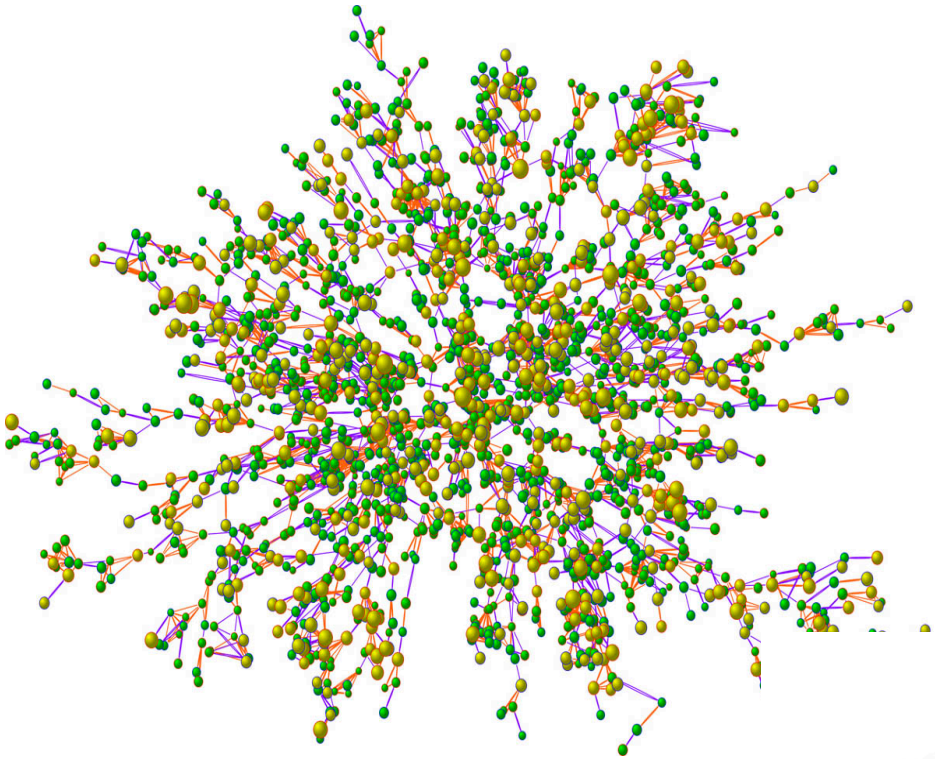
DMNetwork, TSEvent, and Power Grid Simulation

