Unified System Modeling and Simulation via Constraint Hypergraphs

John Morris, Gregory Mocko, John Wagner

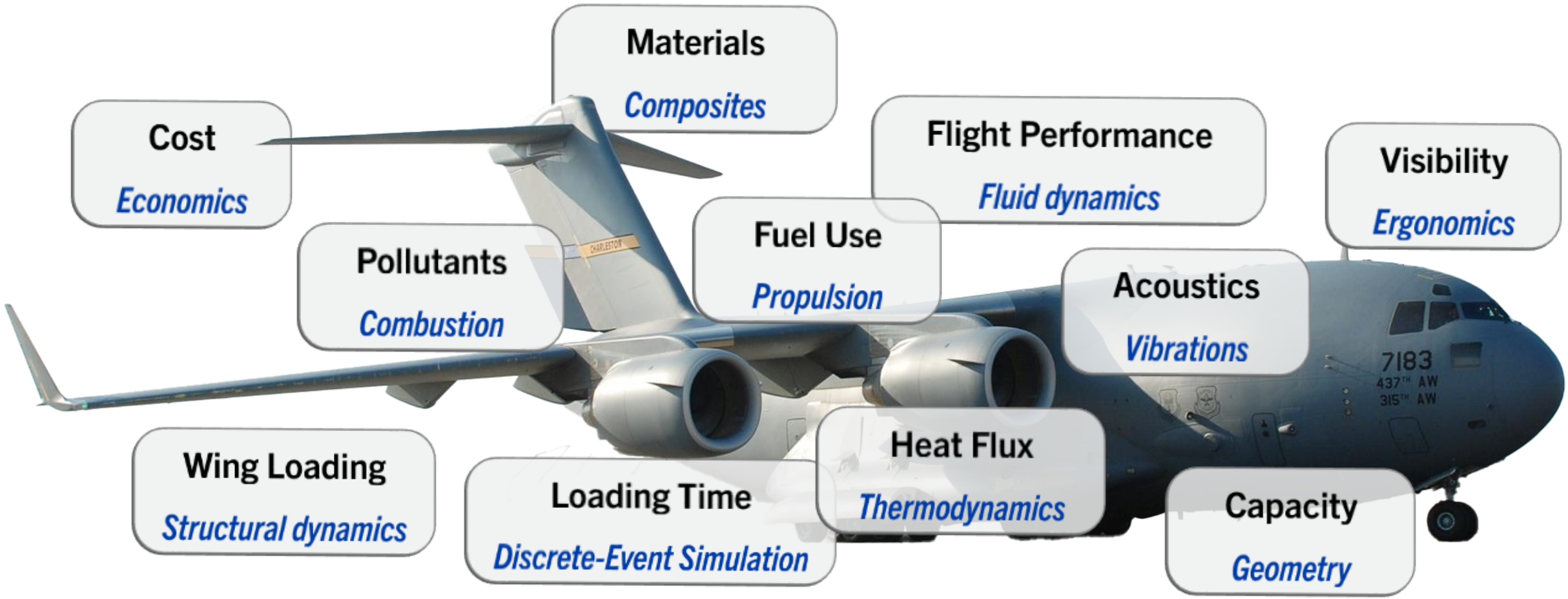
Assistant Professor | Industrial & Systems Engineering and Engineering Mgmt.
The University of Alabama in Huntsville
17 April 2026

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

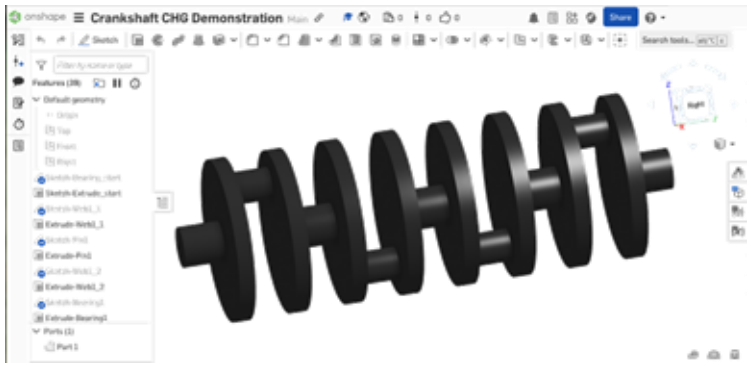
CLEMSON

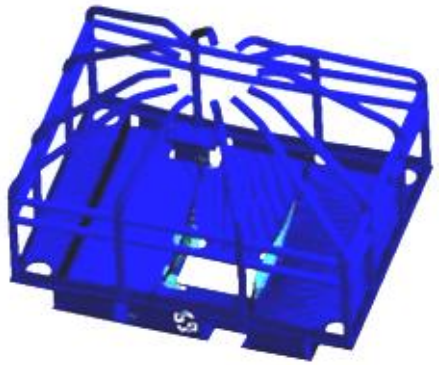
SEARCH
Systems Engineering Applied Research Lab

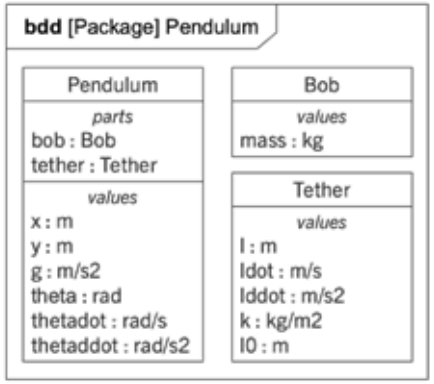
Knowing information about complex systems require models across many domains

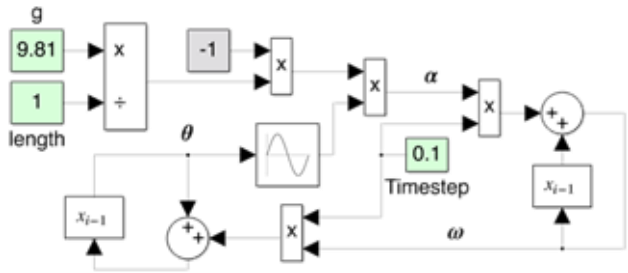


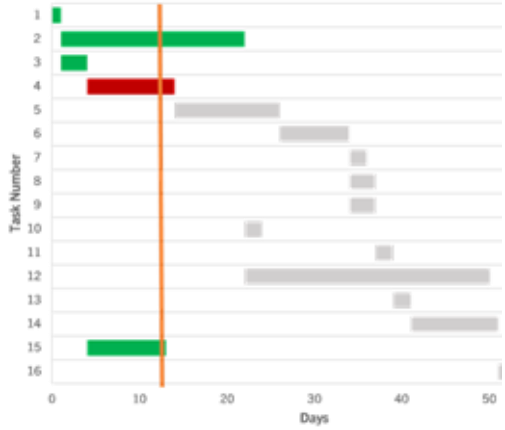
Simulating a model provides a limited view of a system

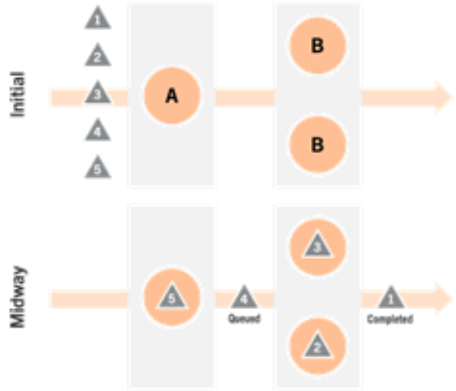


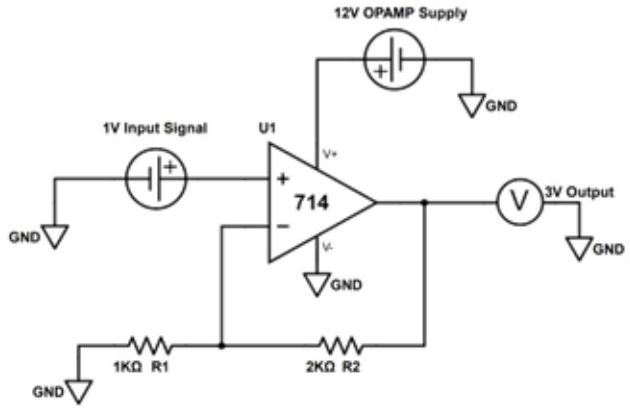


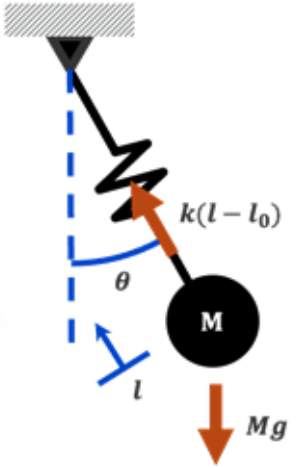






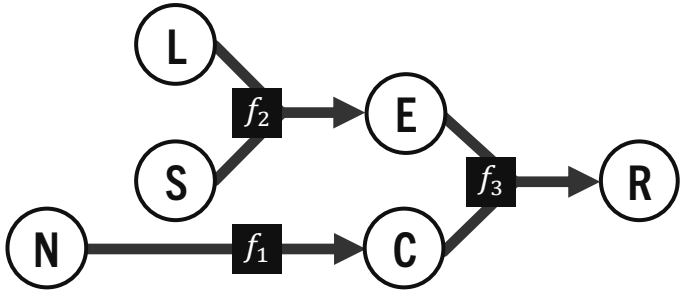


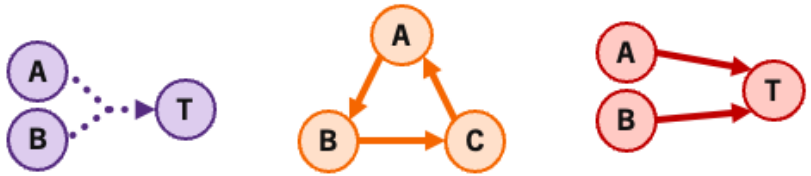




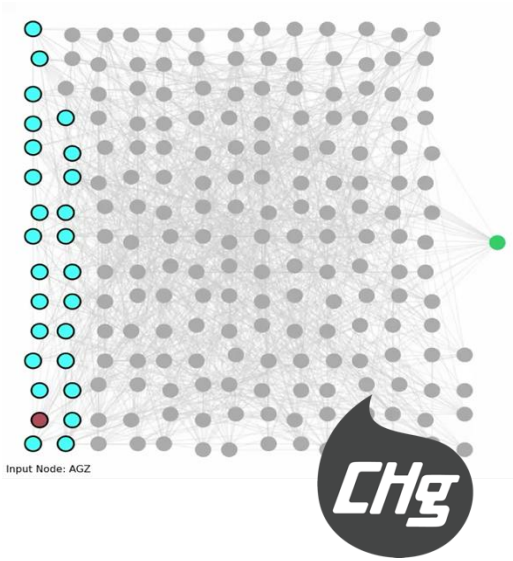
Constraint hypergraphs are a framework for universal, declarative systems modeling

