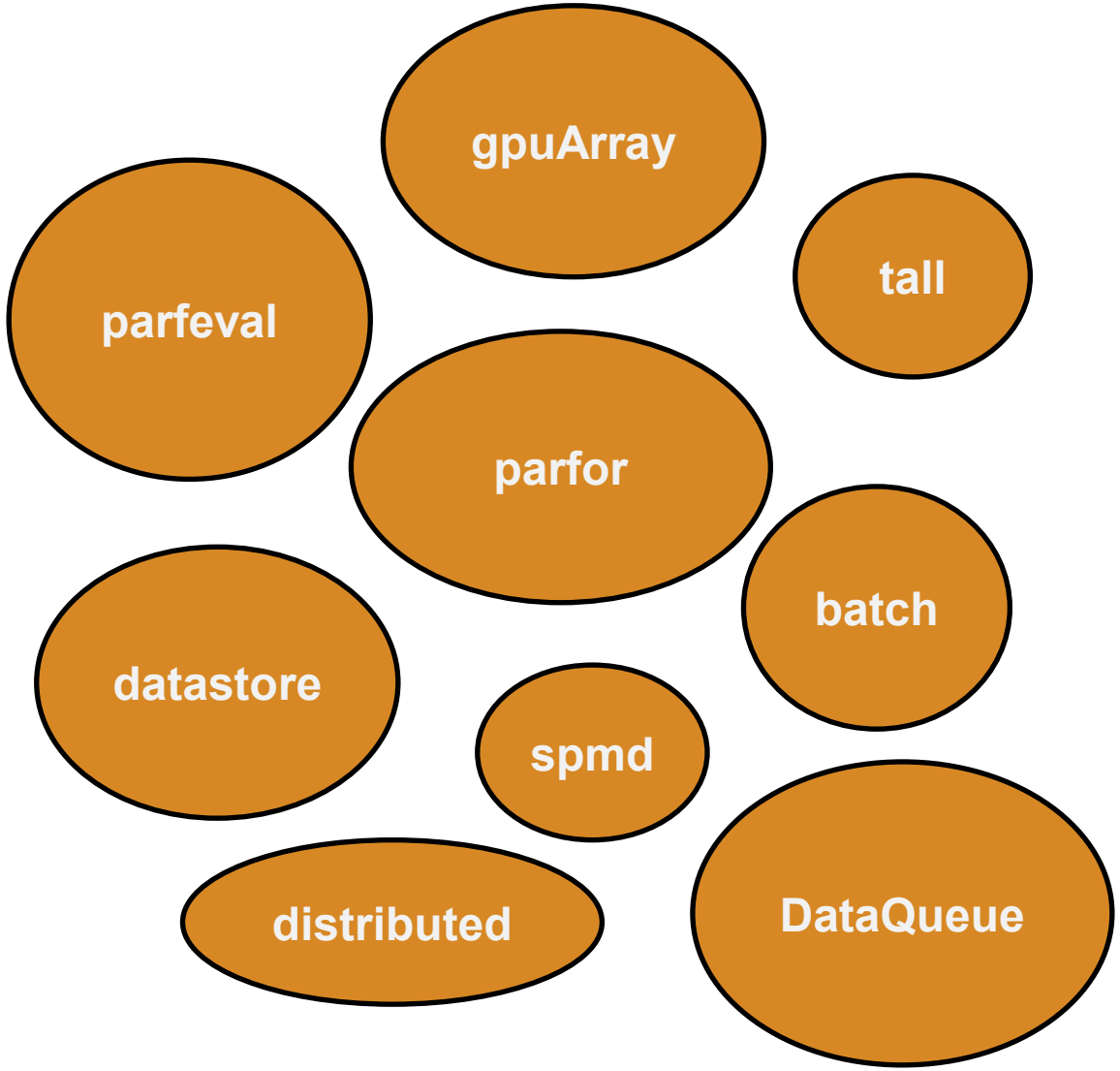
Where does it run?



You have full control on where and how the code the code runs, locally, or in the cloud

```
c = parcluster('clusterName');
c.NumWorkers = x
c.NumThreads = y

p = parpool(c)
parfor i = 1 : N
```

There is more to parallel computing



Is not always a trivial problem..

- Some problems can be difficult to adapt into a parfor loop
- parfor is not always the solution, e.g. a Gibbs sampling process.
- Is the potential parallelism obvious to the researcher?
- What is the random seed of my workers?
- How costly is the data transfer?

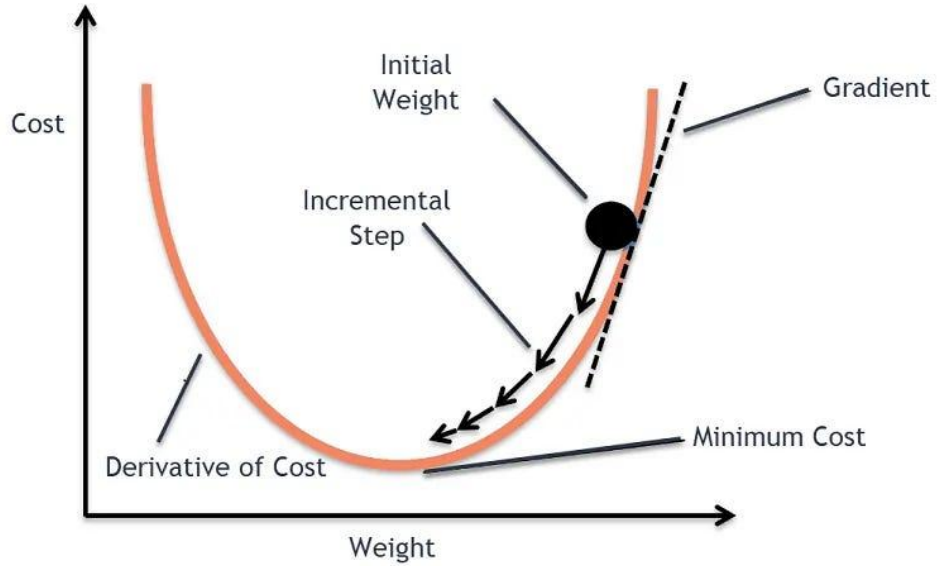
Implementing this requires time and expertise, and there is no silver bullet, but there are common patterns.

Parallelism in optimization problems

A primary computational bottleneck in macroeconomic modeling is often an optimization procedure. Luckily, MATLAB streamlines this process by incorporating parallel computing capabilities into most of its optimization algorithms:

```
options = optimoptions('fmincon', UseParallel = true)
```

This will use parallel computing to calculate the derivative in each direction and step of the optimization process.