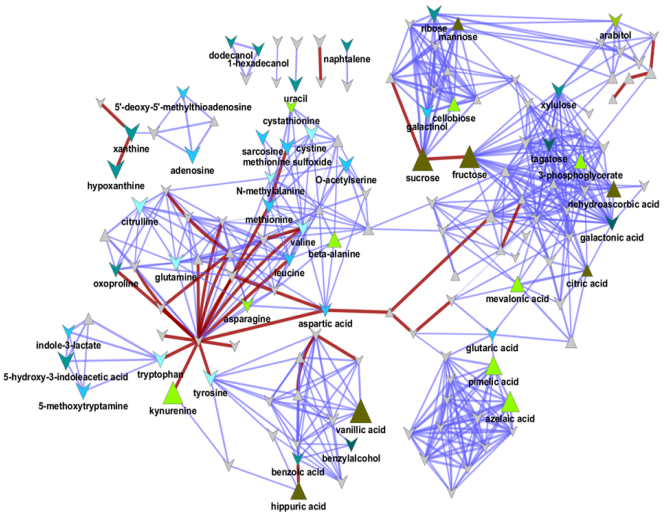
Shri Abhyankar

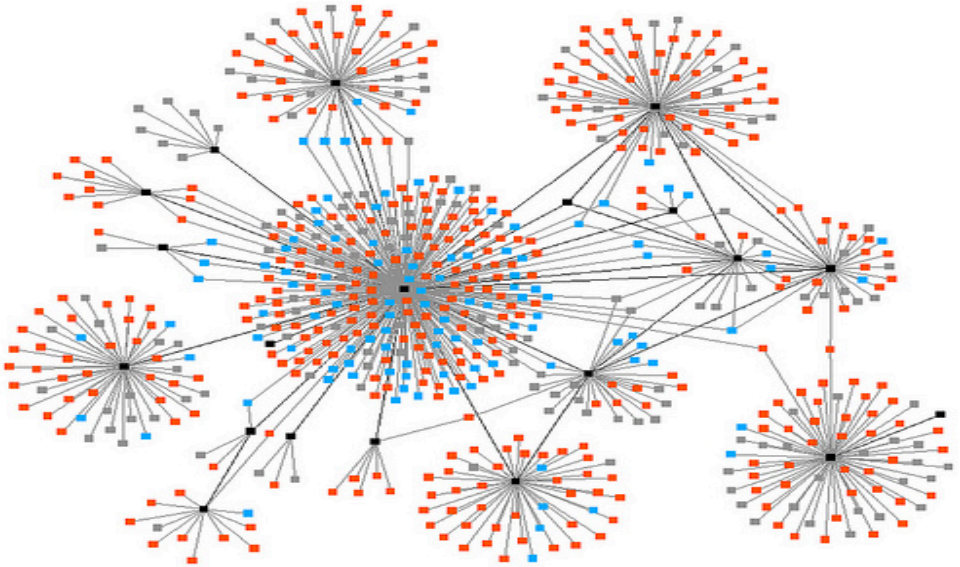
Energy Systems Division

Argonne National Laboratory

abhyshr@anl.gov




ribose
maninose
arabitol
dodecanol
1-hexadecanol
naphtalene
uracil
5'-deoxy-5'-methylthioadenosine
cystathionine
xylulose
cellobiose
galactinol
xanthine
sarcosine
cystine
methionine sulfoxide
tagatose
adenosine
sucrose
fructose
3-phosphoglycerate
hypoxanthine
O-acetylserine
N-methylalanine
dehydroascorbic acid
citrulline
methionine
galactonic acid
valine
beta-alanine
leucine
glutamine
oxoproline
citric acid
mevalonic acid
asparagine
aspartic acid
glutaric acid
indole-3-lactate
5-hydroxy-3-indoleacetic acid
tryptophan
pimelic acid
tyrosine
5-methoxytryptamine
kynurenine
azelaic acid
vanillic acid
benzylalcohol
benzoic acid
hippuric acid

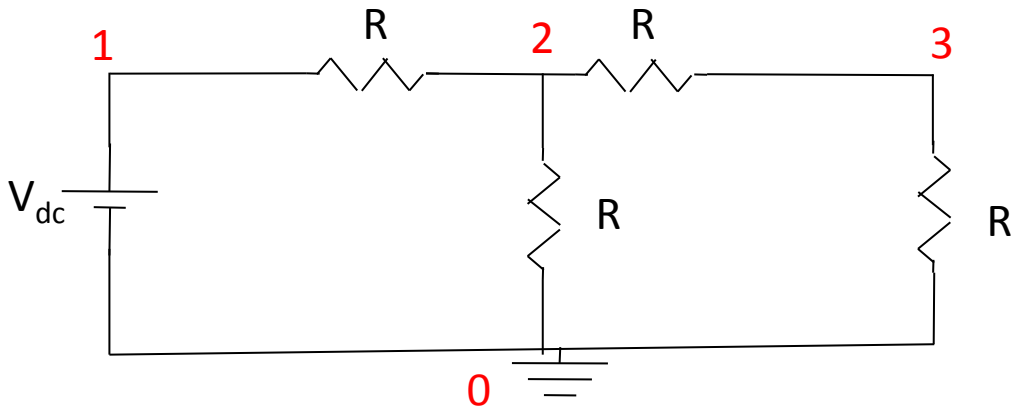


DMNetwork

DMNetwork

- DM for easily expressing and managing unstructured network problems
  - Circuits, graphs, power grid, gas networks, communication networks, radio networks, computer networks, transportation
- Design elements
  - Vertex
  - Edge
  - Component (Associated data)
- Built on top of DMPlex
- Power network example: $PETSC_DIR/src/snes/examples/tutorials/network/pflow/pflow.c

DMNetwork Application Flow

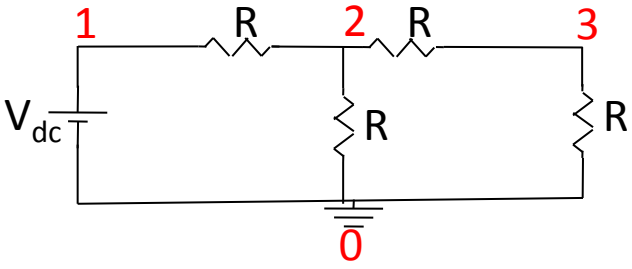


1. Create network object and component library

```
DMNetworkCreate(MPI_Comm,&dmnetwork);

DMNetworkRegisterComponent(dmnetwork,"DC voltage",
                           sizeof(Vdc),&Vdc_key);
DMNetworkRegisterComponent(dmnetwork,"Resistor",
                           sizeof(R),&R_key);
DMNetworkRegisterComponent(dmnetwork,"Ground",
                           sizeof(Gnd),&Gnd_key);
```

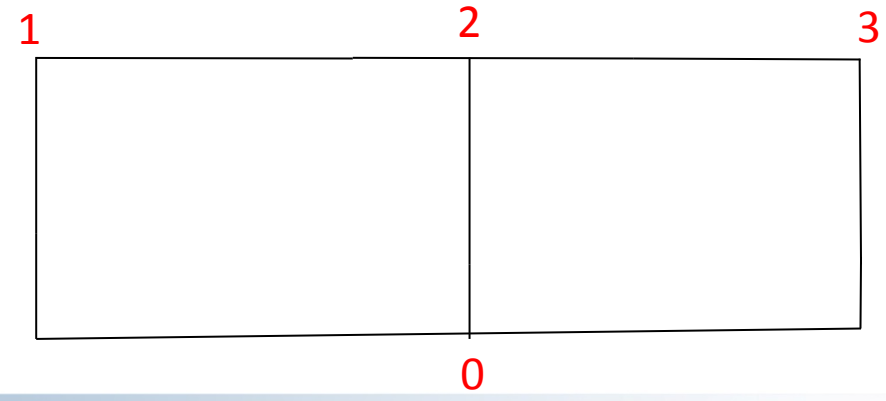
DMNetwork Application Flow



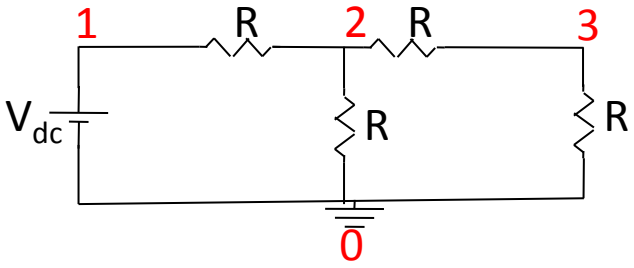
2. Set sizes and create layout

```
DMNetworkSetSizes(dmnetwork, 4, 5, PETSC_DECIDE,
                  PETSC_DECIDE);

int edgeconns = {0,1,1,2,2,3,2,0,3,0};
DMNetworkSetEdgeList(dmnetwork,edgeconns);
DMNetworkLayoutSetUp(dmnetwork);
```