CHG Definition

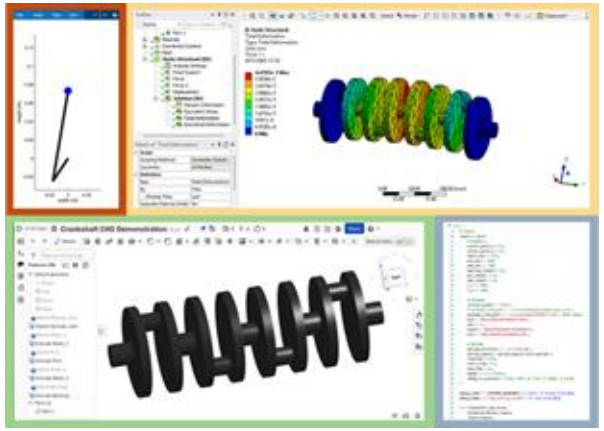


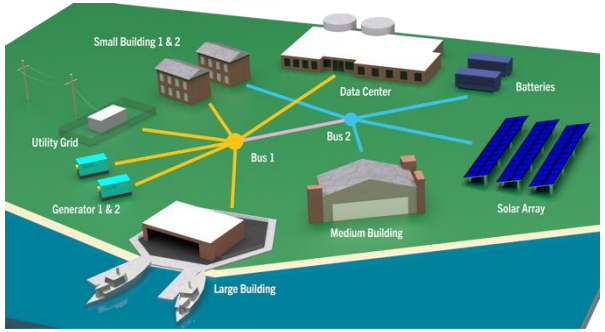


Declarative Simulation



Applications





Systems are described by either observations or simulations

Economy (L/km)
10
11
12

Color
Blue
Gray
Tan

Fuel Capacity (L)
100,000
115,000
130,000

Name
C-17 Globemaster
C-5 Galaxy
C-130 Hercules

Observations are measurements we make of reality

Economy (L/km)
10
11
12

Color
Blue
Gray
Tan

Fuel Capacity (L)
100,000
115,000
130,000

Name
C-17 Globemaster
C-5 Galaxy
C-130 Hercules

Simulations are predictions we make based on relationships

Economy (L/km)
10
11
12

Color
Blue
Gray
Tan

Fuel Capacity (L)
100,000
115,000
130,000

(Database lookup)

Name
C-17 Globemaster
C-5 Galaxy
C-130 Hercules

Model relations are algebraic functions that describe behavior

$f_{n \to e}(\text{Name}) \to \text{Economy}$

Name
C-17 Globemaster
C-5 Galaxy
C-130 Hercules

Economy (L/km)
10
11
12

$f_{n \to c}(\text{Name}) \to \text{Capacity}$

Fuel Capacity (L)
100,000
115,000
130,000

J. C. Willems, "The Behavioral Approach to Open and Interconnected Systems," IEEE Control Systems Magazine (2007)

Functions can have multiple arguments

$f_{n \to e}(\text{Name}) \to \text{Economy}$

$f_{n \to c}(\text{Name}) \to \text{Capacity}$

$f_{ce \to r}(\text{Capacity} \times \text{Economy}) \to \text{Range}$

Name
C-17 Globemaster
C-5 Galaxy
C-130 Hercules

Economy (L/km)
10
11
12

Fuel Capacity (L)
100,000
115,000
130,000

Flight Range (km)
10,000
11,000
12,000

Functions compose to form greater functions

Name
C-17 Globemaster
C-5 Galaxy
C-130 Hercules

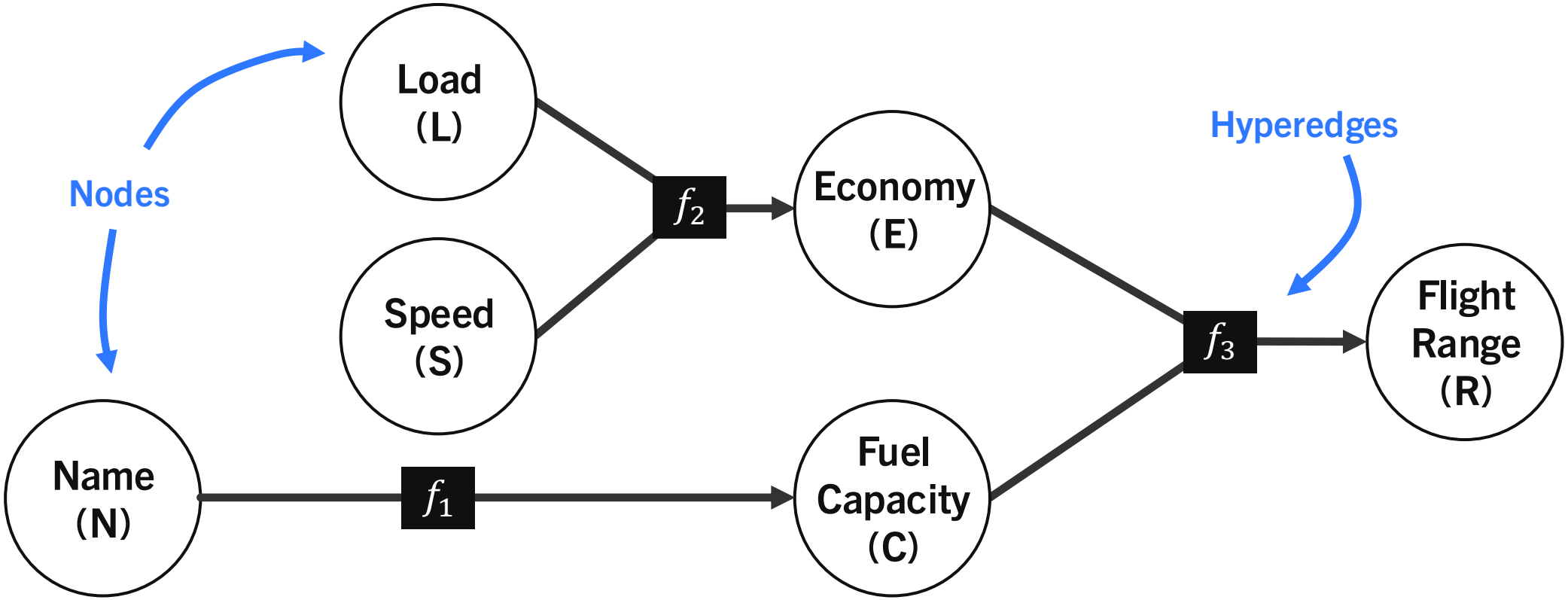
$$f_{n\to r}(\text{Name}) \to \text{Range} = f_{ce\to r}(f_{n\to e}, f_{n\to c})$$

Economy (L/km)
10
11
12

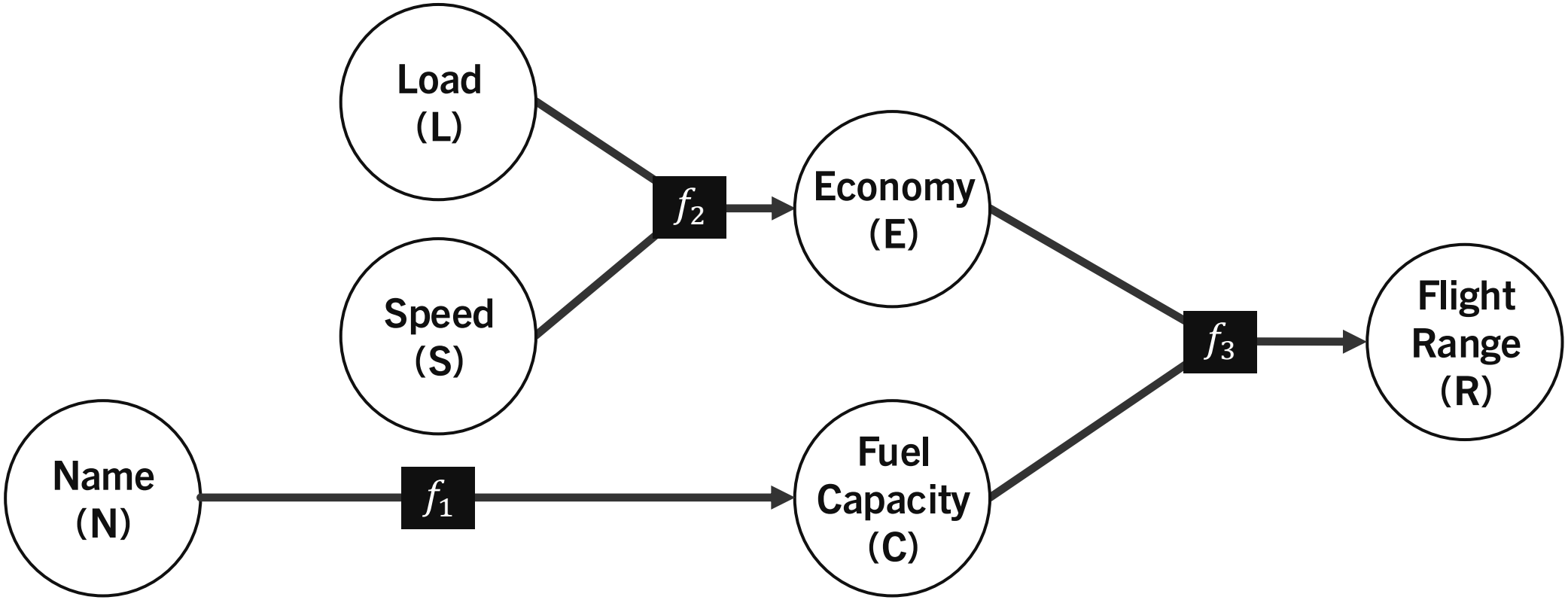
Fuel Capacity (L)
100,000
115,000
130,000