Parallelism in optimization problems

A primary computational bottleneck in macroeconomic modeling is often an optimization procedure. Luckily, MATLAB streamlines this process by incorporating parallel computing capabilities into most of its optimization algorithms:

```
options = optimoptions('fmincon', UseParallel = true)
```

This will use parallel computing to calculate the derivative in each direction and step of the optimization process.

An example is ***Prior Selection for Vector Autoregressions (Giannone, Lenza, Primiceri - 2012)*** where the hyperparameters of a BVAR are calculated by maximizing the marginal likelihood

Pseudocode:

```
opts        = optimoptions('fmincon', UseParallel = true)
hyperparams = fmincon(@nlogmdd, x0, upperBound, LowerBound, opts);
Mdl         = conjugatebvarm(nseries, nlags, hyperparams)
```

We might not be calling an optimizer directly…

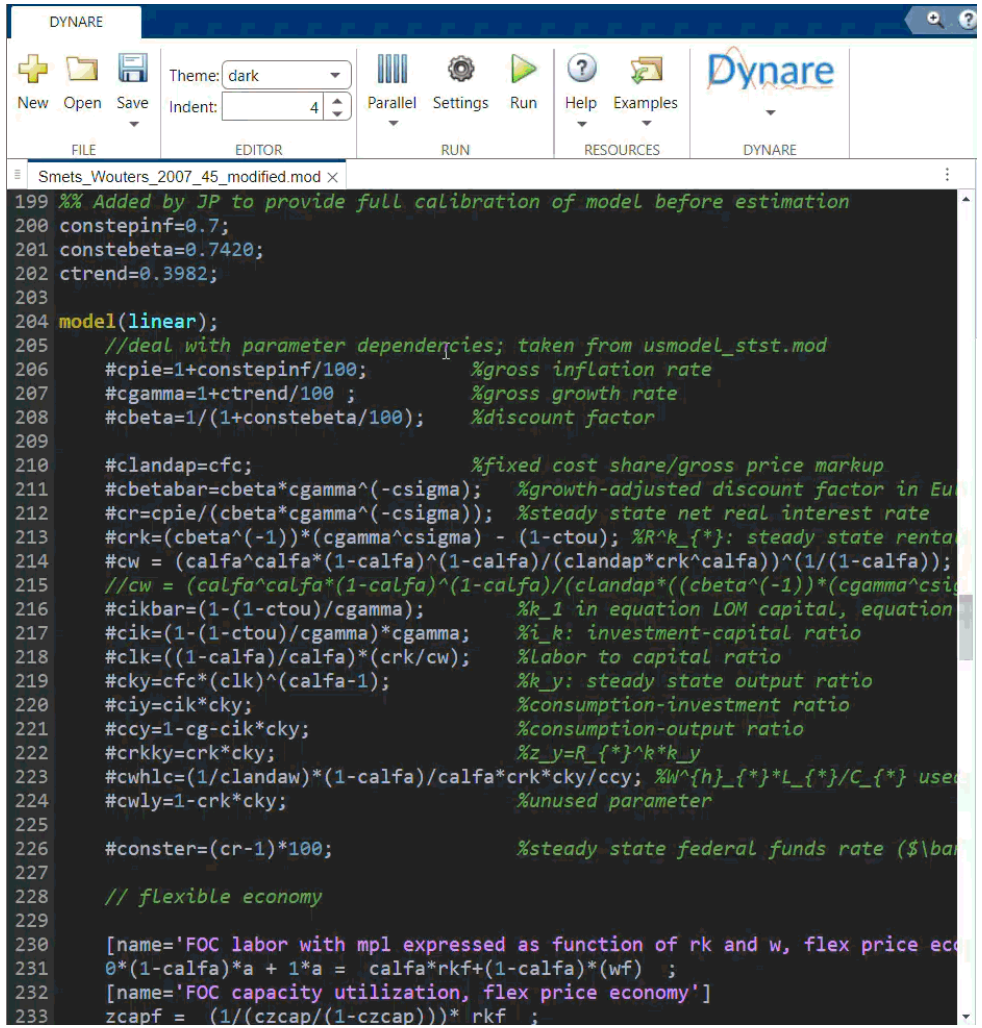
Often, we use a high-level model or tool that wraps the optimization process. For example, the estimation of a Bayesian State-Space requires tunning a proposal distribution (an optimization problem):

```
MdlB = bssm(@dsgeModel, @dsgePrior);
PosteriorMdl = estimate(MdlB,Y,params0);
```

```
opts = optimoptions('fminunc', UseParallel = true);
PosteriorMdl = estimate(MdlB, Y, params0, Options = opts);
```

If the model uses only built-in functions, you can probably take advantage of the above!

What if it was not directly MATLAB, e.g. Dynare?



Shocks and Frictions in US Business Cycles: A Bayesian DSGE Approach *Smets, Frank and Wouters, Rafael (2007)*

- Dynare is a popular tool to enter DSGE models.
- A common option is to estimate a model by maximum likelihood. The code below wraps an “fmincon” problem

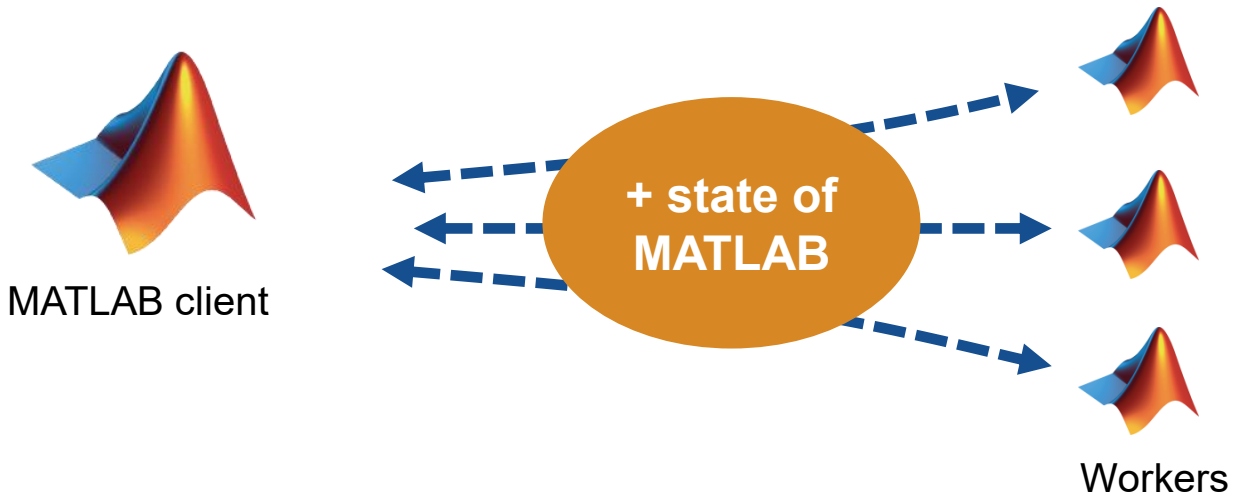
```
estimation(datafile=usmodel_data, mode_compute=1,
           first_obs=1, presample=4, lik_init=2,
           prefilter=0, mh_replic=0, mh_nblocks=2,
           mh_jscale=0.20, mh_drop=0.2, nograph, nodiagnostic, tex);
```

We could be tempted to try this

```
estimation(optim=('UseParallel', true),
           datafile=usmodel_data, mode_compute=1,
           first_obs=1, presample=4, lik_init=2,
           prefilter=0, mh_replic=0, mh_nblocks=2,
           mh_jscale=0.20,mh_drop=0.2, nograph, nodiagnostic, tex);
```

This will not be as trivial… until Dynare 7

- Dynare is a rather complex tool, it uses patterns like variable persistence, and it controls the random seeds. A bit more care needs to be put in distributing parallel work.