DMNetwork Application Flow



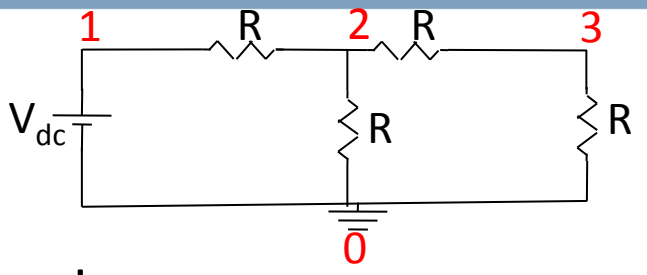
3. Add variables and components

```
DMNetworkGetEdgeRange(dmnetwork,&eStart,&eEnd);
DMNetworkGetVertexRange(dmnetwork,&vStart,&vEnd);

for(i = vStart; i < vEnd; v++) {
  DMNetworkSetNumVariables(dmnetwork,i,1);
  if( i == vStart) {
     DMNetworkAddComponent(dmnetwork,i,Gnd_key,void*(&Gnd));
  }
}

for(i = eStart; i < eEnd; i++) {
  if(i == estart) {
     DMNetworkAddComponent(dmnetwork,i,Vdc_key,void*(&Vdc));
  } else {
     DMNetworkAddComponent(dmnetwork,i,R_key, void*(&R));
  }
}
DMSetUp(DM dm);
```

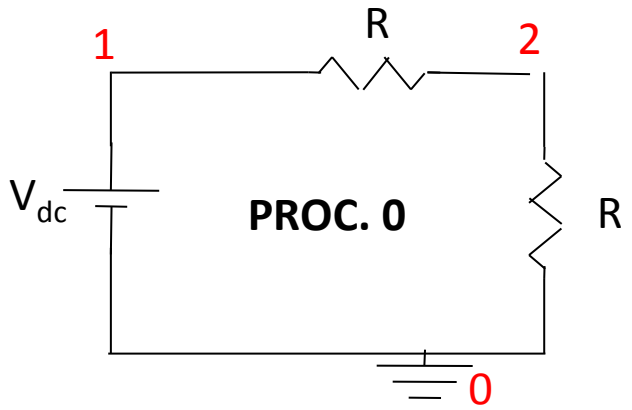
DMNetwork Application Flow

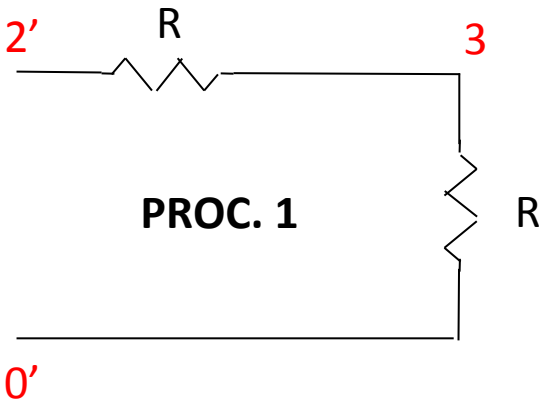


4. Set up and distribute (partition) the network

```
DMNetworkDistribute(dmnetwork, "parmetis",0,&distdm);
```

- Does a non-overlapping edge partitioning
- Moves the components to the appropriate processor





DMNetwork

5. Hook up the solver

```
TSSetDM(ts,distdm);/* Time-stepping */
SNESSetDM(snes, distdm); /* Nonlinear */
KSPSetDM(ksp,distdm); /* Linear */
```

- Helpful DMNetwork functions
  - Adjacent node(s), edge(s) retrieval
  - Local/Global vector creation, Matrix creation, Local <--> Global
  - Local and global offsets

- Current applications
  - Power Grid
  - Radio Network
  - Gas network
  - River network

TSEvent

TSEvent

- Support for discontinuous ODE/DAEs

$$f(x, \dot{x}, t) = \begin{cases} f_1(x, \dot{x}, t) & \text{if } g(t, x) < 0 \\ f_2(x, \dot{x}, t) & \text{if } g(t, x) > 0 \end{cases}$$

