REACTION ENGINES

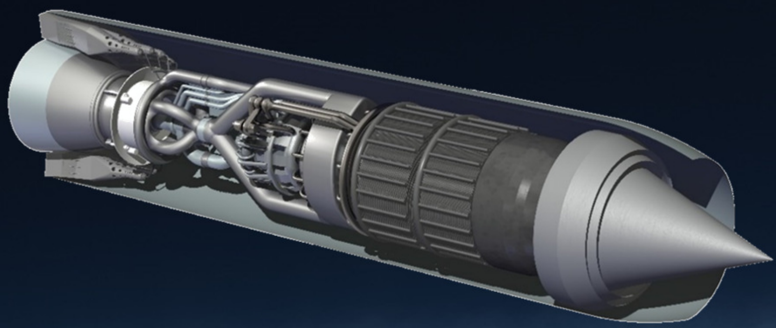
Dynamic Modelling of the SABRE Engine Using Simscape

Dr. Lewis Rees – Reaction Engines Ltd.

MATLAB EXPO 2018 – Silverstone, UK

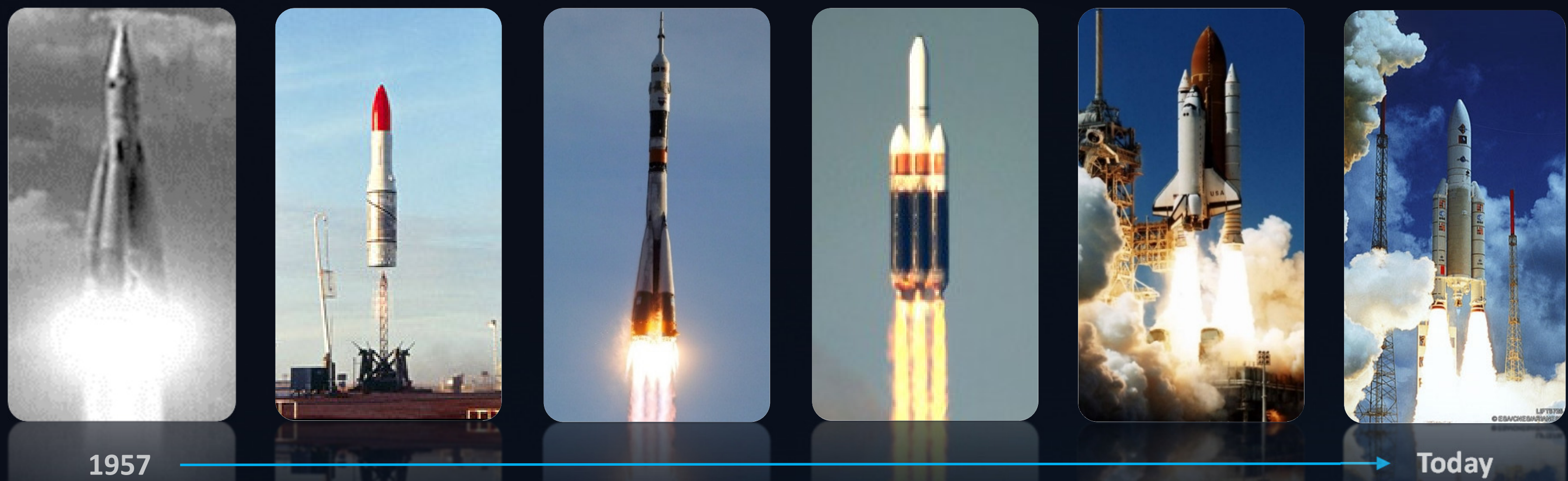
Who Are Reaction Engines?

- British Space Company (150+ employees)
- Based in Abingdon, Oxfordshire
- Founded in 1989
- Developing an advanced combined cycle air-breathing engine
- £60M awarded by UK Space Agency for development of SABRE engine
- Core IP is centred on ultra light weight heat exchangers



UK SPACE AGENCY
BAE SYSTEMS
esa
DARPA
BOEING
ROLLS ROYCE

Current Access to Space Using ELVs



The Restrictions

- Cost ($ 56.5 - $400 million per flight)
- Operations (> 3 month preparation)
- Reliability (2-5% loss rate per flight)