Flight Range (km)
10,000
11,000
12,000

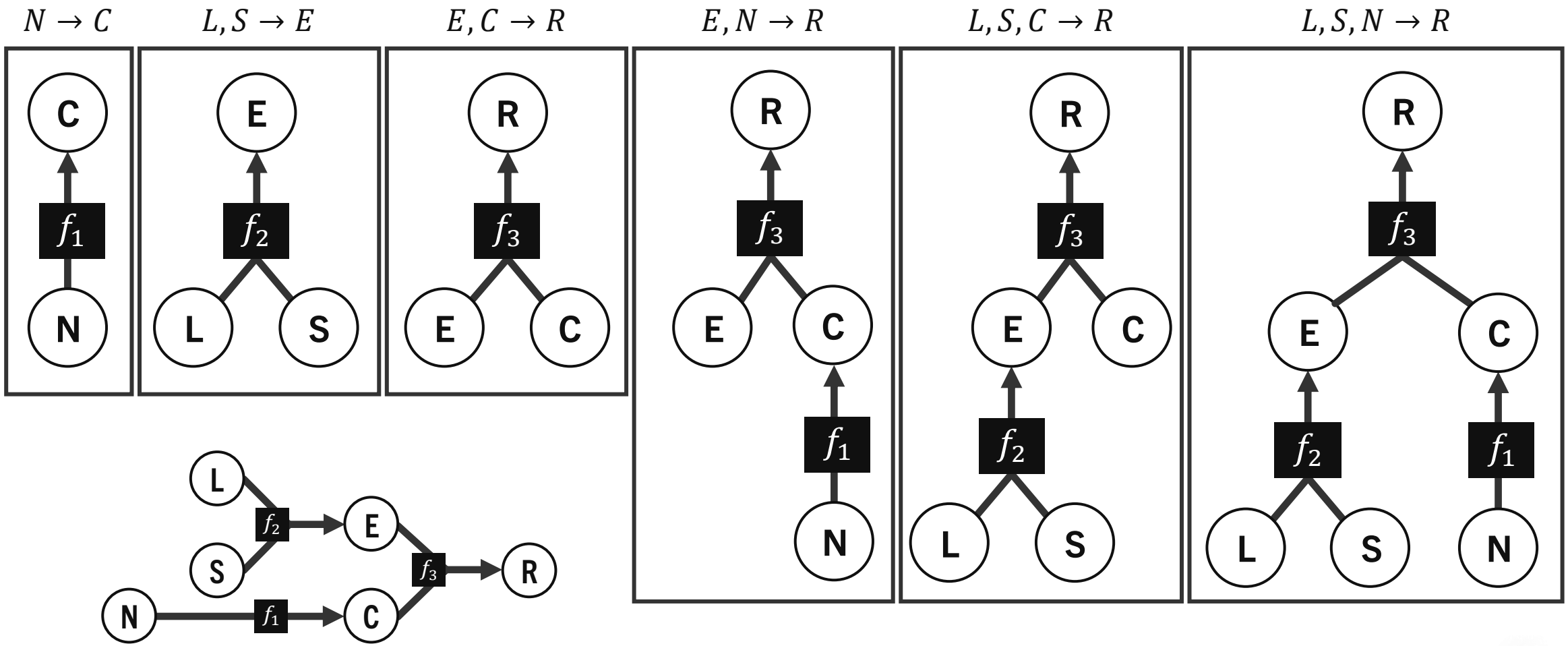
We can represent all models and variables as a constraint hypergraph (CHG)



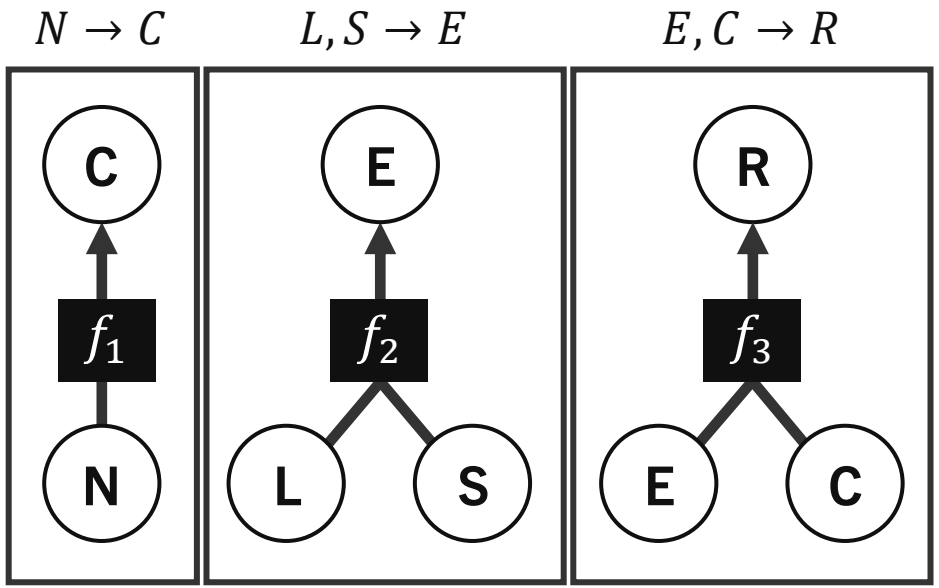
Every path in a CHG is a simulation



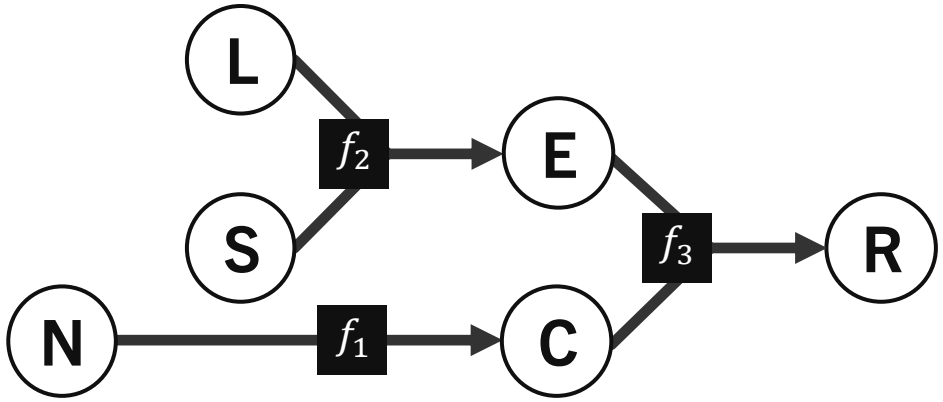
Every path in a CHG is a simulation



Declarative simulation in a CHGs is pathfinding

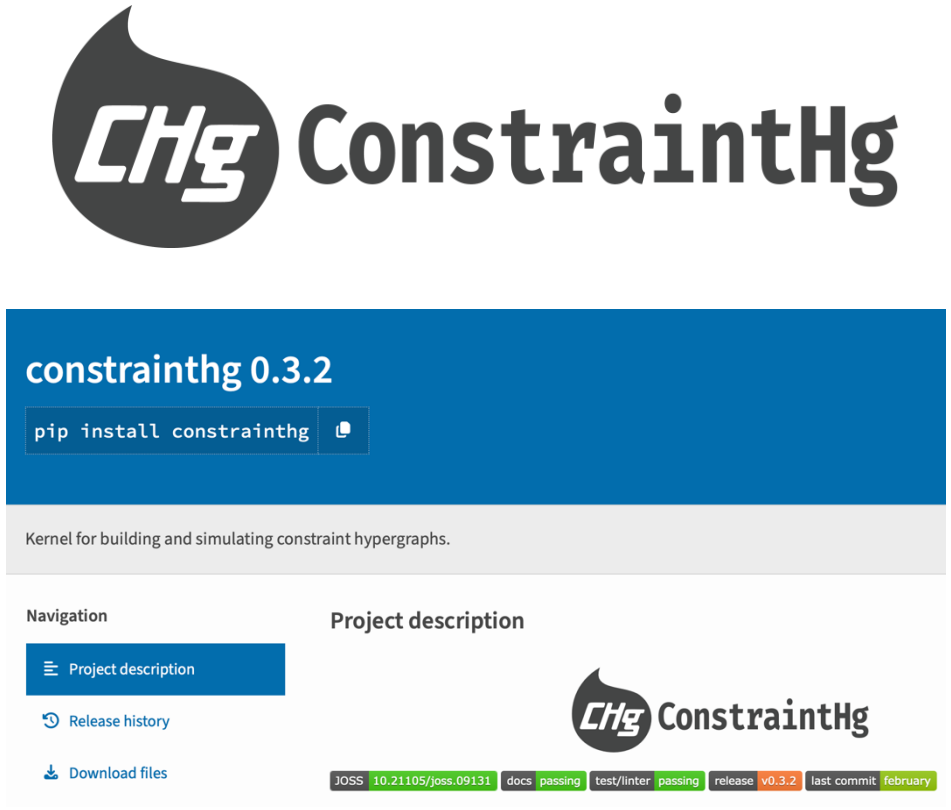


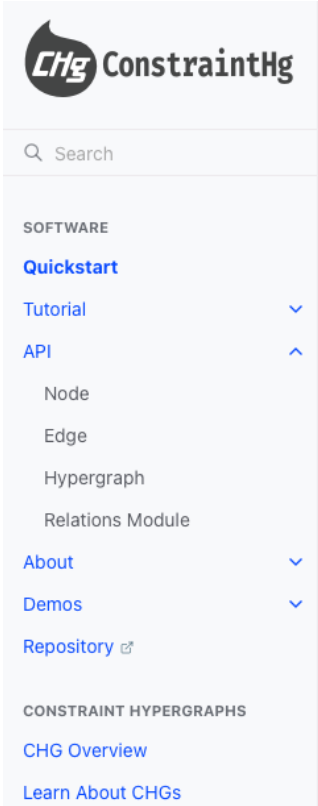
Imperative



Declarative

Declarative simulation provided by ConstraintHg





Quickstart

Use PIP to install ConstraintHg into your Python environment:

```
pip install constrainthg
```

From there you'll want to import the library into your Python script. This is a pretty typical method to use:

```
from constrainthg.hypergraph import Node, Hypergraph
import constrainthg.relations as R
```

Simple Demo

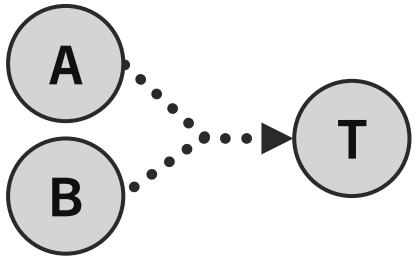
Note that this demo is found in demos/demo_basic.py

Let's build a basic constraint hypergraph of the following equations:

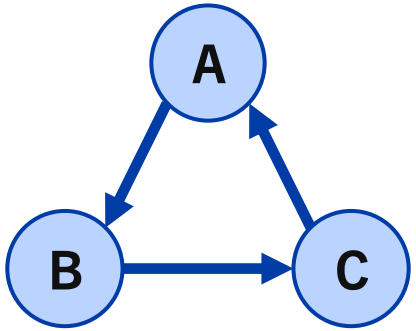
- $A + B = C$
- $A = -D$
- $B = -E$
- $D + E = F$
- $F = -C$

latest

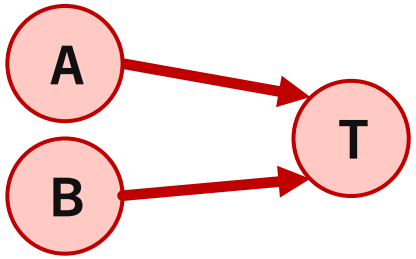
There are three structures in a CHG that a declarative agent needs to process:



Partial Edges



Cycles

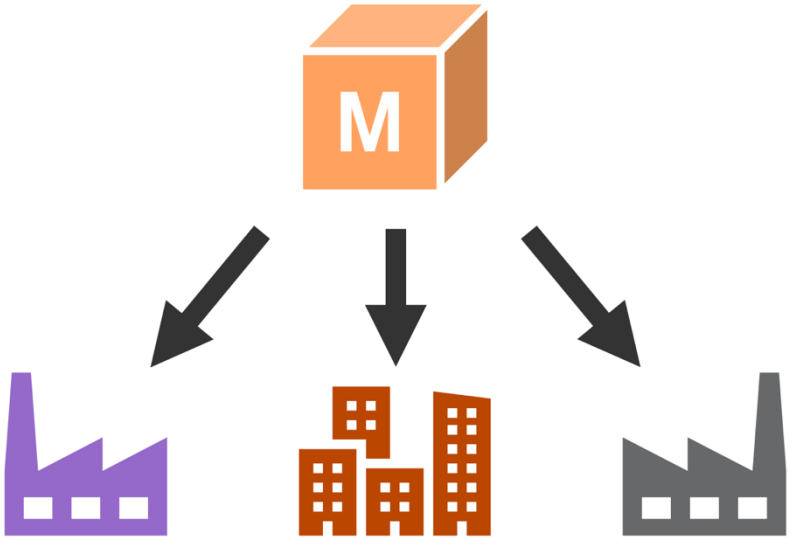


Multi-Edges

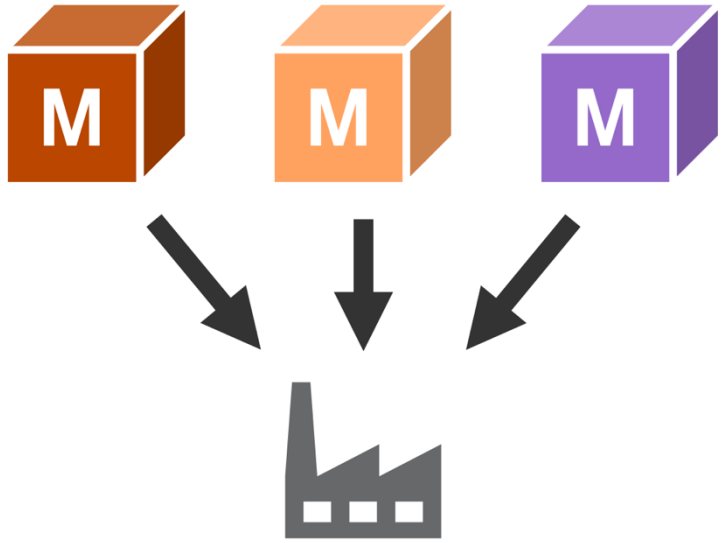
Morris, J., Mocko, G., & Wagner, J. (2025). Unified System Modeling and Simulation via Constraint Hypergraphs. *J. Comput. Inf. Sci. Eng.*