- A careful worker initialization is needed

How is this done?

Since this is a very popular question, we put together the following function:

```
estimation(optim=('UseParallel', true),
           datafile=usmodel_data, mode_compute=1,
           first_obs=1, presample=4, lik_init=2,
           prefilter=0, mh_replic=0, mh_nblocks=2,
           mh_jscale=0.20,mh_drop=0.2, nograph, nodiagnostic, tex);
```

Dynare 5, 6

```
>> parpool Processes
>> out = dynareParallel( 'SmetsWouters.mod', Flags = ["nolog"]);
```

Dynare 7 (tentative)

```
>> dynare SmetsWouters.mod nolog
```

On the original model, with "mode_compute=1", a 4 core intel I7-1270P provides a speed-up of ~30% of the time.

DynareUtils toolbox

Other approaches: Bayesian estimation with MCMC chains

A similar problem

This algorithm rea ntense, but highly parallelizabl

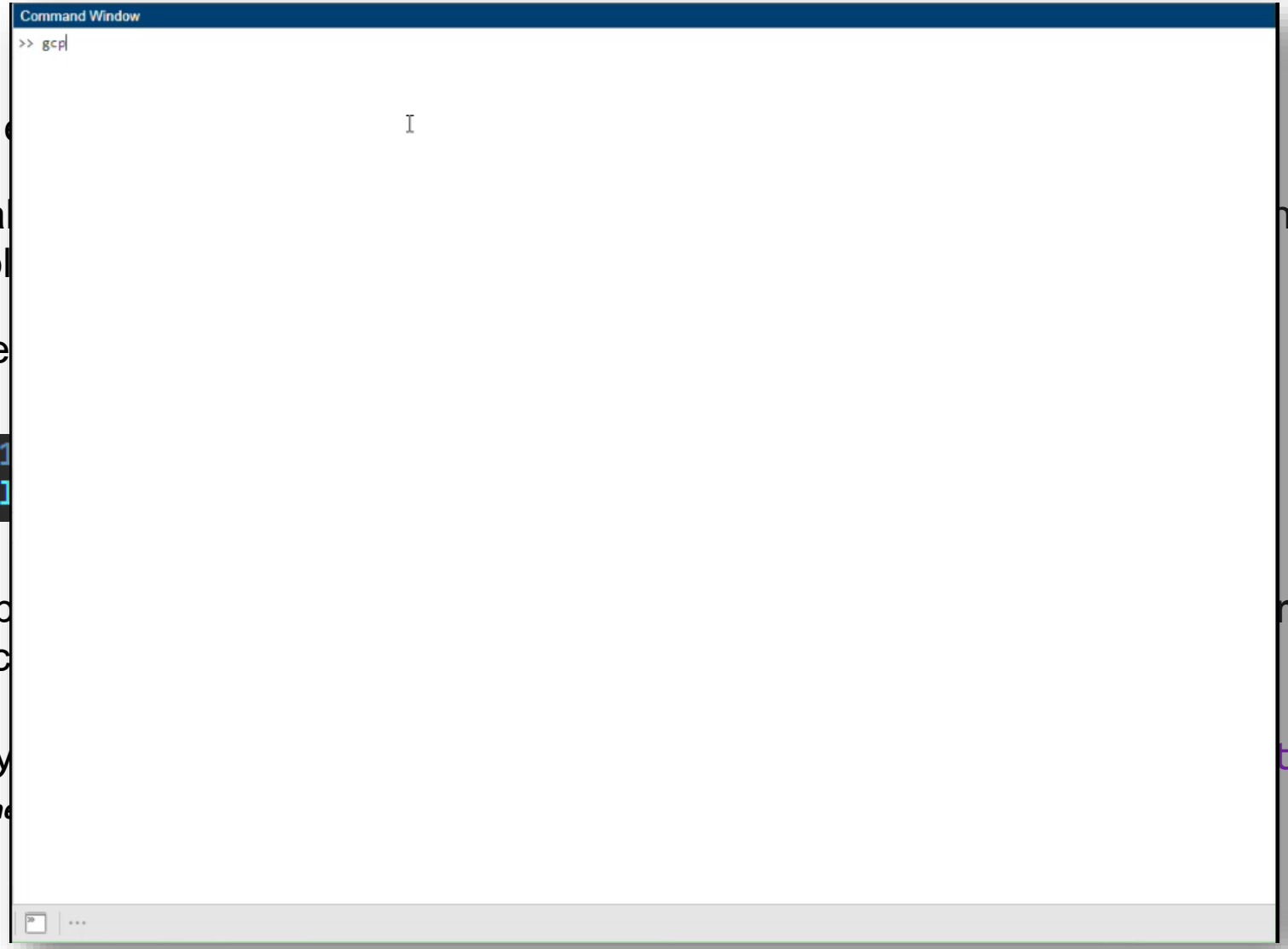
It is a “parfor” type

```
estimation(order=1
           mh_repl
```

The benefits can b **mh_nblocks**” if we have enough c

```
out = dy                                    ta.m')
```

Frank Schorfhe



Why is speed important?

- **Long Computation times**
  - Many macroeconomic models can take 20, 40, or even 100+ hours to run a single estimation.
- **Hardware Utilization**
  - There is often uncertainty about whether current computational resources are being fully leveraged.
- **Scalability needs**
  - Access to more powerful computers is now easier
  - There is a need to distribute calculations across multiple computers
- **Repeated Evaluations**
  - Tasks such as parameter sweeps and Monte Carlo simulations require running many independent model evaluations, which multiplies the total computation time.
  - It also requires a good documentation and coordination of inputs and outputs

Repeated evaluations

```
parfor i = 1 : n
    in = myInputs;
    out = runModel(in);
    results{i} = out;
end
```

A second common workflow is the need for running sensitivity analysis, parameter sweeps, or other processes that require multiple model evaluations

- **Error Handling:** Is your code robust in individual runs?
- **Reproducibility:** Are you systematically recording inputs and outputs to allow for reproducibility and audit trails?
- **Interactivity:** Do you need to visualize intermediate results, and is this process streamlined?
- **Scalability**: Can this code be distributed to multiple machines?