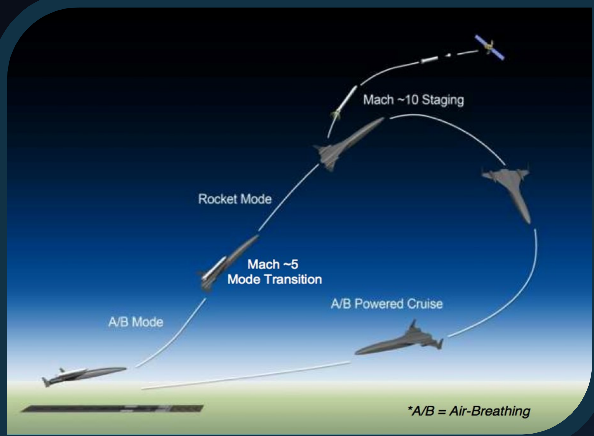
The Outcome

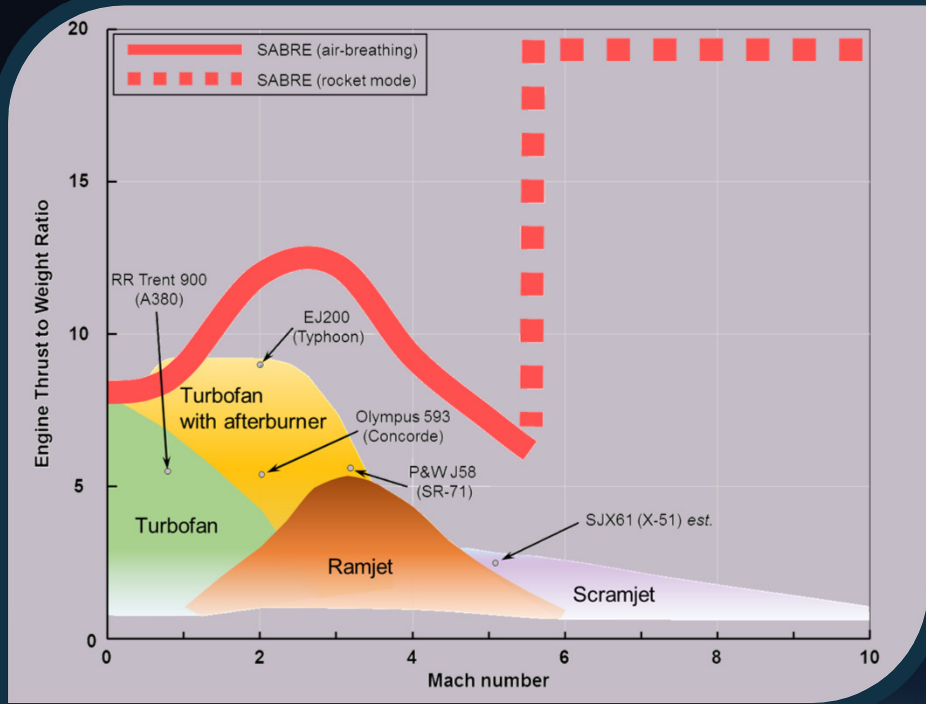
- Only about 80 flights/year worldwide

Reusable Launch Vehicle Option

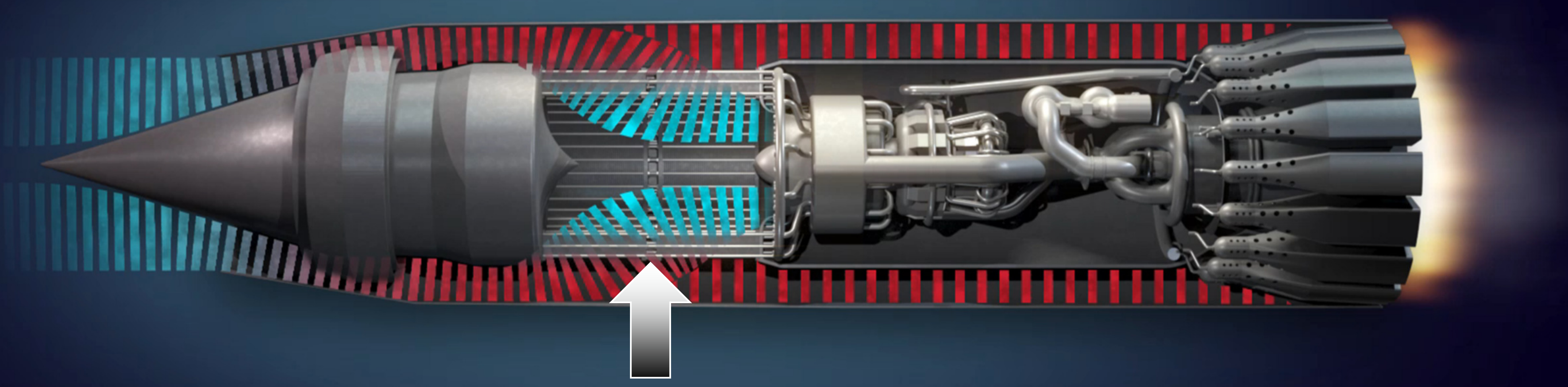
To do this we need a **new** class of propulsion:

- Must be able to operate both in the atmosphere and in space
- Must be efficient at all points during the flight profile

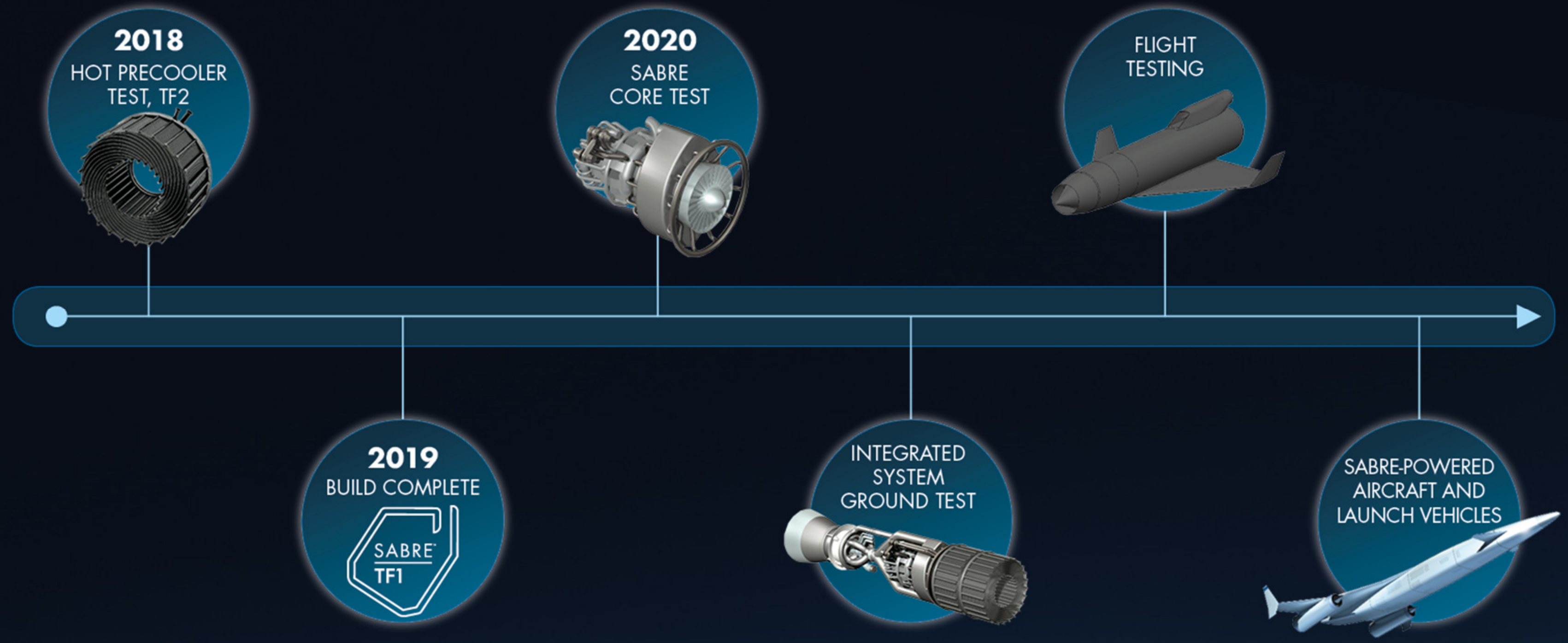




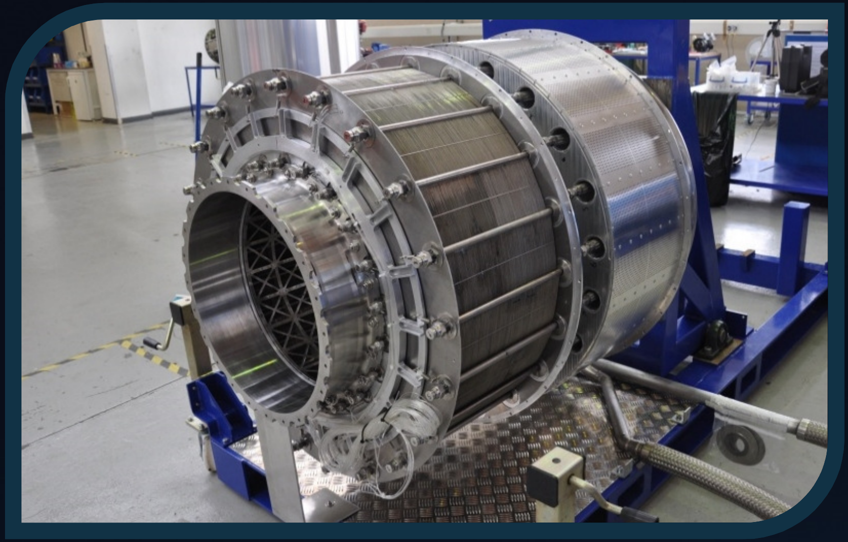
SABRE (Synergetic Air-breathing Rocket Engine)



The SABRE Development Programme



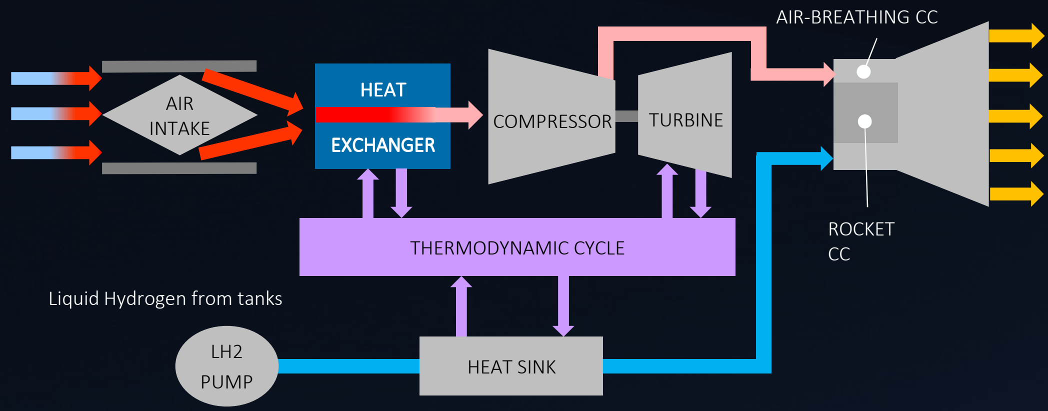
How Far Have We Come?



Dynamic Modelling of the Engine Cycle?

Why do we need a dynamic model?

- Proof of engine cycle concept
- Testing of operability
