Large-Scale Test Bed in April 2024

Towards a Full Timescale Digital Twin

Jinning Wang[1], Fangxing Li[1], Kevin Tomsovic[1]

1. University of Tennessee, Knoxville

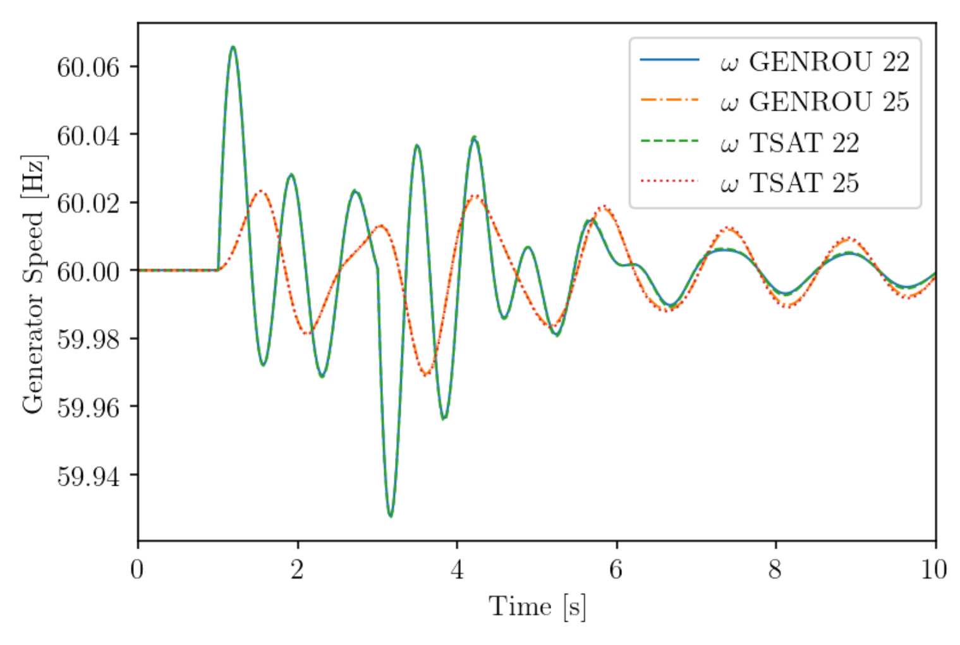
Motivations

❖ Enable dispatch-dynamic co-simulation via interoperable simulation
❖ Facilitate full timescale digital twin via dispatch-centric virtual power grid

Recent Milestones

❖ AMS development as dispatch simulator
❖ AGVis web application for online use

Products



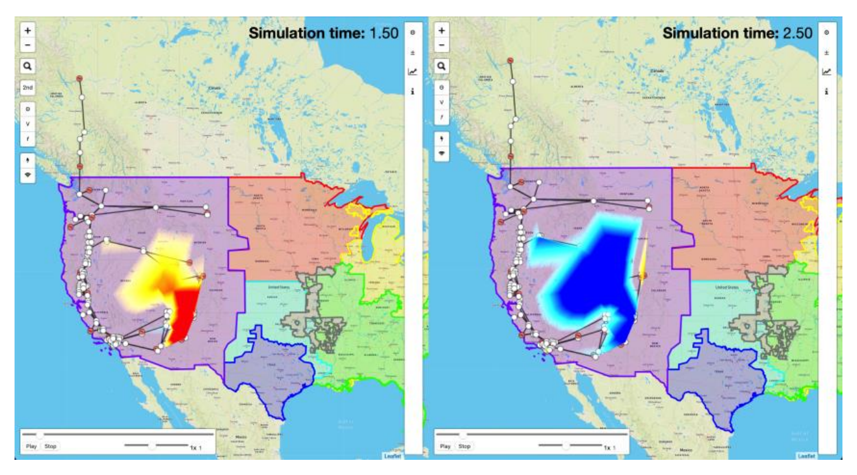
Cost [$]	AMS	MATPOWER	pandapower
PEGASE 1354-Bus	1,173,590.63	1,173,590.63	1,173,590.63
PEGASE 2869-Bus	2,338,915.61	2,338,915.61	2,338,915.61
GOC 4020-Bus	793,634.11	793,634.11	793,634.11
EPIGRIDS 5658-Bus	1,195,466.12	1,195,466.12	1,195,466.12
EPIGRIDS 7336-Bus	1,855,870.94	1,855,870.94	1,855,870.94