Robust parallel runs can get compilated…

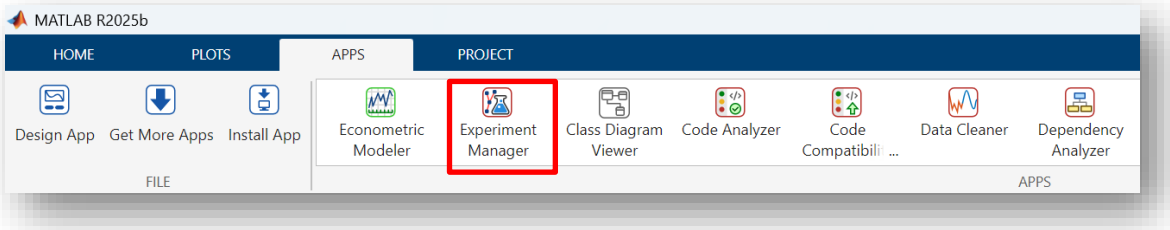
```
f(1:n) = parallel.FevalFuture;
for i = 1 : n
   f(i) = parfeval(@runModel, inputs{i});
end


results = cell(n,1);
for i = 1:n
   [idx,out] = fetchNext(f);
   results{idx} = out;
end
```

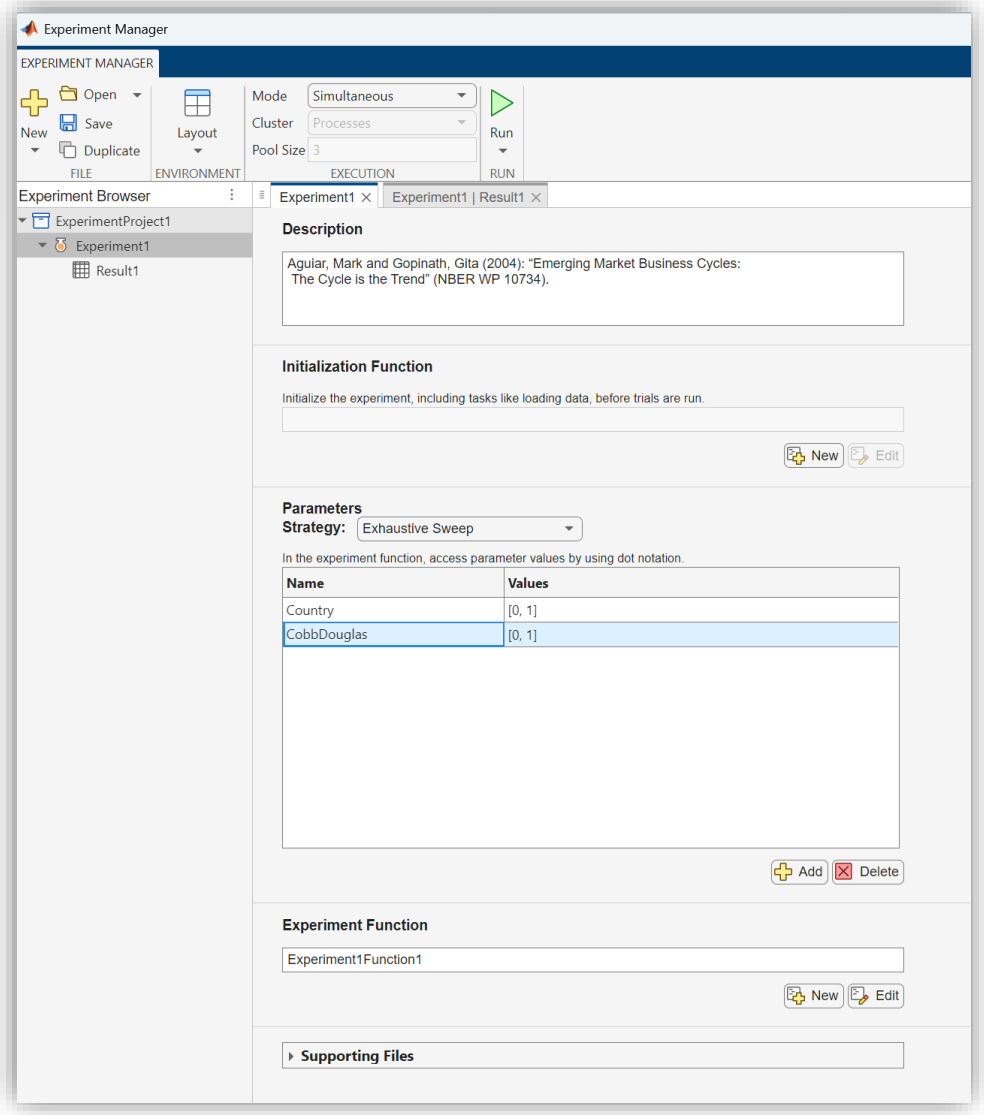

Submit asynchronous independent job

Collect and store the results as they finish (not as we submit them)