- Provide subsystems with limits and constraints feedback
- Platform for the control system to be designed around



Why Do We Use Simscape?

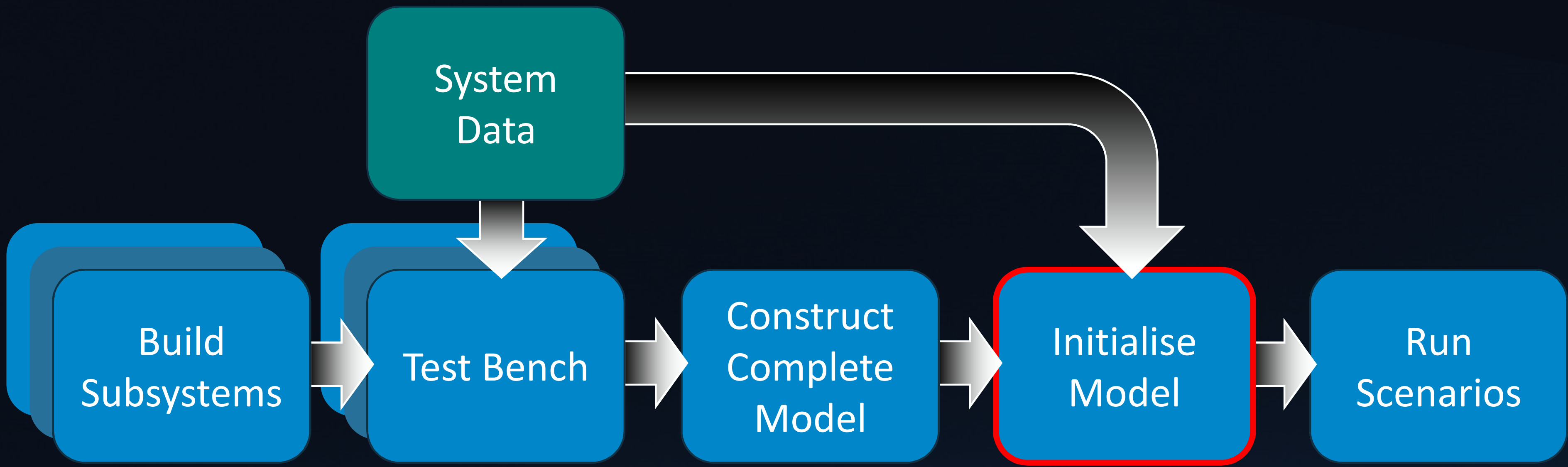
- Flexibility within the MATLAB environment (importing, exporting, manipulation of data, graphics, etc.)
- Familiarity of Simulink
- Expert local support
- Auto-code generation for controllers
- Plant and controller in the same tool
- Integration with **SpeedGoat** for hardware-in-loop testing

