Dynamic Modeling and High Performance Computing Simulation for Power Grid

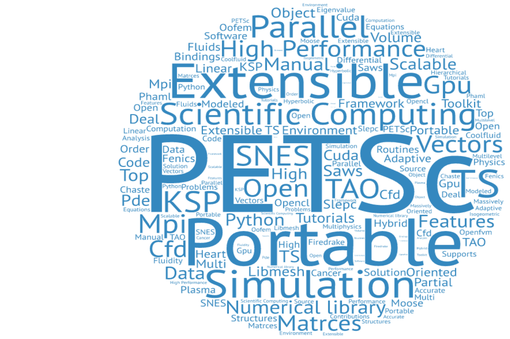
Mathematics and Computer Science and Energy Systems

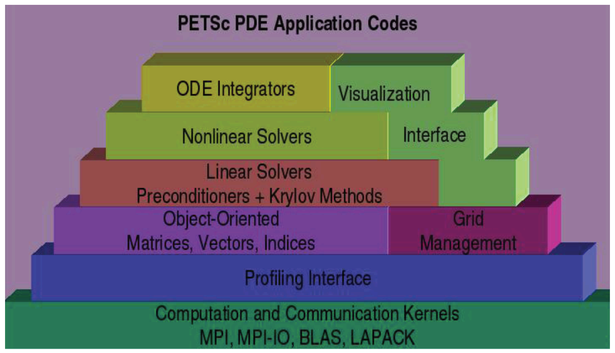
Capabilities

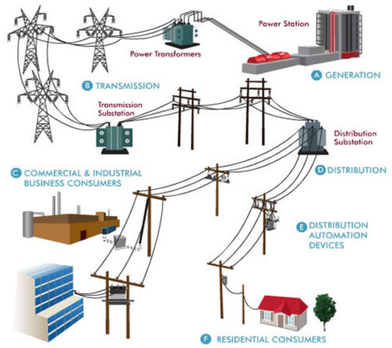
- High-performance computing modeling and simulation of power grid systems
- Fast solvers harnessing parallelization through domain decomposition and graph partitioning
- Scalable linear, nonlinear, and adaptive time-stepping solvers.

Solutions and Extensions

- Support for large-scale computing from models to super-computing and real-time solutions
- *High-end solvers*
- *Provides automatic link to native to optimization algorithms.*
- *Computed automatically gradients and sensitivitivities*







Dynamics

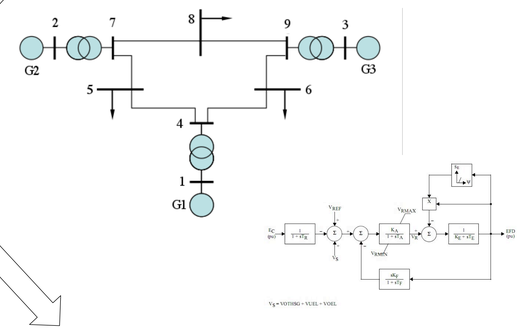
Contingency Analysis

Optimal Power Flow

Power Flow

Dynamics Constrained Optimal Power Flow

Application



Power System

$$\frac{dx_{gen}}{dt} = f(x_{gen}, I_{dq}, V_{DQ})$$
$$0 = h(x_{gen}, I_{dq}, V_{DQ})$$

Generator

$$\begin{bmatrix} G & -B \\ B & G \end{bmatrix} \begin{bmatrix} V_D \\ V_Q \end{bmatrix} = \begin{bmatrix} I_{gen,D}(x_{gen}, I_{dq}) \\ I_{gen,Q}(x_{gen}, I_{dq}) \end{bmatrix} - \begin{bmatrix} I_{load,D}(x_{load}, V_{DQ}) \\ I_{load,Q}(x_{load}, V_{DQ}) \end{bmatrix}$$

$$\frac{dx_{load}}{dt} = f_2(x_{load}, V_{DQ})$$

Load

Math, Solver, Simulation (PETSc)