Getting robust parallel constructs is not always trivial

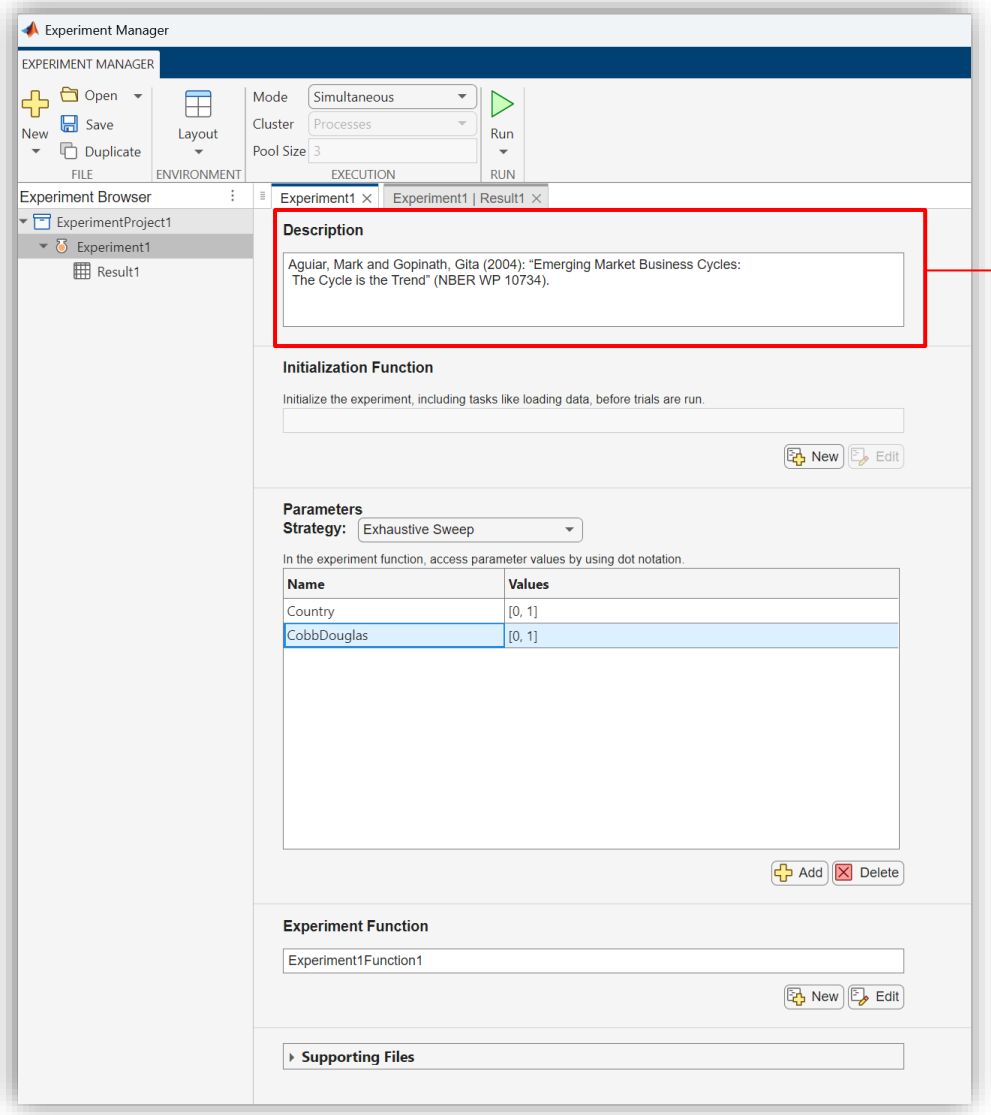
Experiment Manager





- Organizing multiple experiments and the experiment artifacts and results
- Providing visualizations, filters, and annotations for comparing results
- Storing the experiment definition and parameter combinations for each experiment result

Experiment Manager

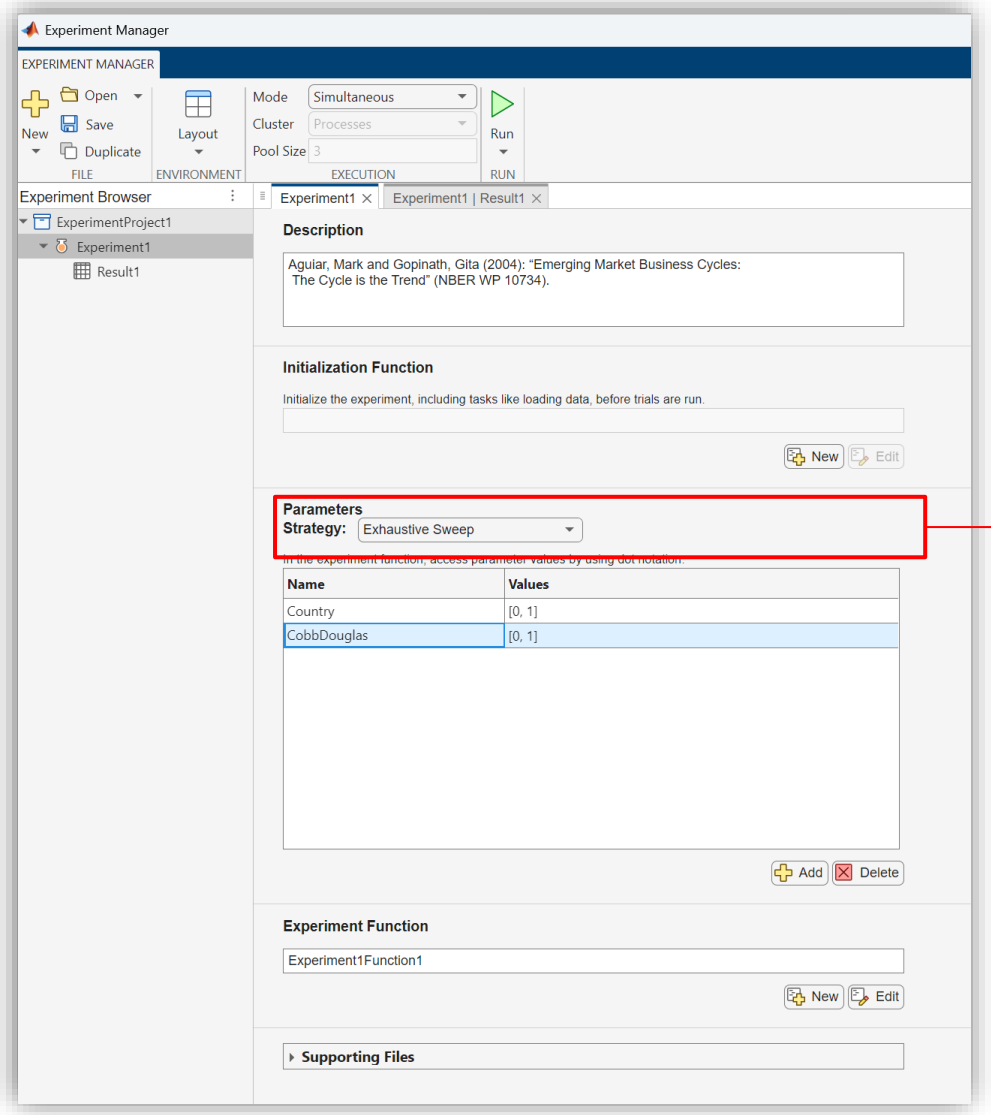


Start with a model parameterized by certain variables. For example, in Dynare

```
// Set the following variable to 0 to get Cobb-Douglas utility
@#define ghh = 1
// Set the following variable to 0 to get the calibration for Canada
@#define country = 1
```

```
dynare myModel.mod -Dghh=1 -Dcountry=0
```