Switching condition

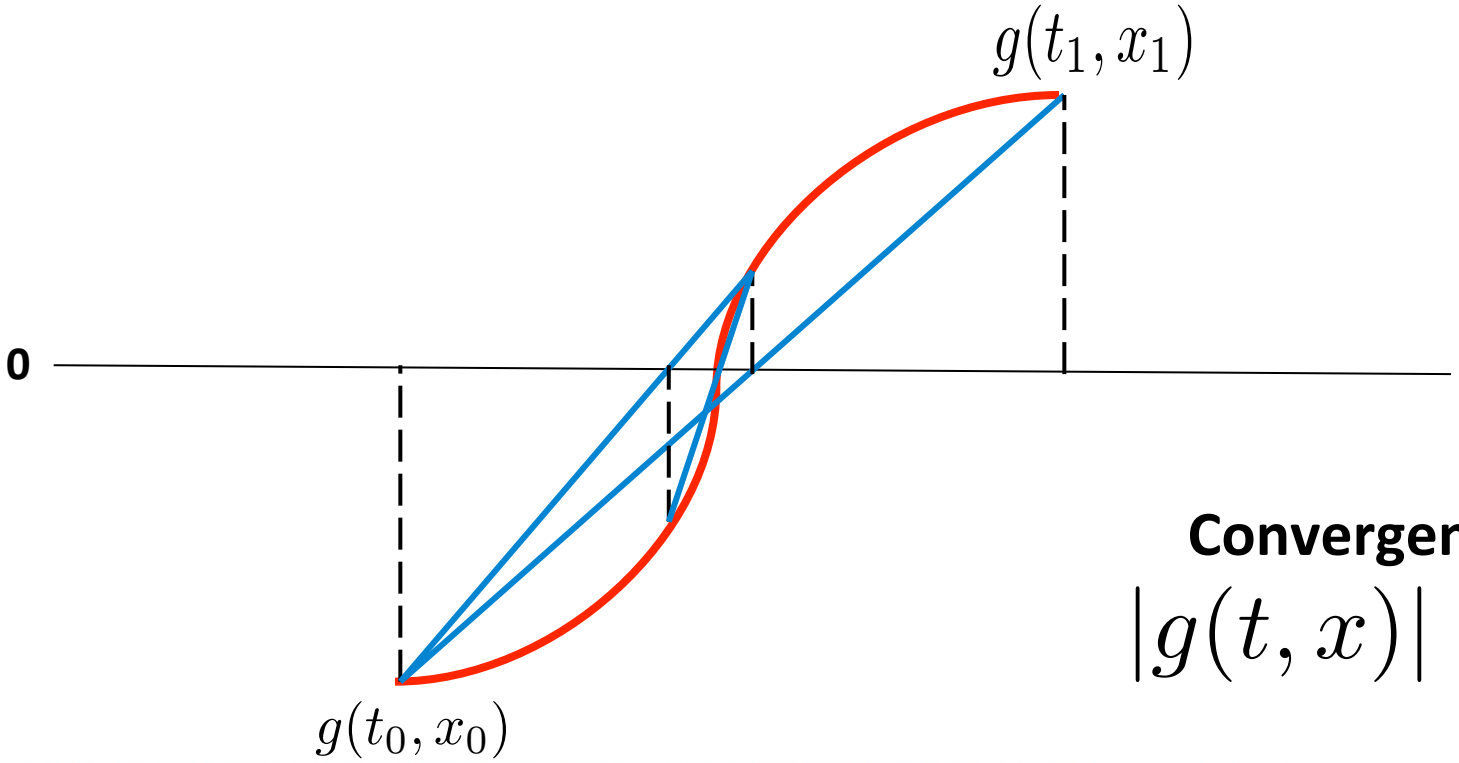
- TS does integration along with finding “events” -- zero crossings of g(t,x)
- Currently available with TSTheta, TSARKIMEX, and TSROSW

TSEvent: Event detection and location

- Event detection

$$\mathrm{sgn}(g(t,x)) \neq \mathrm{sgn}(g(t+\Delta t,x))$$

- Event location
  - Double false position method



Convergence

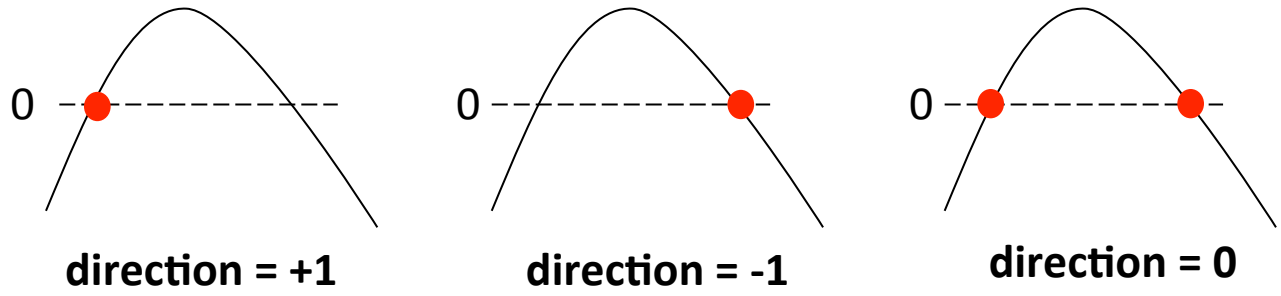
$$|g(t,x)| \leq \epsilon$$

TSEvent: Usage

```
TSSetEventMonitor(TS ts,PetscInt
nevents,PetscInt *direction, PetscBool
*terminate,PetscErrorCode (*eventmonitor)
(..), PetscErrorCode (*postevent)(..),void
*ctx);
```

nevents – number of events (local)