Modelling Complex Systems In Simscape

Problems:

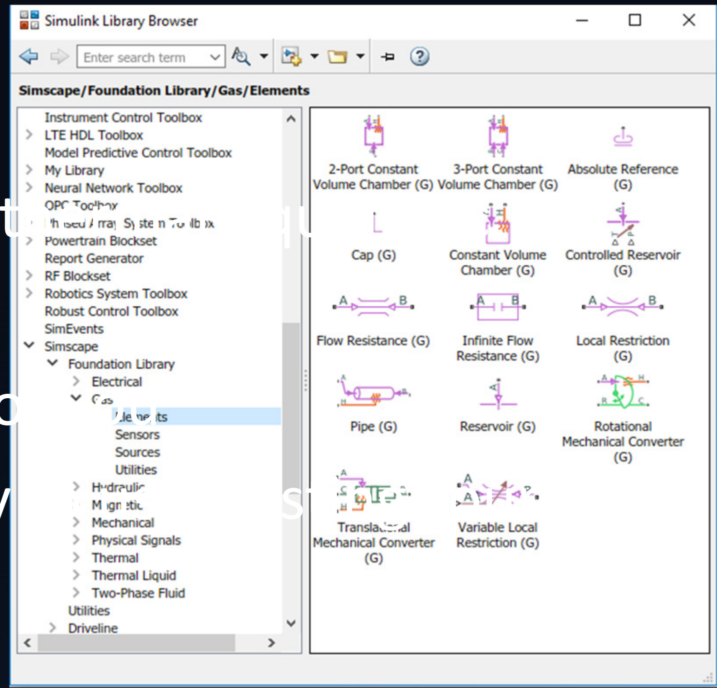
- Integrating large amounts of data
- Cannot expect a complex model to just simulate, regardless of tool used
- High degree of coupling and multiple modes/states
- Hence, **Initialisation is very sensitive to initial conditions**

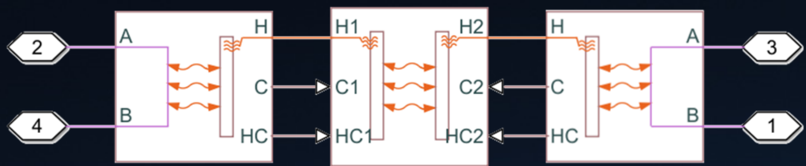
Modelling Process Overview



Building Subsystems

- Try to use a built in Simscape domain (i.e. gas, t
- Try to build functionality out of existing blocks
- Don’t forget MathWorks have done the work fo
- Don’t be afraid to edit your own blocks, but try code to build off of





Data

The Problem:

- You are now a systems integrator (gulp!)