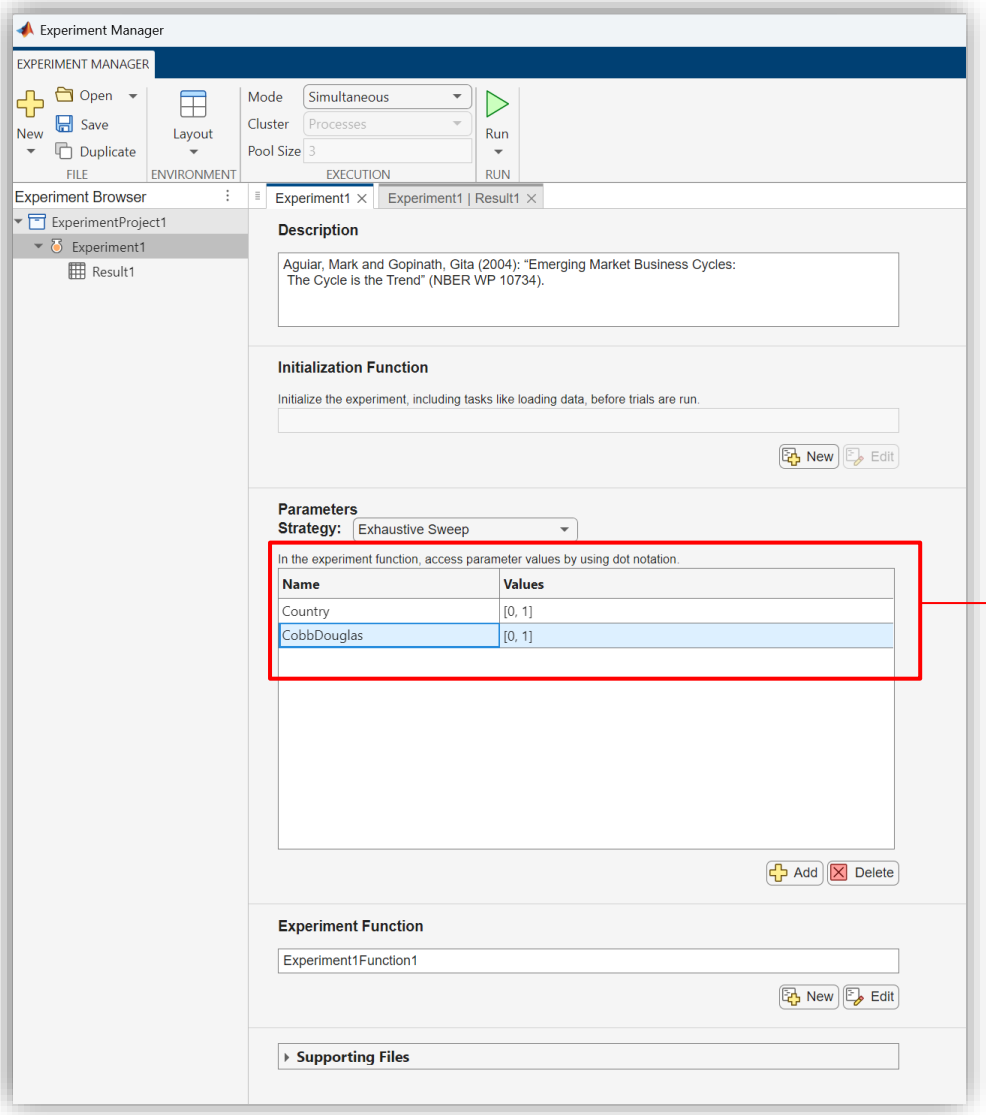
Experiment Manager



Define a strategy to run your model.

- All parameter combinations?
- A random sample?

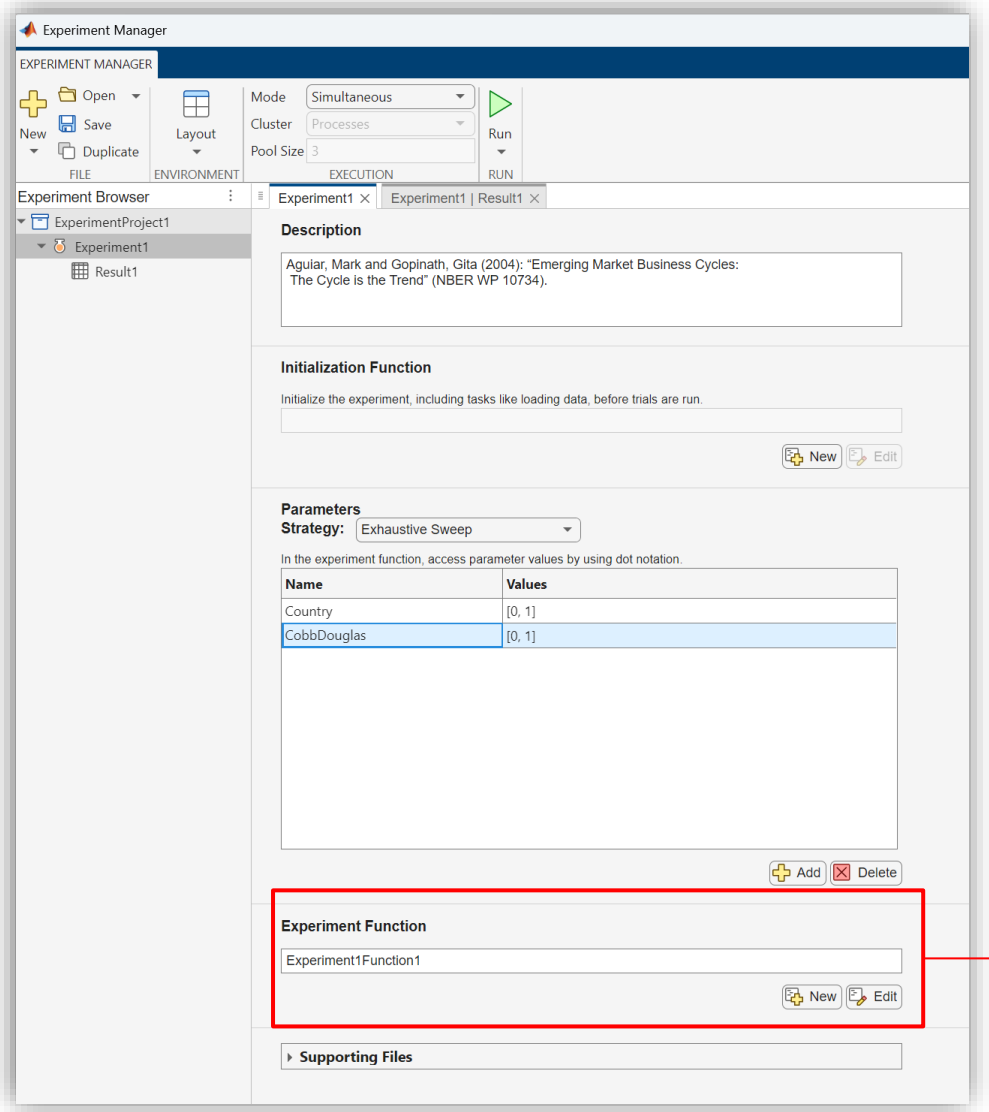
Experiment Manager



State the values for your parameter sweep

```
// Set the following variable to 0 to get Cobb-Douglas utility
@#define ghh = 1
// Set the following variable to 0 to get the calibration for Canada
@#define country = 1
```