direction – type of zero crossing detection

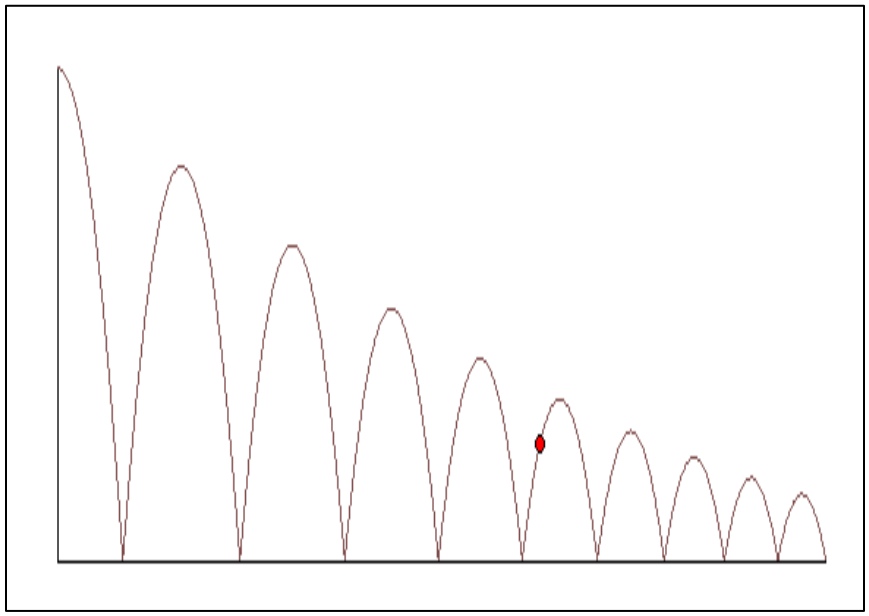


terminate – terminate on event detection?