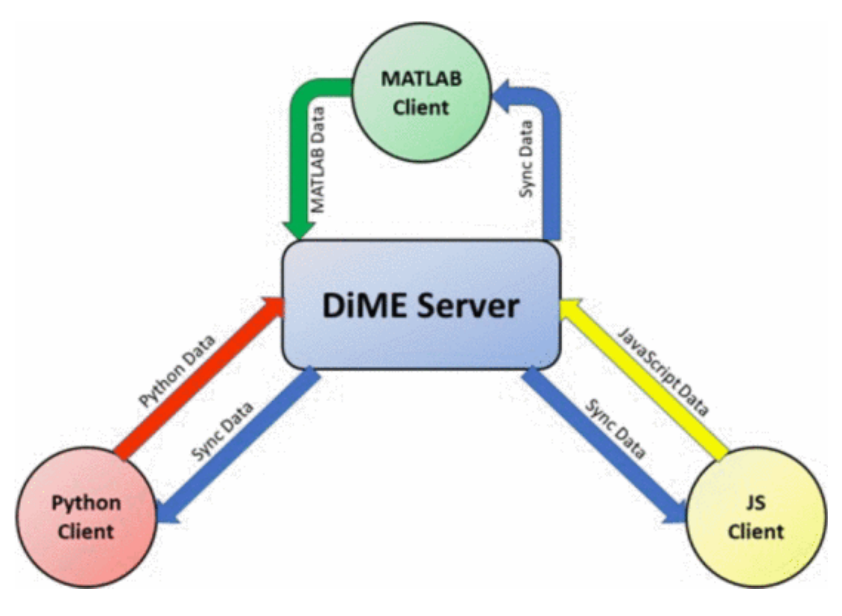
ANDES
Dynamic Modeling and Simulation

AGVis
Energy System Visualization

AMS
Dispatch Modeling and Simulation

DiME
Multi-terminal Data Streaming





Features

Facilitate rapid prototyping for **research**

Streamline Modeling

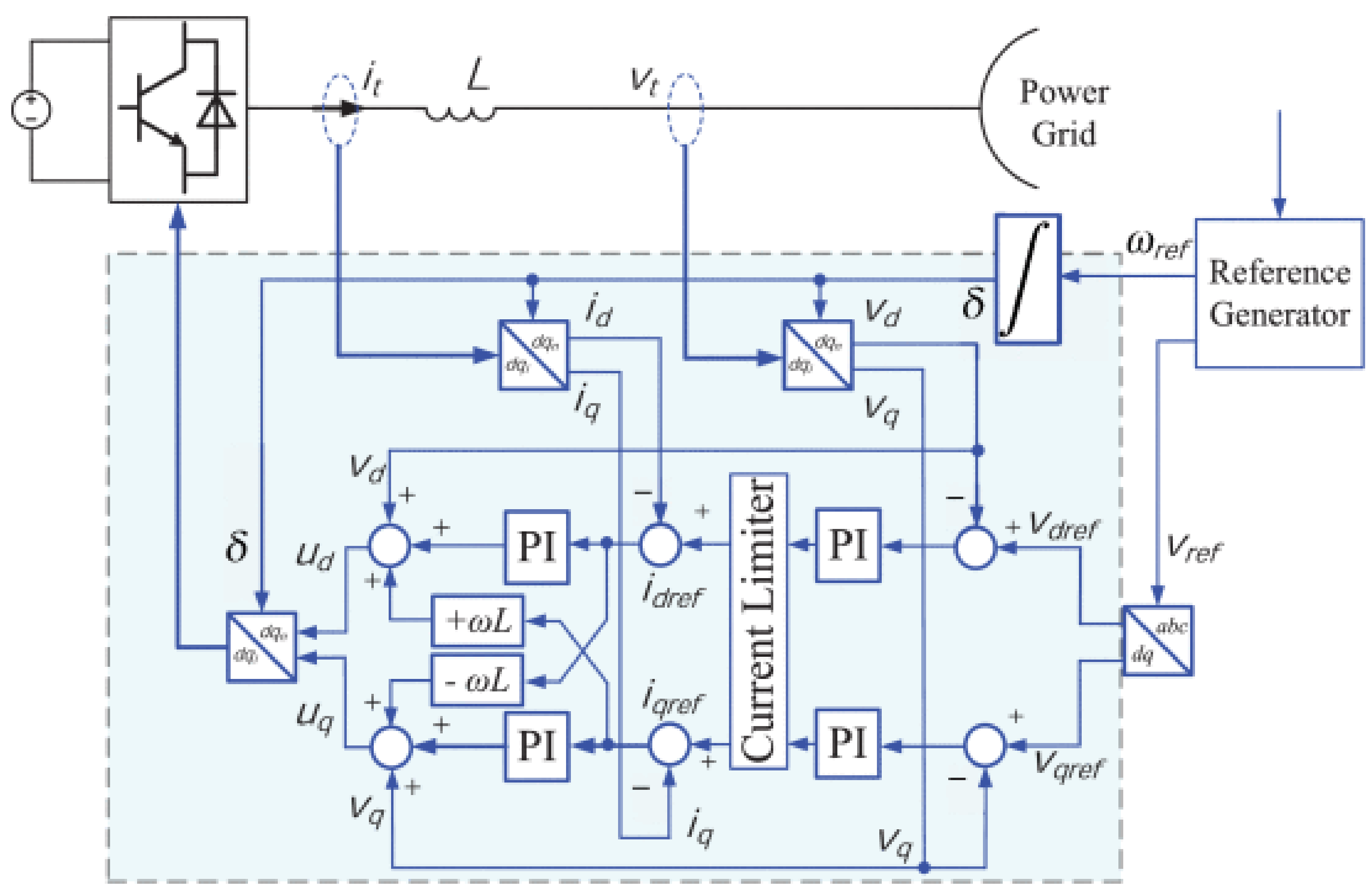
Interface Diverse Tools

Standardize Input Formats

Scale to Large Cases

Modeling Examples

ANDES: Grid Forming Inverter



```
class REGCV1(ModelBase):
    def __init__(self):
        ModelData.__init__(self)
        self.M = NumParam(
            default=10, tex_name='M',
            unit='s', power=True,)
        self.D = NumParam(
            default=0, tex_name='D',
            unit='p.u.', power=True,)
        self.Pref2 = Algeb(
            tex_name=r'P_{ref2}',
            e_str='u*Pref-dw*kw-Pref2',
            v_str='u*Pref')
        self.vref2 = Algeb(
            tex_name=r'v_{ref2}',
            e_str='(u*Qref-Qe)*kv+vref-vref2',
            v_str='u*vref')
        self.dw = State(
            info='delta␣virtual␣rotor␣speed',
            tex_name=r'\Delta\omega',
            unit='pu␣(Hz)',
            v_str='0', t_const=self.M
            e_str='Pref2-Pe-D*dw',)
        self.omega = Algeb(
            info='virtual␣rotor␣speed',
            unit='pu␣(Hz)', tex_name=r'\omega',
            v_str='u', e_str='1+dw-omega')
        self.delta = State(
            info='virtual␣delta',
            unit='rad', tex_name=r'\delta',
            v_str='a', e_str='2*pi*fn*dw')
```

AMS: Virtual Inertia Scheduling in Real-time Economic Dispatch