$$e_{n+1} = \frac{x_{n+1} - x^*_{n+1}}{\epsilon}$$

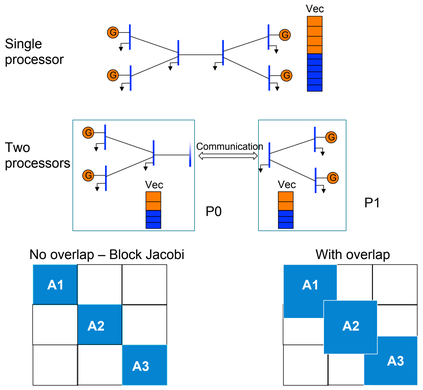
$$\epsilon = \epsilon_{abs} + \epsilon_{rel} * max(x_{n+1}, x^*_{n+1})$$

$$\Delta t_{n+1} = \Delta t_n \|e_{n+1}\|^{-1/p}$$

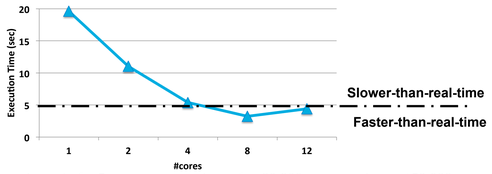
Computing and Analysis

```
/* Exciter Jacobian */
if(dyngen->dynexc) {
  DYNExcModel dynexc;

  ierr = PSGENGetDYNExc(gen,&dynexc);CHKERRQ(ierr);

  dynlocglob = locglob + dynexc->startloc;
  ierr = DYNExcModelDAERHSJacobian(dynexc,J,t,VD,VQ,xdyn,dynlocglob,V_loc);CHKERRQ(ierr);
  for(j=0; j < dynexc->nvar; j++) {
    row[0] = col[0] = dynlocglob+j;
    val[0] = -a*dynexc->eqtypes[j];
    ierr = MatSetValues(J,1,row,1,col,val,ADD_VALUES);CHKERRQ(ierr);
  }
}

/* Turbine governor Jacobian */
if(dyngen->dynturbgov) {
  DYNTurbgovModel dynturbgov;

  ierr = PSGENGetDYNTurbgov(gen,&dynturbgov);CHKERRQ(ierr);

  dynlocglob = locglob + dynturbgov->startloc;
  ierr = DYNTurbgovModelDAERHSJacobian(dynturbgov,J,t,VD,VQ,xdyn,dynlocglob,V_loc);CHKERRQ(ierr);
  for(j=0; j < dynturbgov->nvar; j++) {
    row[0] = col[0] = dynlocglob+j;
    val[0] = -a*dynturbgov->eqtypes[j];
    ierr = MatSetValues(J,1,row,1,col,val,ADD_VALUES);CHKERRQ(ierr);
  }
}
```

Multiple processors (cores) used for solving the problem



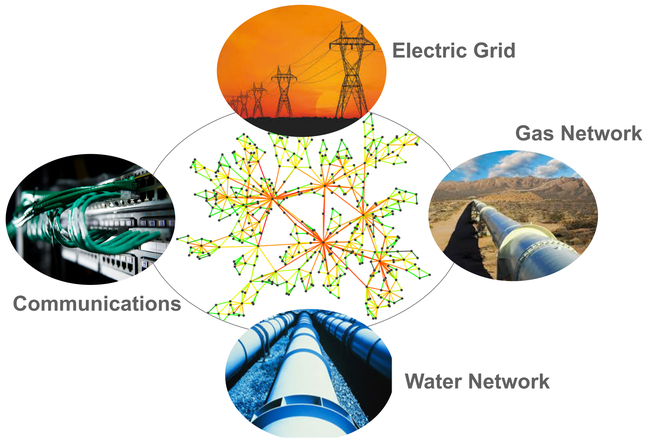
Achieving Real-Time Dynamic Simulation Speed: Putting it all together: Test case 2



Scalability plot of a 5 second simulation of a 20,000 node system ~ 150,000 variables

- **Achieved real-time speed of under 5 seconds execution time on 8 cores.**
- **Execution time using state-of-the-art algorithm on single core = 300 seconds**

Power Grid and Beyond…



Simulation of network of networks

DMNetwork

- Abstractions for rapidly developing network of networks simulations on high-performance machines
- Freely available with PETSc distribution
- Generic design suited for most network simulations

Contact **Mihai Anitescu (MCS), Emil Constantienscu (MCS), Shrirang Abhyankar (ES & MCS).**

Argonne NATIONAL LABORATORY