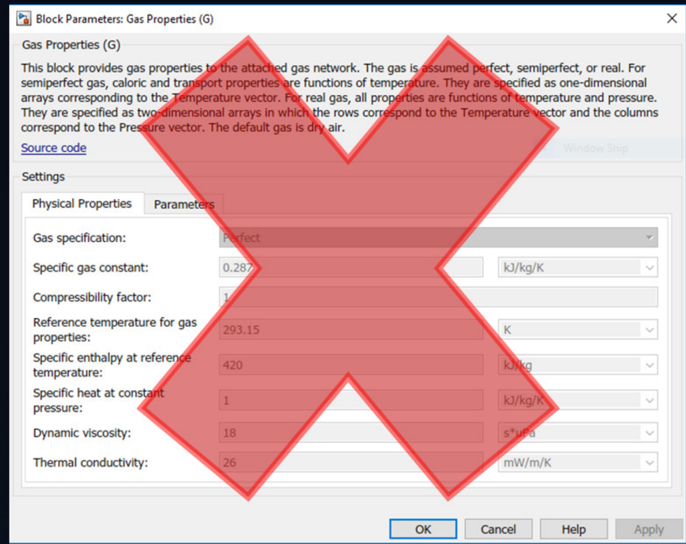
Dealing with data:

- ***TRUST NO ONE*** (assume the data is always wrong)
- Test the data on your test benches
- Define data protocols, structure types etc.
- Automate, Automate, Automate! Will take time, but is worth it

Time to Drop the GUI!

SABRE Dynamic Model contains:
27 ducts
13 valves
10 subsystems
8 boundary conditions
= approx. **850** variables to set

! ! !



- *Write code to automatically populate blocks*

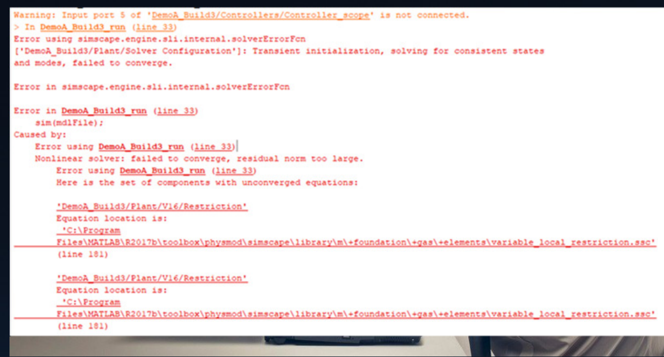
    Use *set_param(<blk>,<param>,<value>)* to set values within blocks

Test Benching of Subsystems

- Test models using typical values and validate against steady state data or hand calculations, if available
- Test model at extremes
- Test benches should run in the same data and configuration environment as your main model
- New data will be continually introduced, so a solid test bench is essential

Model Construction

- By now putting the complete model together should be easy.
- Build model up in stages. Don't just try and put all blocks together and hope it will work… **IT WON'T!**
- Initialization can be very difficult to get to work. Don't lose faith – it's an inherently hard problem, but solvable



https://www.techdotmatrix.com/2017/11/why-are-most-of-the-it-engineers-frustrated-in-their-life/

Getting the Model to Initialisation

What do we mean by initialisation?

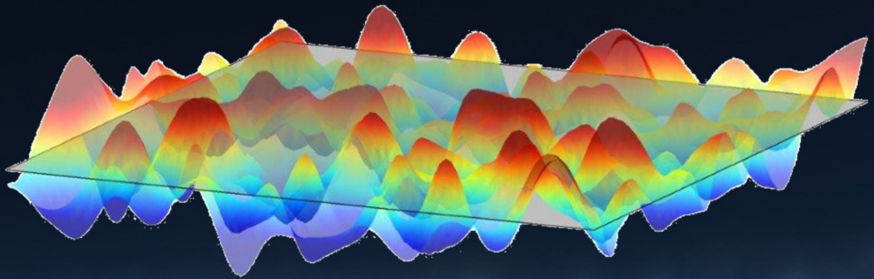
- Solver computing the solution to the first time-step of the simulation

What to do:

- Give the model as much information as possible (steady state data, data from test benching, etc.)
- Don’t try and start at values of zero, certain equations are likely to not work
- Set priorities where required

  Use *set_param(<blk>,’<variable>_priority’,<value>)*
- Change solver tolerance

  Use *set_param(… TBC)*
- Use initialisation aids



https://robot.gmc.ulaval.ca/en/research/research-thrusts/mecanismes-paralleles-entraines-par-cables/3d-foam-printer/