Experiment Manager



Experiment Function

```
function [tb_y, c_y, i_y ,oo_] = Experiment1Function1(params)

out = dynareParallel(which('agtrend.mod'),  ...
    Flags = [sprintf("-Dghh=%d",params.CobbDouglas), sprintf("-Dcountry=%d", params.Country), "nolog", "nowarn"], ...
    UseParallel = true);

oo_ = out.oo_;

tb_y = oo_.mean(1);
c_y  = oo_.mean(2);
i_y  = oo_.mean(3);

end
```

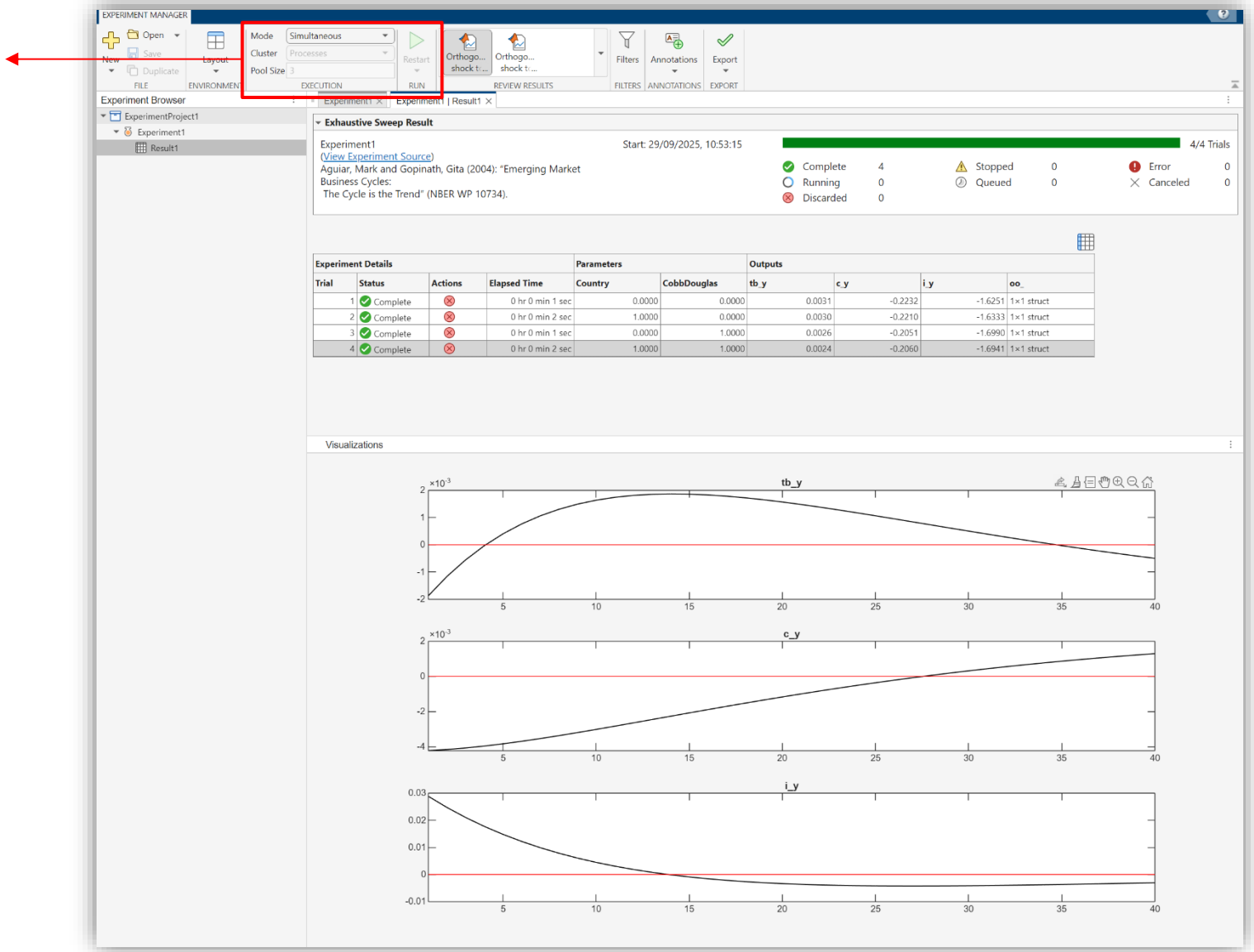
Define the function that calls your model

- Inputs are our parameters
- Recorded outputs are our choice
- All plots are recorded