CHGs provide platforms for model-based engineering and digital twins

Maintains information in different contexts

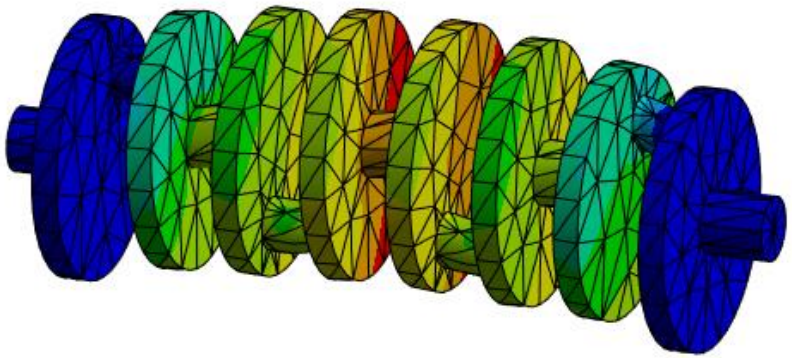


Maintains information with different models

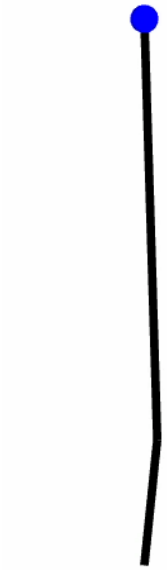


Simulation paths can cross between calculating software

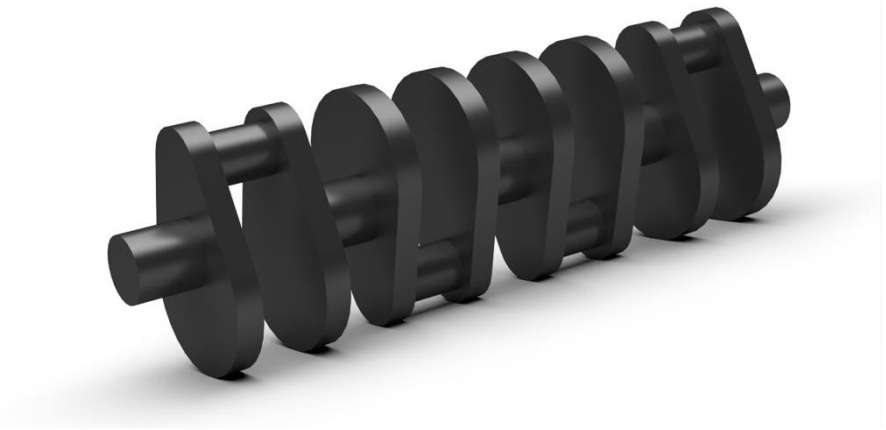
Material Mechanics



Kinematic Analysis



Mass Properties

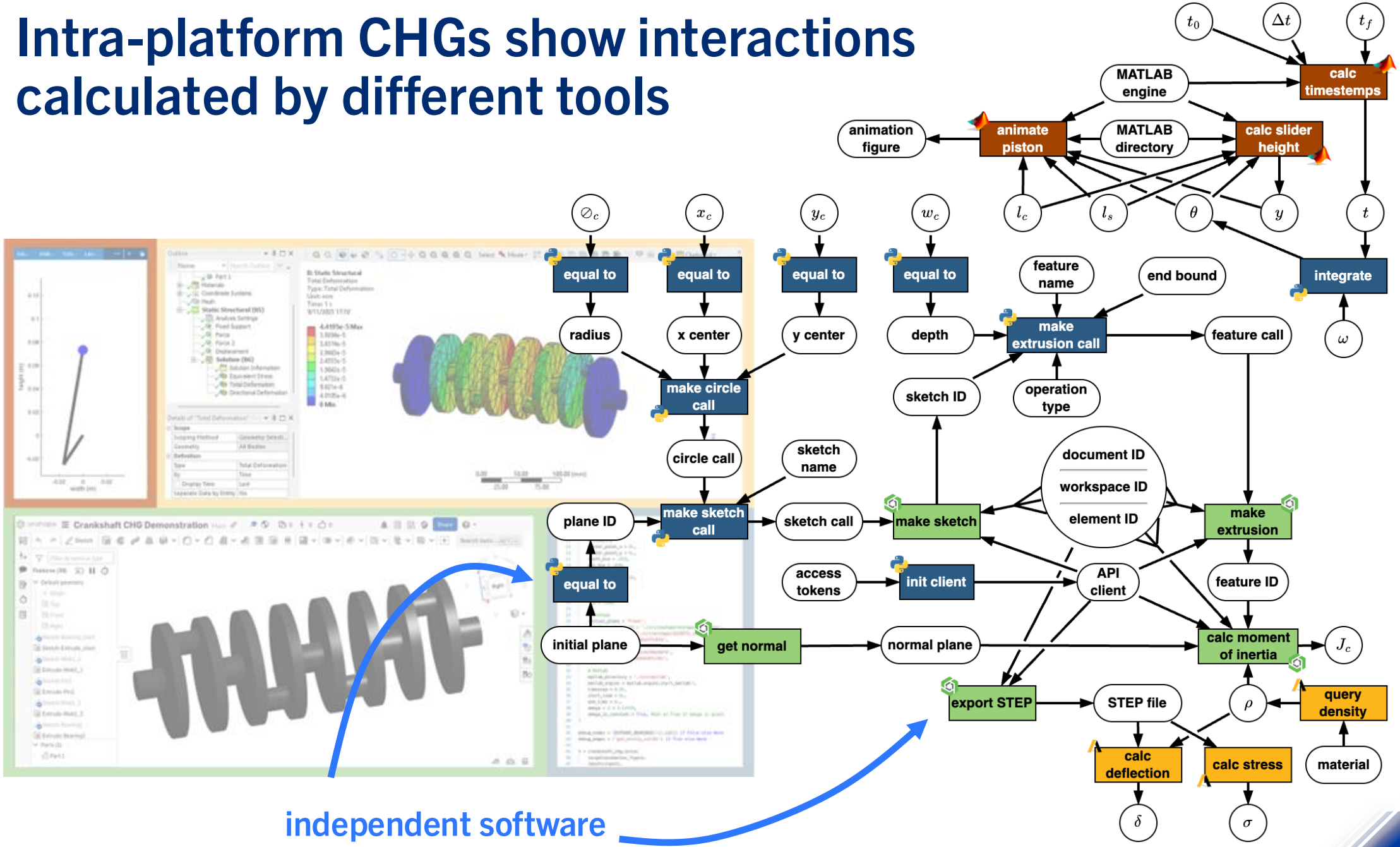


Ansys

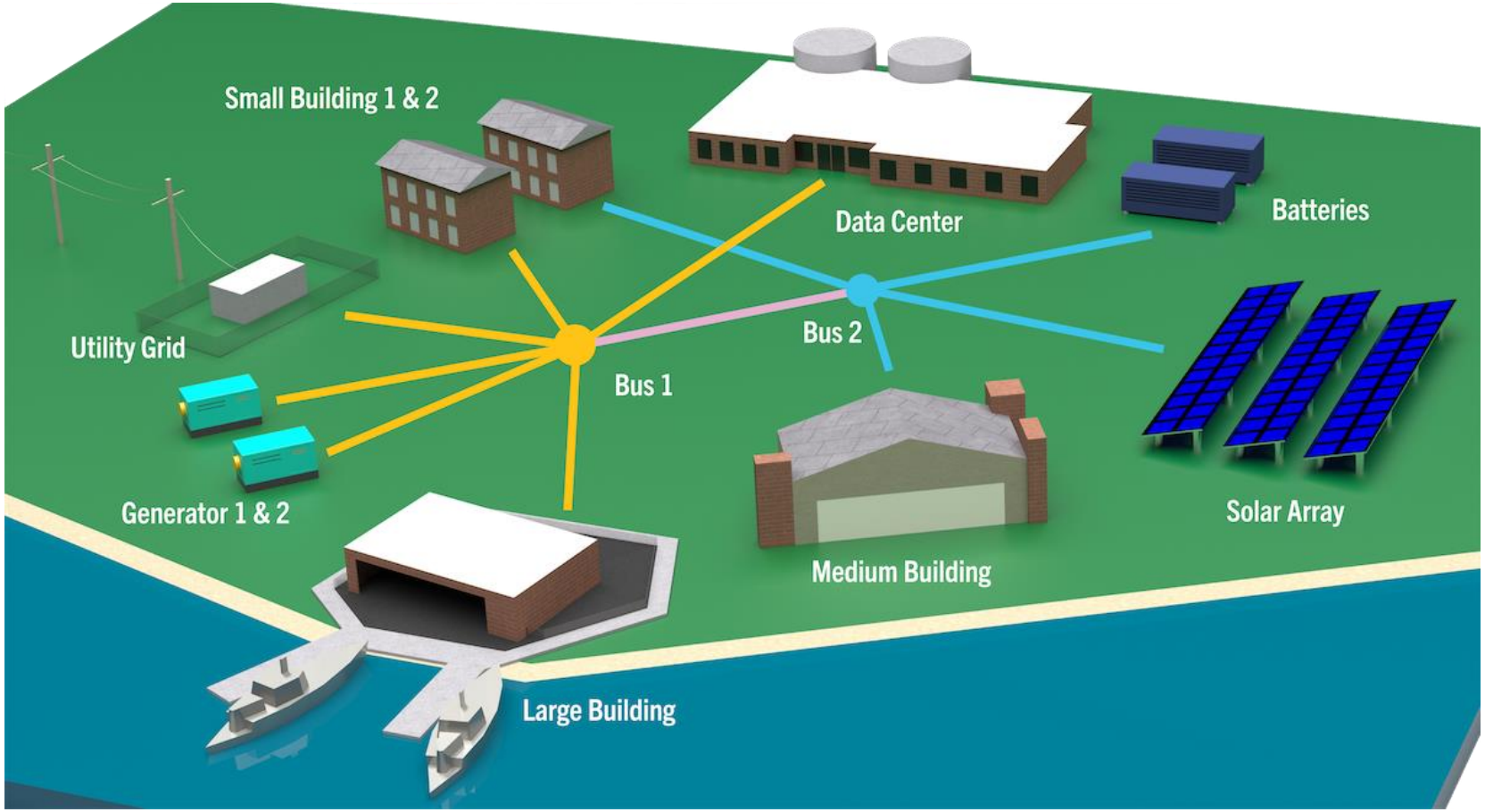
MATLAB® & SIMULINK®

onshape®

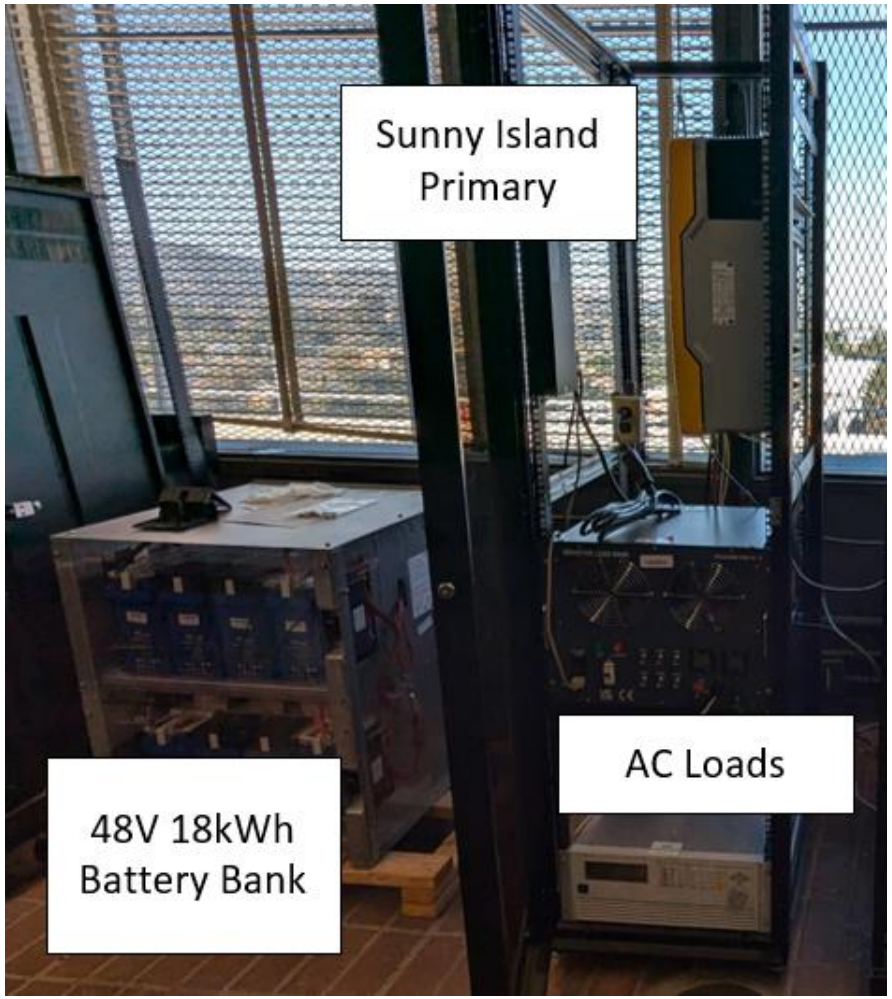
Intra-platform CHGs show interactions calculated by different tools