eventmonitor – Event monitoring function

postevent – Post event handling routine

TSEvent Example: Bouncing ball example ts/examples/tutorials/ex40.c



Ball dynamics

$$\frac{du}{dt} = v$$
$$\frac{dv}{dt} = -9.8$$

Event Function

$$g(t,x) = u$$

PostEvent function

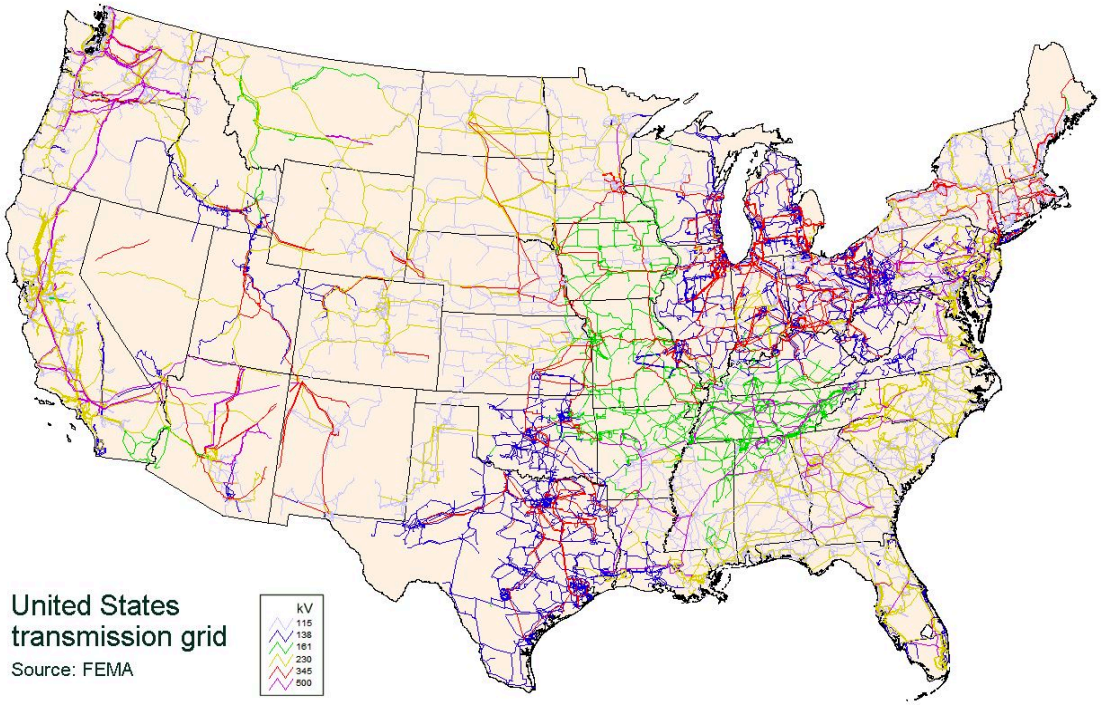
$$v = -0.9v$$

10% reduction in speed

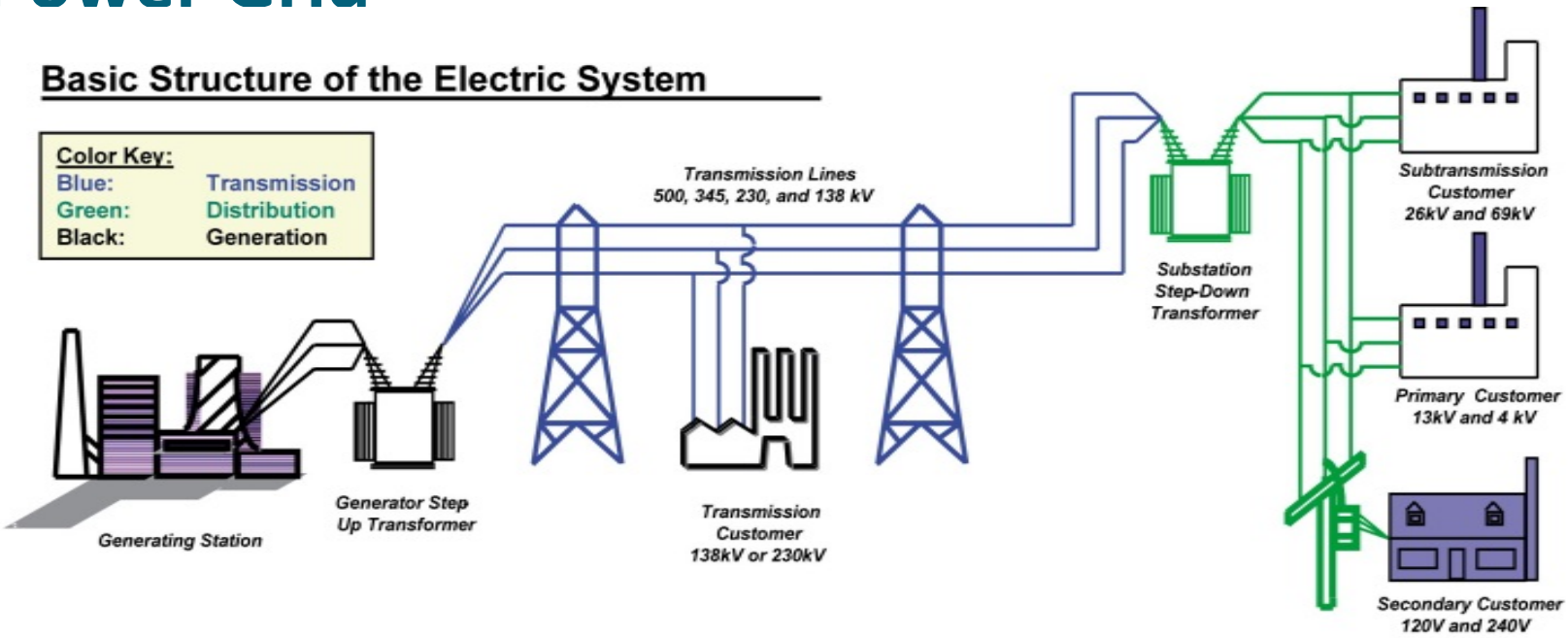
TSEvent

- Handles multiple events within a time-step
- Vector tolerances
- Compatible with TSAdjoint
- Examples
  - $PETSC_DIR/src/ts/examples/tutorials/ex40.c, ex41.c (bouncing ball)
  - $PETSC_DIR/src/ts/examples/tests/ex22.c (1-D quench front problem)
  - $PETSC_DIR/src/ts/examples/tutorials/ex3adj_events.c (Power grid ODE problem with adjoints)