Experiment Manager

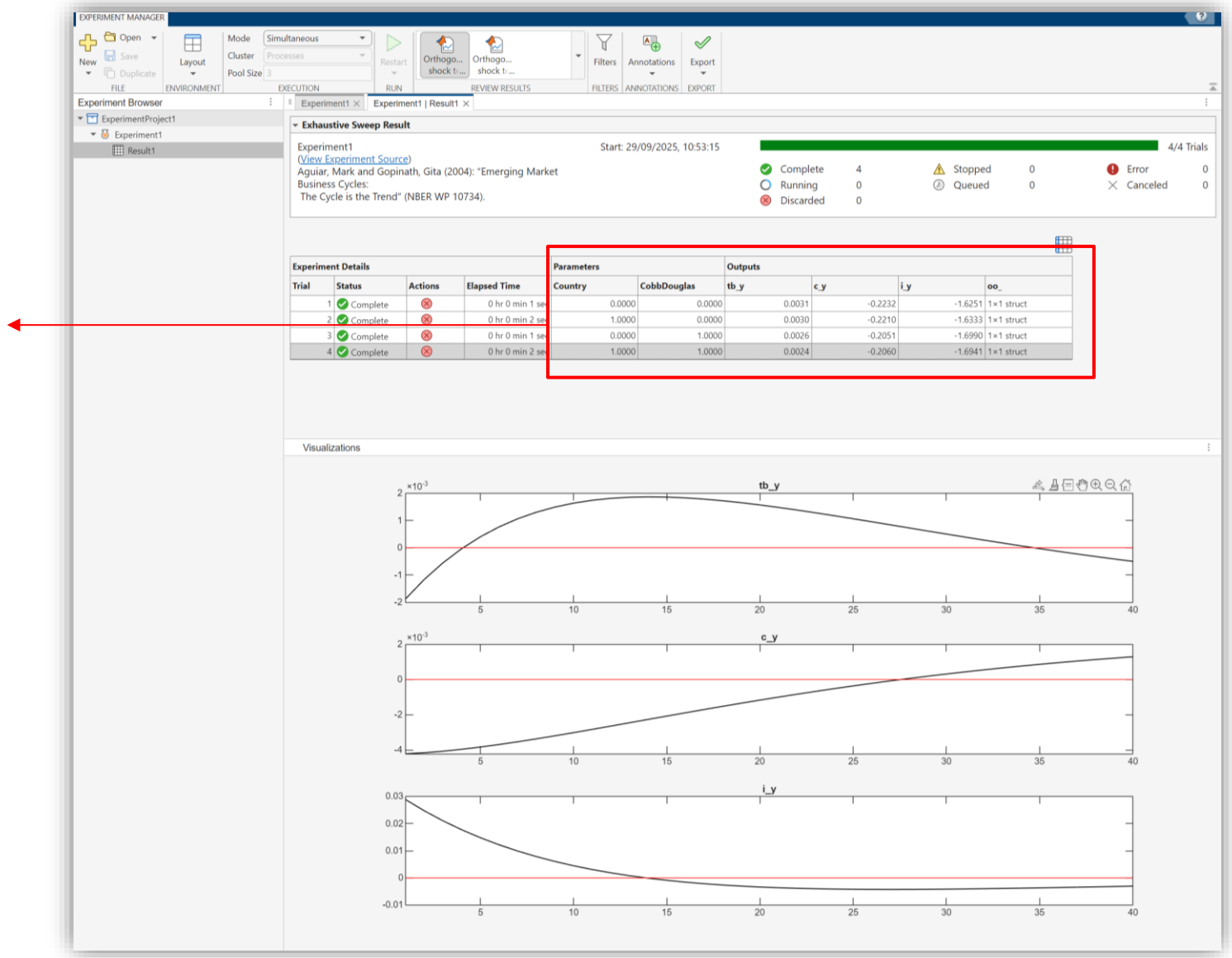
The Experiment Manager will run our experiments sequentially or in parallel

It will also run it in your environment of choice: the current computer or a selected cluster



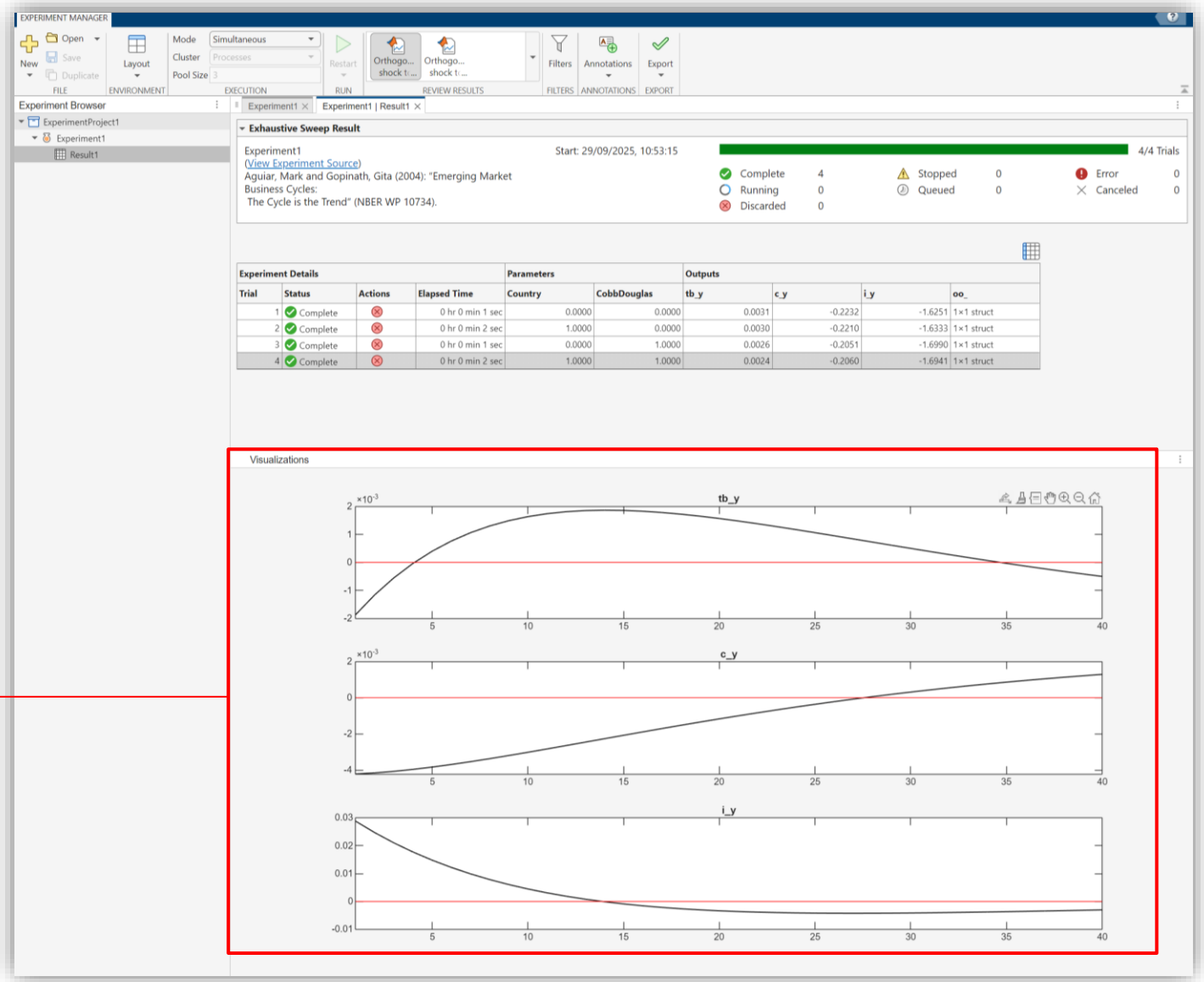
Experiment Manager

It will save all combinations of inputs and outputs stated in the experiment strategy, and it will let us put any notes into each experiment for the future



Experiment Manager

Any plot produced by our function, will also be recorded with its corresponding set of inputs and outputs



Key Takeaways

- **Significant Time Savings**
  - Parallel computing can potentially reduce model estimation times from days to hours
- **Improved Workflow Efficiency**
  - Automate and coordinate large-scale simulation tasks, making research more reproducible and scalable
- **Practical Implementation**
  - MATLAB provides user-friendly constructs (parfor, parfeval, and integration with Dynare) to make parallelization accessible, even for complex macroeconomic models
- **Reach out**
  - If you need help speeding up a model, scaling it, or improving your performance in general, our team is always available to help!

Thank You for joining!

Questions?