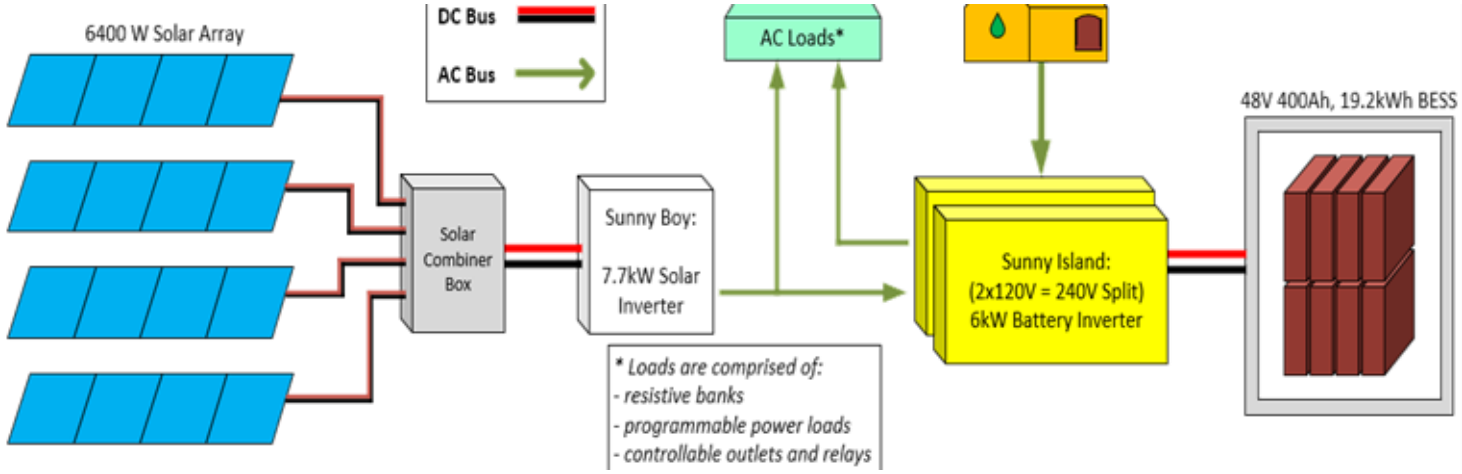


Demonstration: Microgrid Digital Twin



Physical Microgrid at Naval Postgraduate School



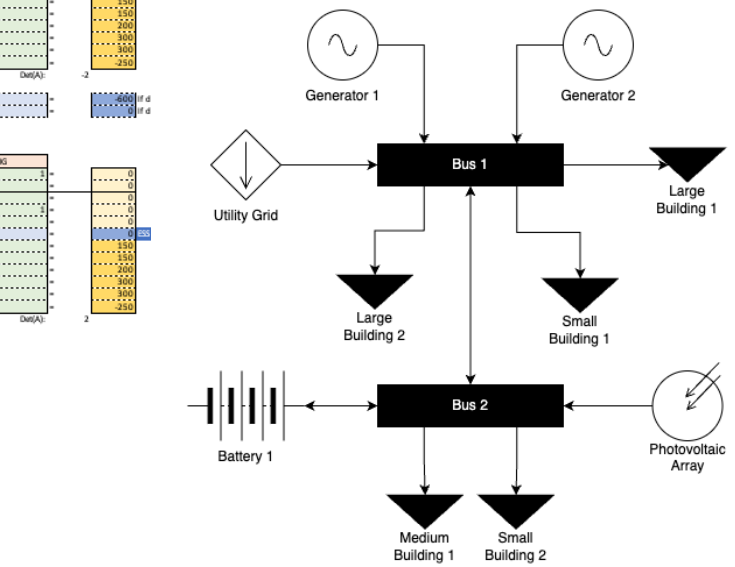


Previous Imperative Modeling of Microgrid

MG_LP_Test_v10.m

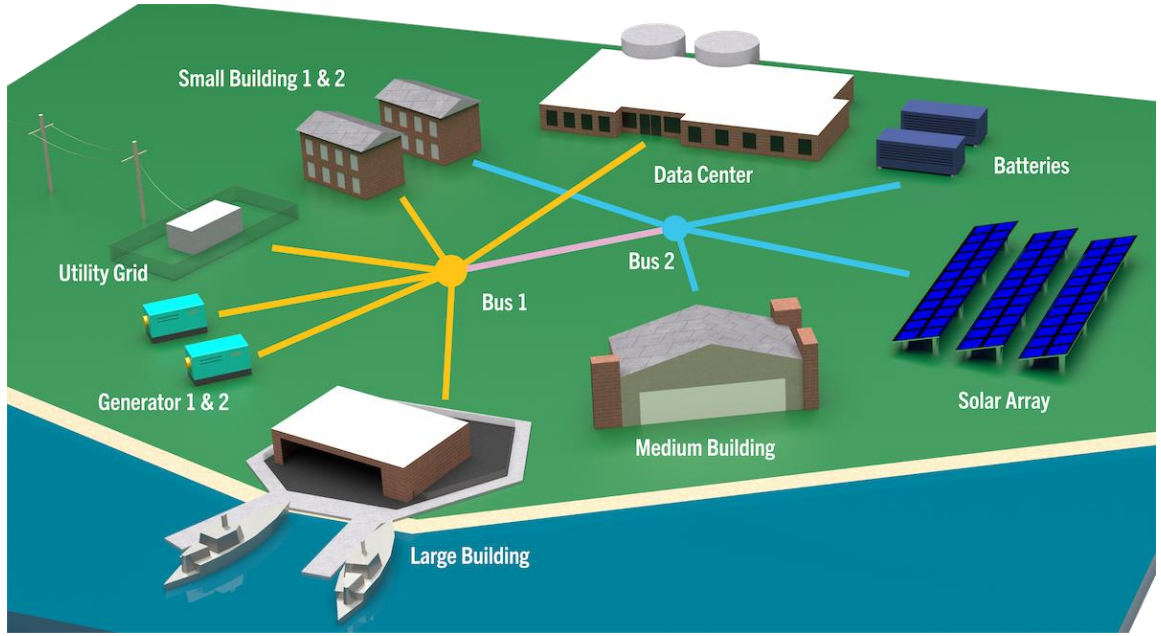
/Users/john_morris/Documents/Clemson/Research/MicrogridHg/archived_model/MG_LP_Test_v

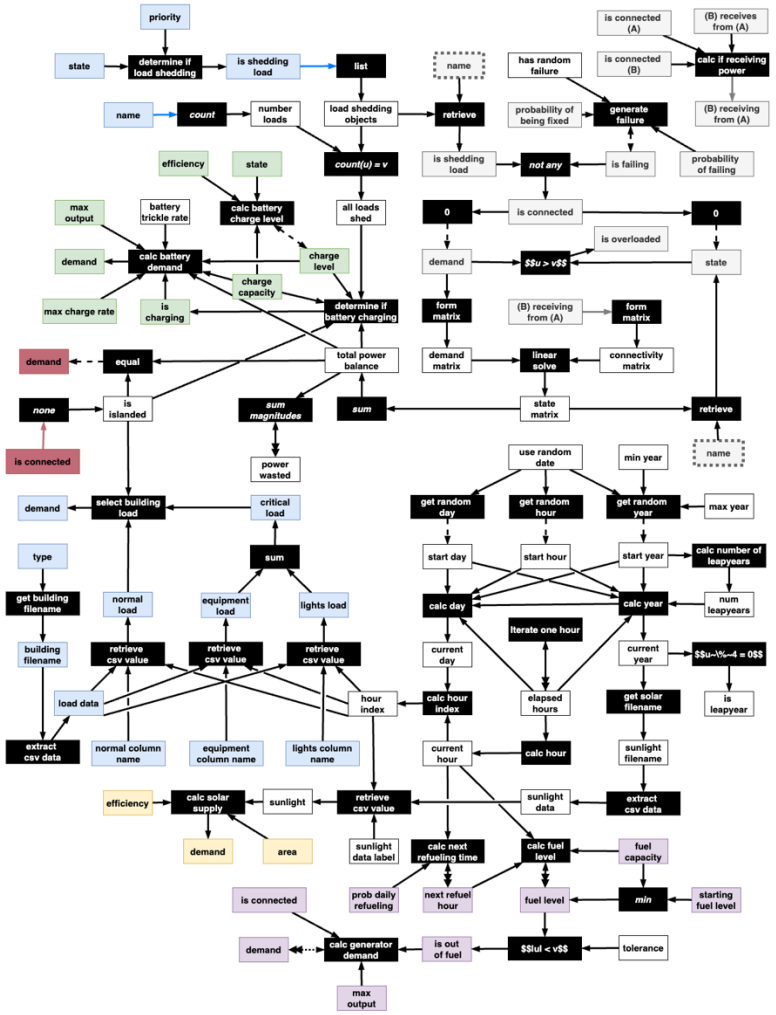
```
423         %Initial Solve
424         x(:,n) = linsolve(A2,b);
425
426         %Determine if generator demand exceeds generat
427         Gen_Demand = -x(pos.Gens,n);
428
429         if any(Gen_Demand > [Gens.Capacity]')
430             %If the B2 to B1 Buss line or BT1 line is
431             %exhausted cannot utilize ESS to make up g
432             if MG_State.B2_B1(n) == false || MG_State.
433                     || BT1_Charge(n) < 0
434                 % Below line was useful for debugging
435                 % disp("Gens Overloaded at time step "
436                 Overload.Gens = true;
437             else
438                 %Otherwise set Generators at full outp
439                 %make up unmet demand
440                 A2(6,:) = pos.Gens & MG_State{n,:};
441                 b(6) = -MG_State{n,{'B1_GEN1' 'B1_GEN2
442                 x(:,n) = linsolve(A2,b);
443             end
444         %If Generator demand is negative, then generat
445         %charge batteries with excess PV generation
446         elseif any(Gen_Demand < 0)
447             A2(6,:) = pos.Gens;
448             b(6) = 0;
449             x(:,n) = linsolve(A2,b);
450         end
451
452         % Check for Battery Output Exceeded
453         % Add 0.1 Due to Rounding Errors in Linear Solver
454         if -x(contains(LoadVars,'BT1'),n) - 0.01 > BT1.Output * (BT1_Charge(n) > 0)
455             Overload.BT1 = true;
456             % Below line was useful for debugging in single runs. Commented out
```



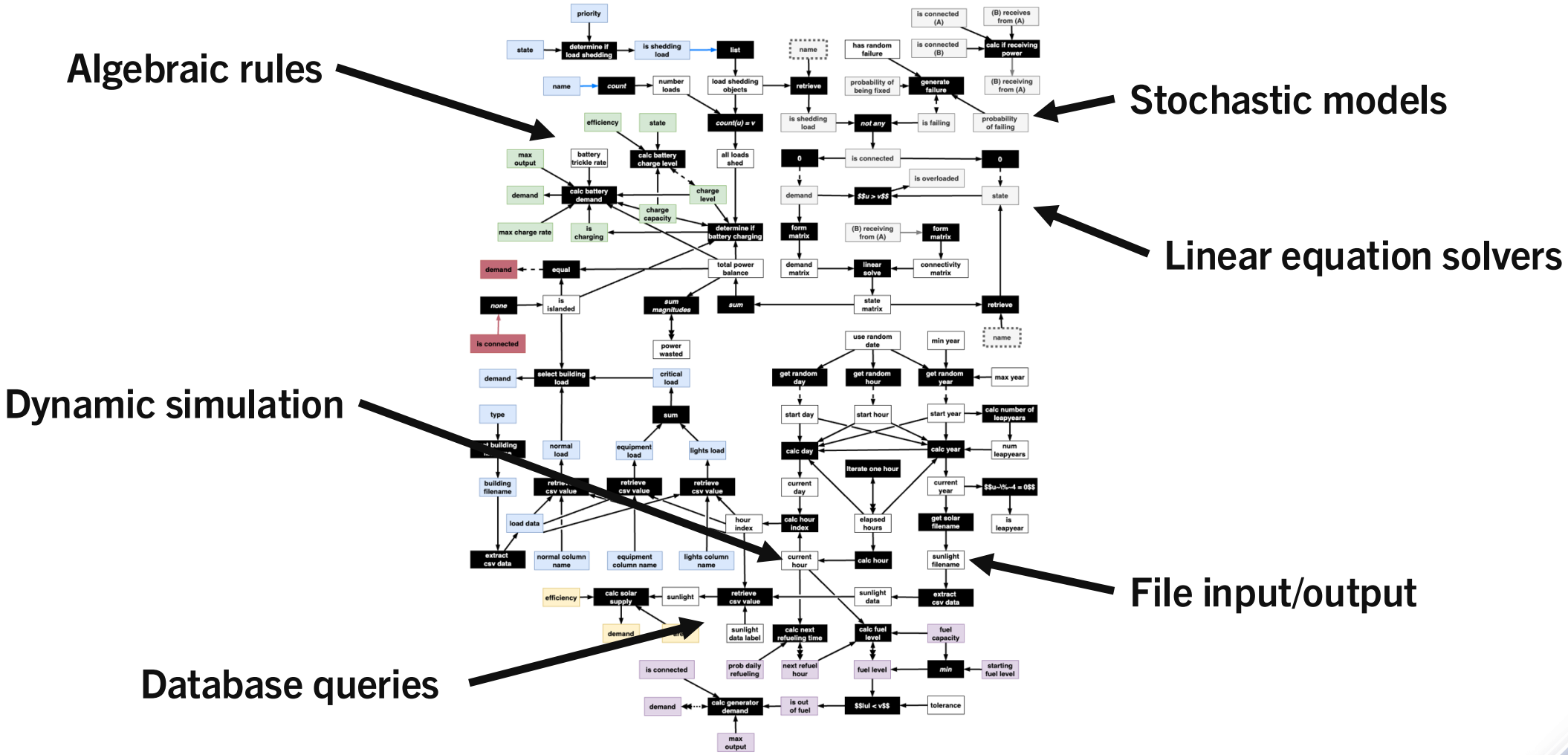
C. J. Peterson, "Systems Architecture Design and Validation Methods for Microgrid Systems," Master's thesis, Naval Postgraduate School, Monterey, CA, 2019