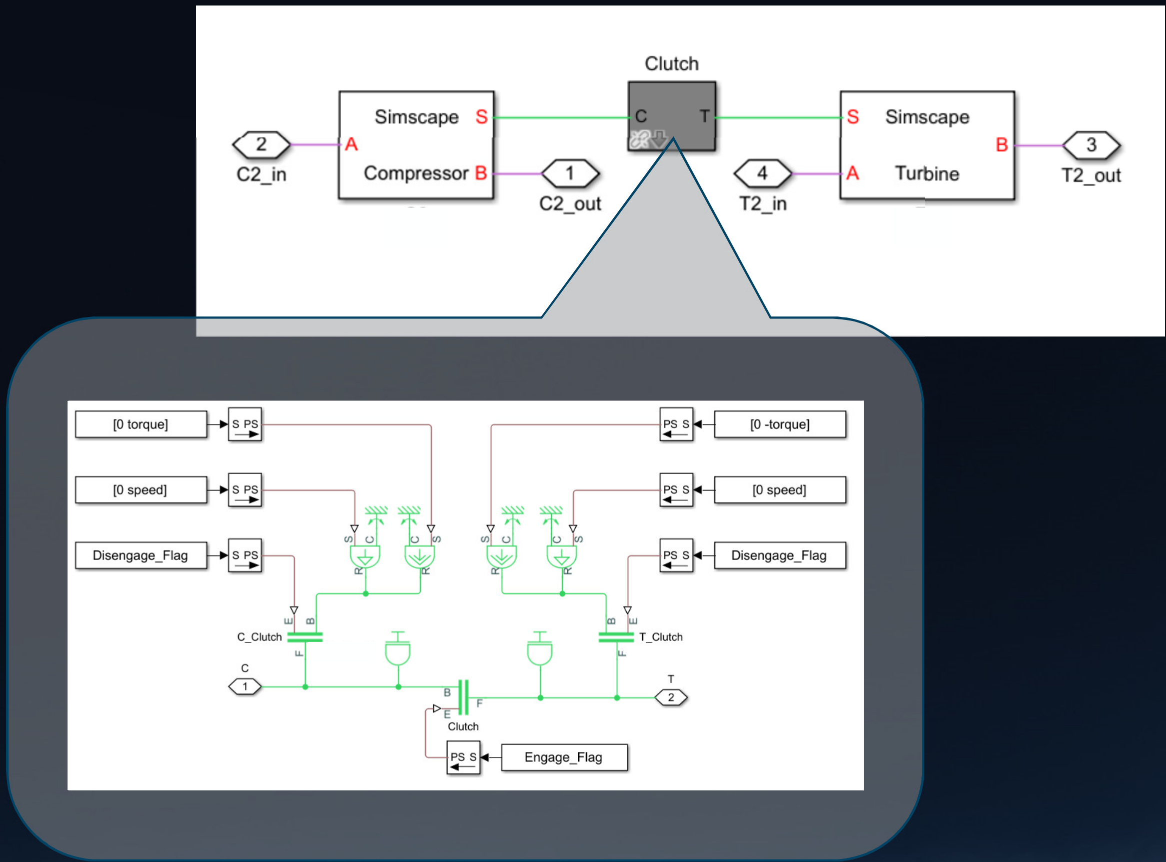
Initialisation Aids

Aim

- Decouple the system
- Allow all components to settle on operation point
- Switch out aid after a given time

Example:

Clutching turbomachinery



Pre-Flight Checks

Check for simple mistakes:

- Compare units in blocks against list of accepted units

  Use *get_param(<blk>,'<variable>_unit')*

- Compare initial conditions of source blocks to neighbouring blocks

  Use *get_param(<blk>,'<variable>') for both blocks and compare*

- Check that property blocks are present, e.g. Gas Properties (G)

  Use *find_system(<mdl>,'classname',<classname>) or find_system(<mdl>,'masktype',<blkname>)*

Running the Model Using Initial States

The model Initialises. YAY!

What next?

- Don't re-initialise each time
- Save 'xout' states, so that model can be run quickly

Use *set_param(<mdl>,'SaveState','on')* to save states

Index into *xout.values to get the required starting value and return to form the states structure*

Use *set_param(<mdl>,'LoadInitialState','on')* to load starting state

Where is Our Dynamic Model Now

- Model initialises
- Now running scenarios, including:
  - Throttling up and down throughout the operational range
  - Start-up sequence
  - Shutdown sequence
- Reporting back to subsystems with time-series data from the model
- Model continues to develop, adding fidelity and testing new data

Lawrence Pryn

Oliver Hyde

Piotr Zulawski

Yashi Kuplish

Chris Haynes

Special thanks to:

MathWorks

Chris Lim

Rick Hyde

Thanks for Listening