$$\min_{P,M,D} \sum_{t\in T} \Big[\underbrace{\sum_{i=1}^{N_{sg}} \left(a^{sg}_{i,t}(P^{sg}_{i,t})^2 + b^{sg}_{i,t}P^{sg}_{i,t} + c^{sg}_{i,t} + b^{sg}_{r,i,t}P^{sg}_{i,r,t}\right)}_{SG} \\ + \underbrace{\sum_{i=1}^{N_{ibr}} \left(a^{ibr}_{i,t}(P^{ibr}_{i,t})^2 + b^{ibr}_{i,t}P^{ibr}_{i,t} + c^{ibr}_{i,t} + b^{ibr}_{r,i,t}P^{ibr}_{i,r,t}\right)}_{IBR} \Big] \quad (1)$$

$$P^{ibr}_{s,i,t} + P^{ibr}_{i,r,t} \le P^{\max,ibr}_{i,t} \quad (2)$$

$$P^{ibr}_{i,t} - P^{ibr}_{i,r,t} \ge P^{\min,ibr}_{i,t} \quad (3)$$

$$P^{ibr}_{i,r,t} = \Delta P^{ibr}_{i,peak,t} \quad (4)$$

$$M^{\min,ibr}_{i} \le M^{ibr}_{i,t} \le M^{\max,ibr}_{i} \quad (5)$$

$$D^{\min,ibr}_{i} \le D^{ibr}_{i,t} \le D^{\max,ibr}_{i} \quad (6)$$

$$-\mathrm{RoCoF}_{\lim} \le f_0 \frac{\Delta P_{e,t}}{M_t} \le \mathrm{RoCoF}_{\lim} \quad (7)$$

```
class VISBase:
    def __init__(self):
        ...
        self.M = Var(
            info='Emulated␣inertia␣(M=2H)',
            name='M', tex_name=r'M', unit='s',
            model='VSG', nonneg=True,)
        self.D = Var(
            info='Emulated␣damping',
            name='D', tex_name=r'D', unit='p.u.',
            model='VSG', nonneg=True,)
        self.Mreq = Constraint(
            name='Mreq', type='eq',
            info='Emulated␣inertia␣requirement',
            e_str='-gvsg@M+dvm',)
        self.Dreq = Constraint(
            name='Dreq', type='eq',
            info='Emulated␣damping␣requirement',
            e_str='-gvsg@D+dvd',)

class RTEDVIS(RTED, VISBase):
    def __init__(self, system, config):
        RTED.__init__(self, system, config)
        VISBase.__init__(self)
        gcost = 'sum(mul(c2,power(pg,2)))'
        gcost += '+sum(c1@(t␣dot␣pg))+ug*c0'
        rcost = '+sum(cru*pru+crd*prd)'
        vsgcost = '+sum(cm*M+cd*D)'
        self.obj.e_str = gcost + rcost + vsgcost
```