Transformed CHG contains 594 nodes and 334 edges

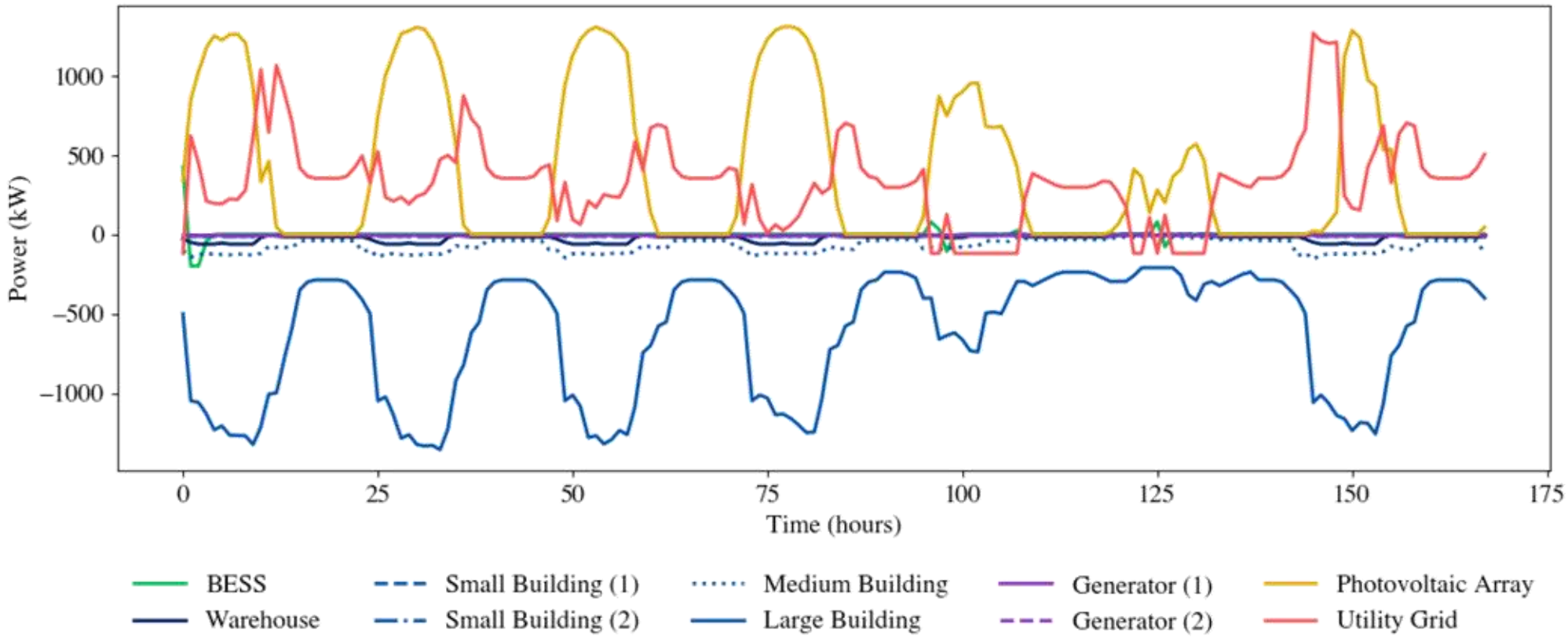




Includes all kinds of simulation and modeling

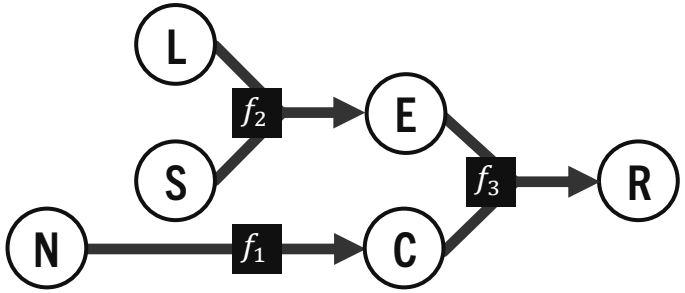


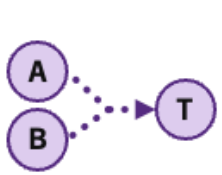
Autonomously observes and simulates the real system

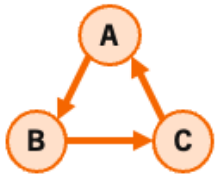


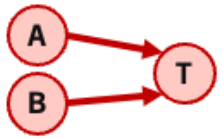
CHGs provide a universal language for systems modeling and simulation

Universal Modeling









Declarative Simulation

More information on CHGs + JCISE paper

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

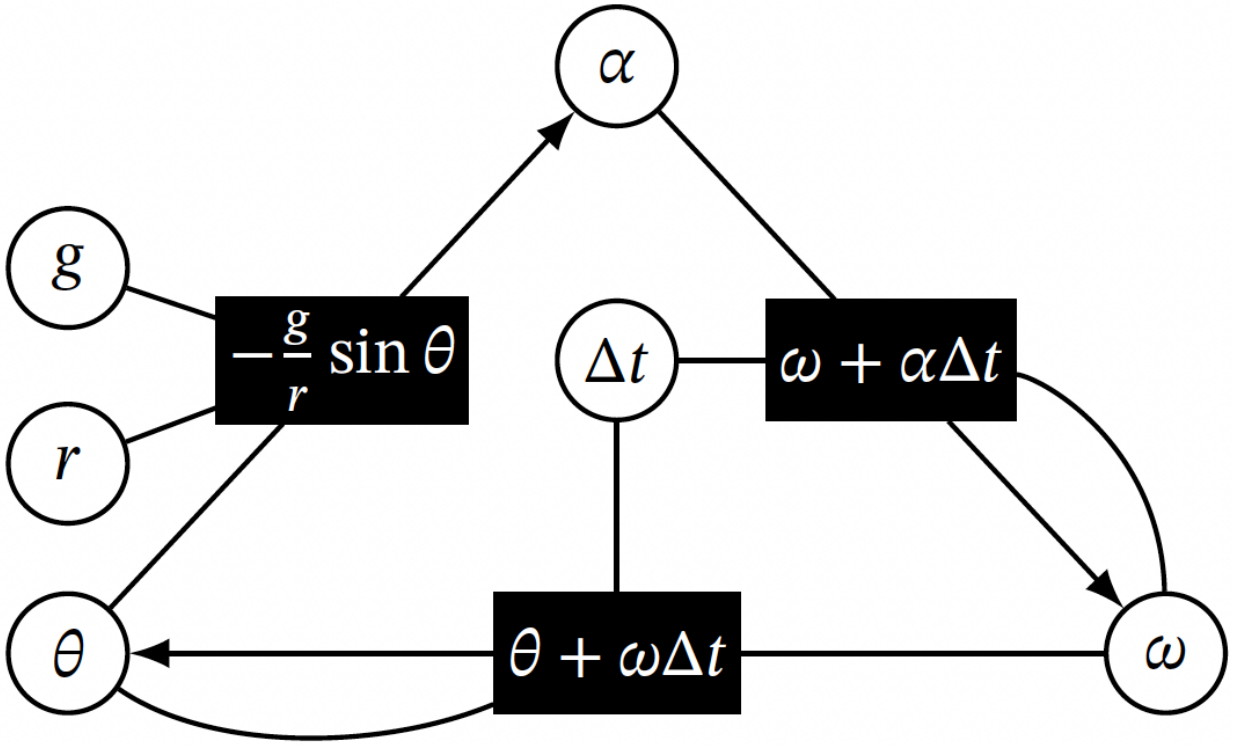
John Morris
john.h.morris@uah.edu

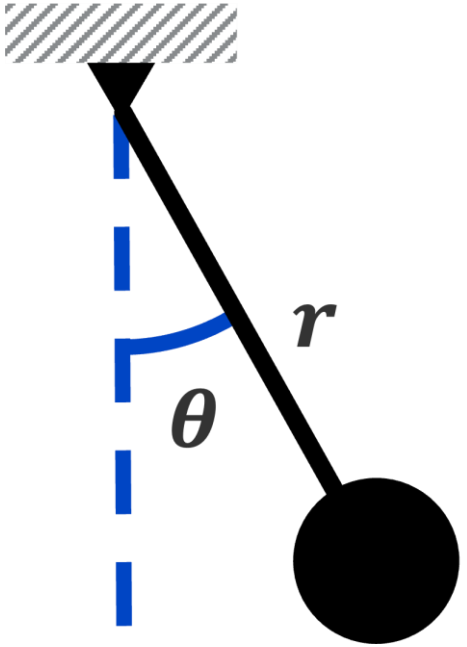
SEARCH
Systems Engineering Applied Research Lab

Additional Slides

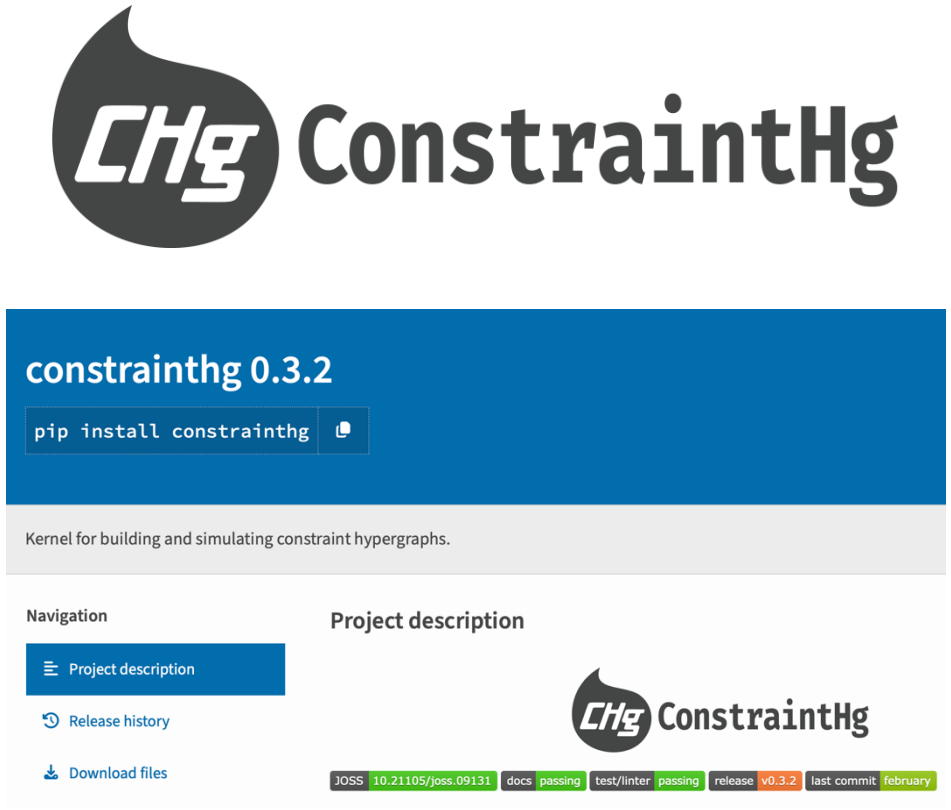
Universal System Simulation via Constraint Hypergraphs