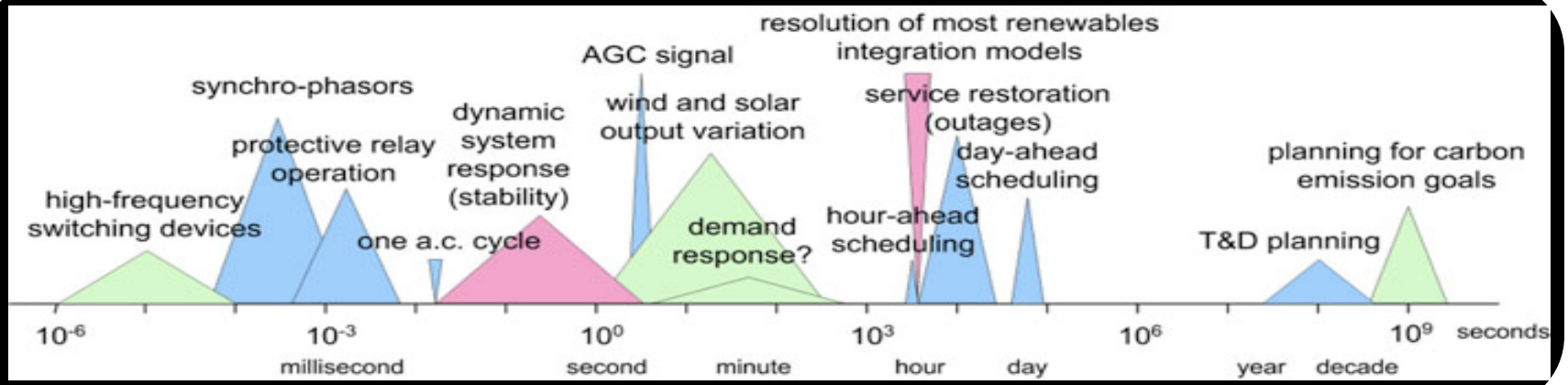
Power Grid


United States
transmission grid
Source: FEMA
kV
115
138
161
230
345
500

Power Grid



Study time-scales

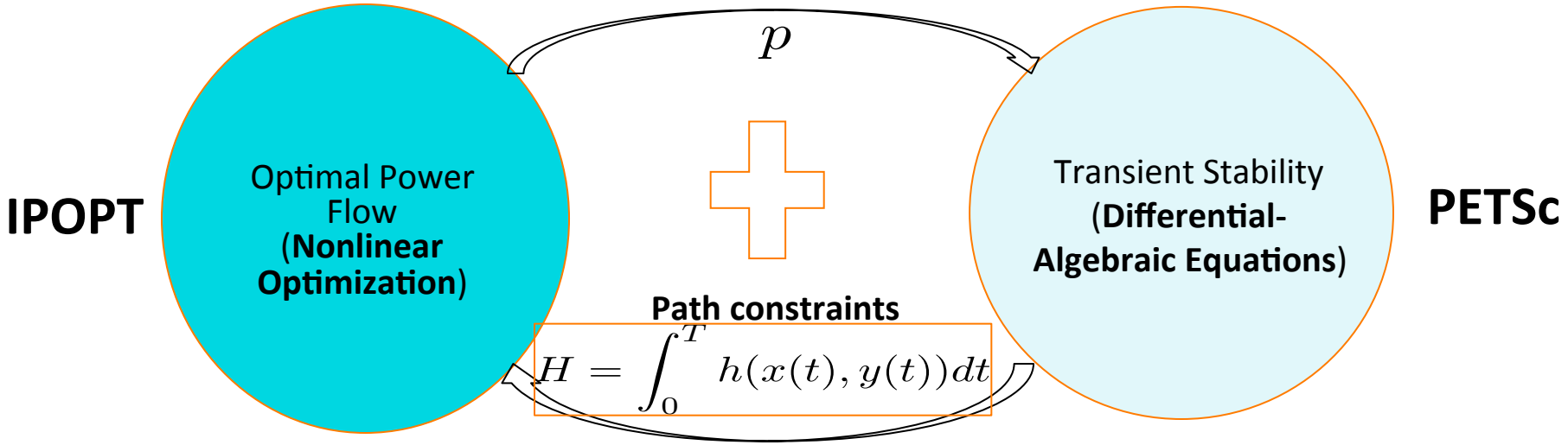


Power grid applications developed using PETSc

Application	Mathematical Model	Solver
Load Flow	Nonlinear	SNES
Contingency analysis	Nonlinear (embarrassingly parallel)	SNES
Dynamics	Differential-Algebraic with discontinuities	TS
Temporal Multiscale	Differential-Algebraic with 2 different time-steps	SNES
Trajectory Sensitivity	DAE Discrete Adjoint	TS
Dynamics constrained optimization	Convex NLP with DAE constraints	TS + IPOPT

- GridPACK project (https://www.gridpack.org)

DAE constrained optimization



$$\begin{aligned} \min \quad & C(p) \\ \text{s.t.} \quad & g_s(p) = 0 \\ & h_s(p) \le h^+ \\ & p^- \le p \le p^+ \end{aligned}$$

Adjoint-sensitivity based gradient calculation

$$\begin{aligned} \dot{\lambda} &= -f_x^T \lambda + g_x^T \mu - h_x \\ 0 &= -f_y^T \lambda + g_y^T \mu - h_y \end{aligned}$$

$$\nabla_p H = \int_0^T f_p^T \lambda \, dt - (x_p^T \lambda)\Big|_{t=0}$$

$$\begin{aligned} \dot{x} &= f(x, y, p), \quad x(t_0) = I_{x0}(p) \\ 0 &= g(x, y, p), \quad y(t_0) = I_{y0}(p) \end{aligned}$$

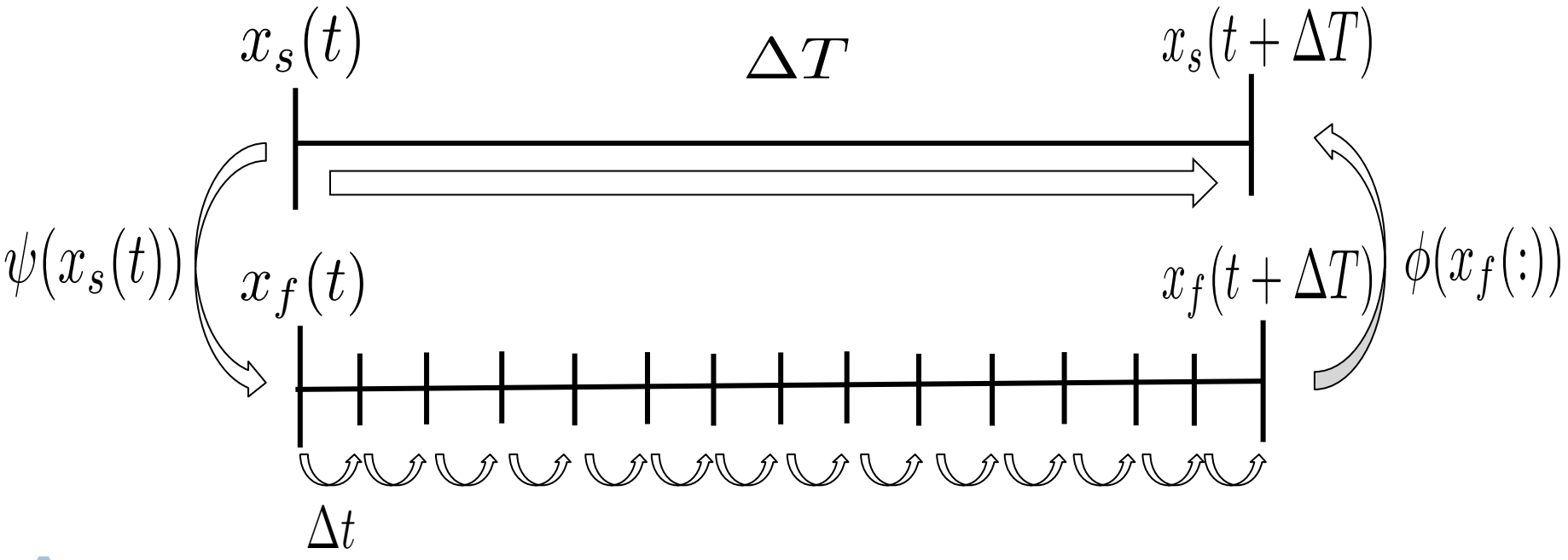
Solving temporal multiscale problems: Explicit coupling approach