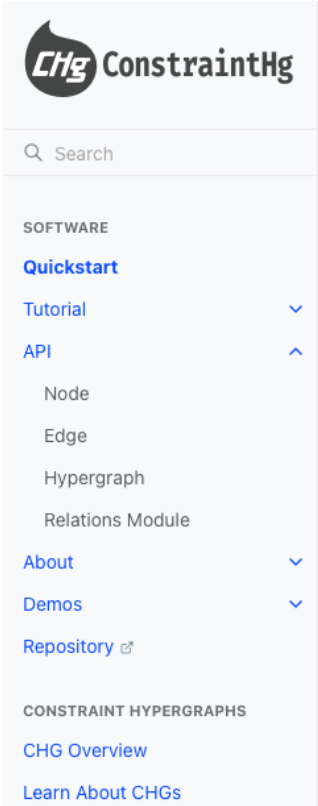
Pendulum System





Declarative simulation provided by ConstraintHg





Quickstart

Use PIP to install ConstraintHg into your Python environment:

```
pip install constrainthg
```

From there you'll want to import the library into your Python script. This is a pretty typical method to use:

```
from constrainthg.hypergraph import Node, Hypergraph
import constrainthg.relations as R
```

Simple Demo

Note that this demo is found in demos/demo_basic.py

Let's build a basic constraint hypergraph of the following equations:

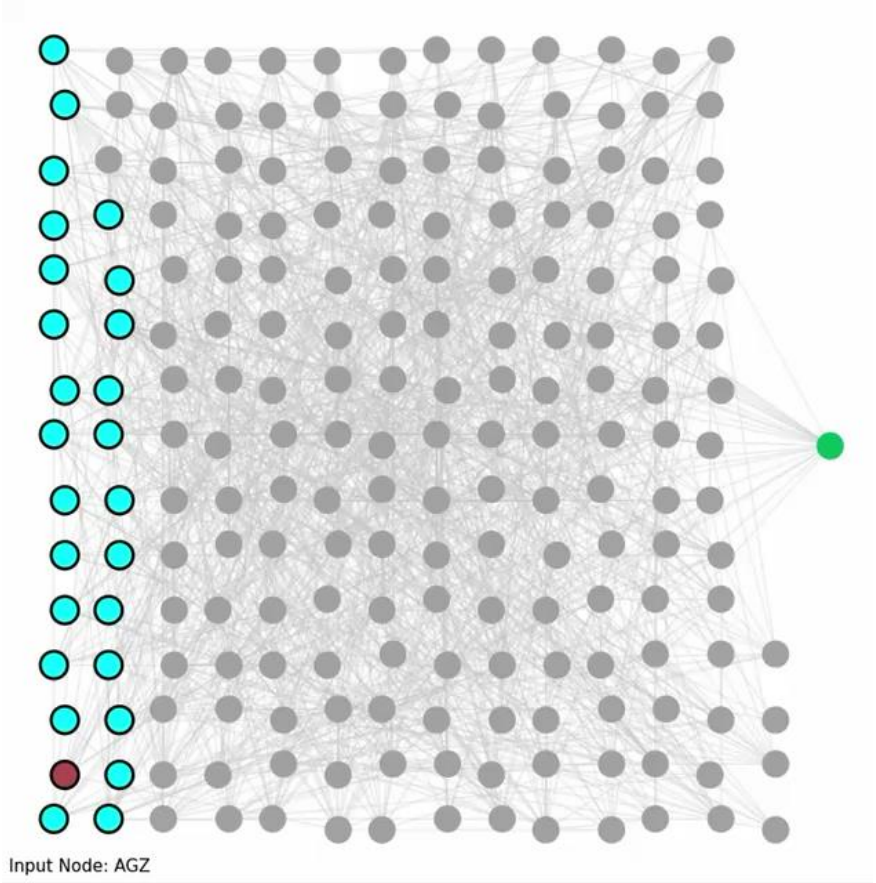
- $A + B = C$
- $A = -D$
- $B = -E$
- $D + E = F$
- $F = -C$

latest

ConstraintHg Functionality

Provides methods for:

- Representing and visualizing CHGs
- Forming cycles, partial edges, and edge weights
- Pathfinding for declarative simulation
  - Extensive logging tools
- Merging CHGs



Morris, J., (2026). “ConstraintHg: A Kernel for Systems Modeling and Simulation,” Journal of Open-Source Software.

Process of making a CHG

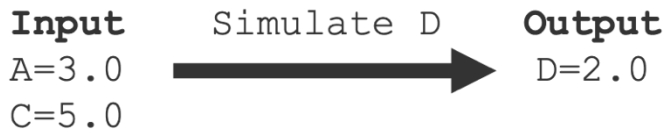
1. Identify system facts (nodes)
Parameters, application variables, API tokens, etc.

2. Form relations (edges) between nodes
Relations show how one node is determined by a set of other nodes

3. Pass to CHG solver
Solver parses the CHG

ConstraintHg

4. Request simulation
Solver simulates requested output by finding the shortest path mapping it to a set a known inputs



Morris, J., Mocko, G., Wagner, J., & Ramnath, S. (2025). Declarative Integration of CAD Software into Multi-Physics Simulation via Constraint Hypergraphs. ASME IDETC-CIE 2025 conference

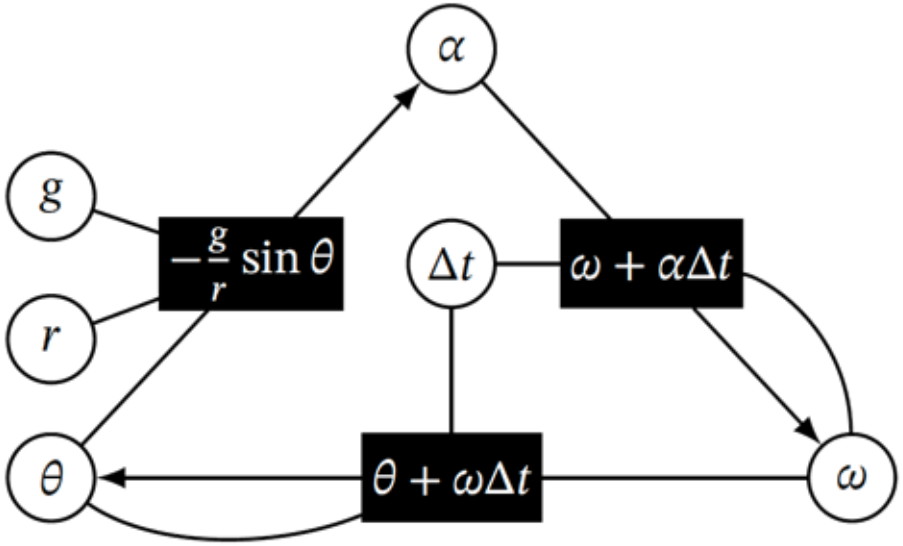
Declarative simulation requires only start & end points

```
hg.solve(
    target='theta',
    inputs=(
        theta=0.785,
        omega=0.0,
        g=9.81,
        r=0.25,
        delta_t=0.02
    ),
    min_index=2,
)
```

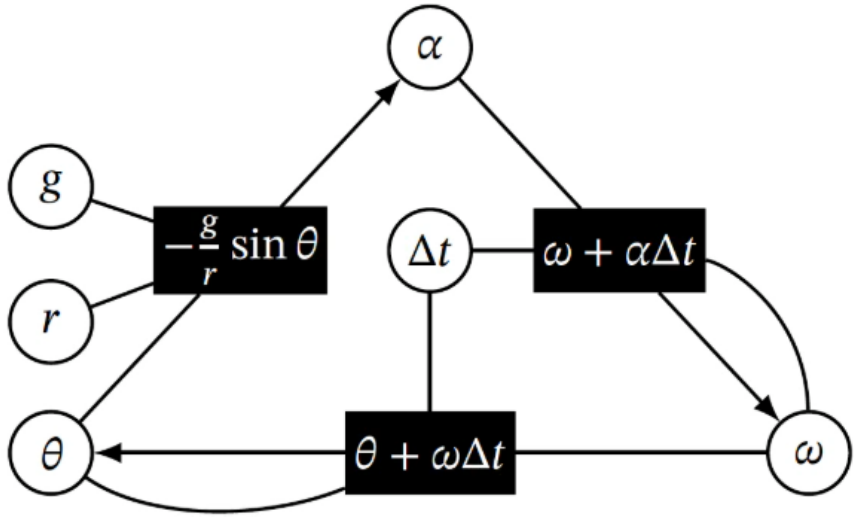
Simulation call

```
└──theta(2)=0.7739
    ├──omega(2)=-0.5547
    │   ├──alpha(2)=-27.74
    │   │   ├──g=9.81
    │   │   ├──theta=0.785
    │   │   └──r=0.25
    │   ├──delta_t=0.02
    │   └──omega=0
    ├──delta_t=0.02
    └──theta=0.785
```

Discovered, unraveled simulation



```python
"""
Demonstration of ConstraintHg for a simple pendulum.
More examples and information available online at constrainthg.readthedocs.io
"""

Run Cell | Run Below | Debug Cell
# %%
# Setup constraint hypergraph
from constrainthg import Hypergraph
from math import sin

# Define relational rules as methods in Python
def Rtheta_to_alpha(theta, g, r):
    return -g / r * sin(theta)

def Rintegrate(initial, slope, step):
    return initial + slope * step

def Rcheck_increment(initial, slope):
    return slope == initial + 1

# Initialize constraint hypergraph and add edges
hg = Hypergraph()

hg.add_edge(
    sources=dict(theta='theta', g='g', r='r'), target='alpha',
    rel=Rtheta_to_alpha, index_offset=1,
    label='-g/r*sin(theta)'
)

hg.add_edge(
    sources=dict(initial='omega', slope='alpha', step='del_t'), target='omega',
    rel=Rintegrate, index_via=Rcheck_increment, disposable=['initial', 'slope'],
    label='Integrate alpha'
)
```



Interactive-1

Interrupt | Clear All | Restart | Jupyter Variables | Save | ··· | .venv (