Two-scale problem

$$\dot{x_s} = F(x_s, x_f)$$ Slow time-scale

$$\dot{x_f} = f(x_f, x_s)$$ Fast time-scale

Each time-step of 'Explicit Coupling' approach



Fully-implicit approach for temporal multiscale problems

- Form a larger system of equations that contains discretized equations for 1 time-step of slow scale + *n* coupled-in-time fast time-scale steps

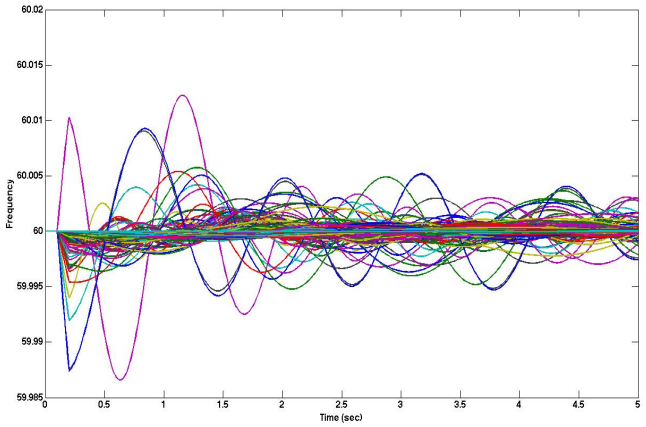
$$\begin{bmatrix} x_s(t+\Delta T) - x_s(t) - F(x_s(t), \phi(x_f(:))) \\ x_f(t+\Delta t) - x_f(t) - f(x_f(t+\Delta t), \psi(x_s(t), x_s(t+\Delta T)) \\ x_f(t+2\Delta t) - x_f(t+\Delta t) - f(x_f(t+2\Delta t), \psi(x_s(t), x_s(t+\Delta T)) \\ x_f(t+3\Delta t) - x_f(t+3\Delta t) - f(x_f(t+3\Delta t), \psi(x_s(t), x_s(t+\Delta T)) \\ \vdots \\ x_f(t+\Delta T) - x_f(t+\Delta T - \Delta t) - f(x_f(t+\Delta T), \psi(x_s(t), x_s(t+\Delta T)) \end{bmatrix} = 0$$

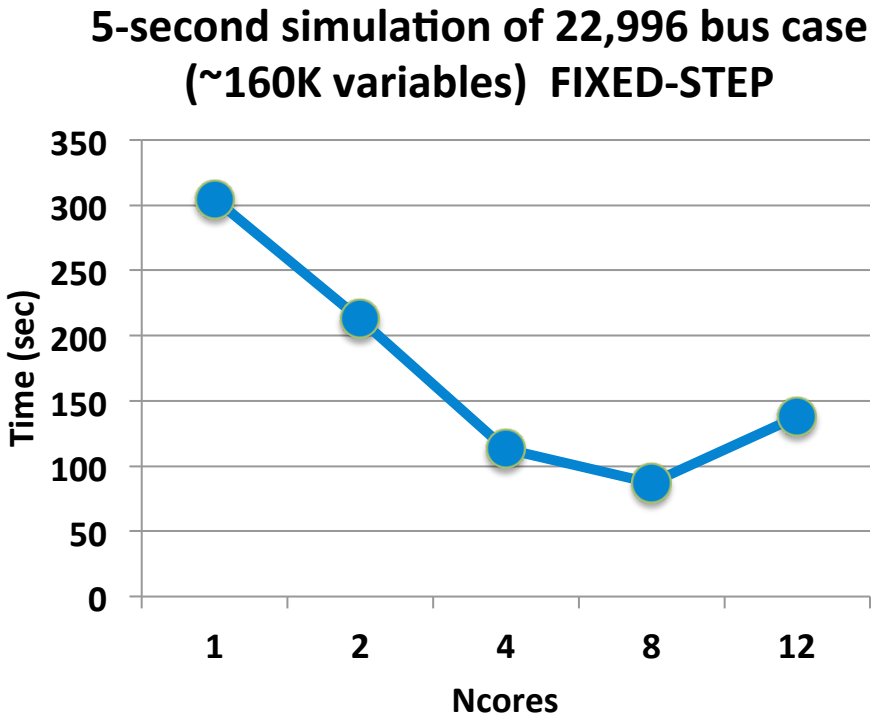
- More robust approach
- PCFIELDSPLIT!!

Real-Time Power System Dynamic Simulation

- Objective
  - Develop faster than real-time power grid dynamics simulation capabilities.

$$\dot{x} = f(x, y, p), \quad x(t_0) = I_{x0}(p)$$
$$0 = g(x, y, p), \quad y(t_0) = I_{y0}(p)$$





3rd order TSARKIMEX + KSPGMRES + PCASM + LU (sub-block) + AMD ordering

Ncores	Error-norm type	Execution time (sec)
8	2-norm	**4.25 (under real-time)**
8	Infinity	11.51

QUESTIONS?