An overview of RAMSES

(RApid Multiprocessor Simulation of Electric power Systems)

Petros Aristidou and Thierry Van Cutsem

Dept. Electrical Eng. & Comp. Sc.,
Liège, Belgium

June 2019

LIÈGE université
Sciences Appliquées

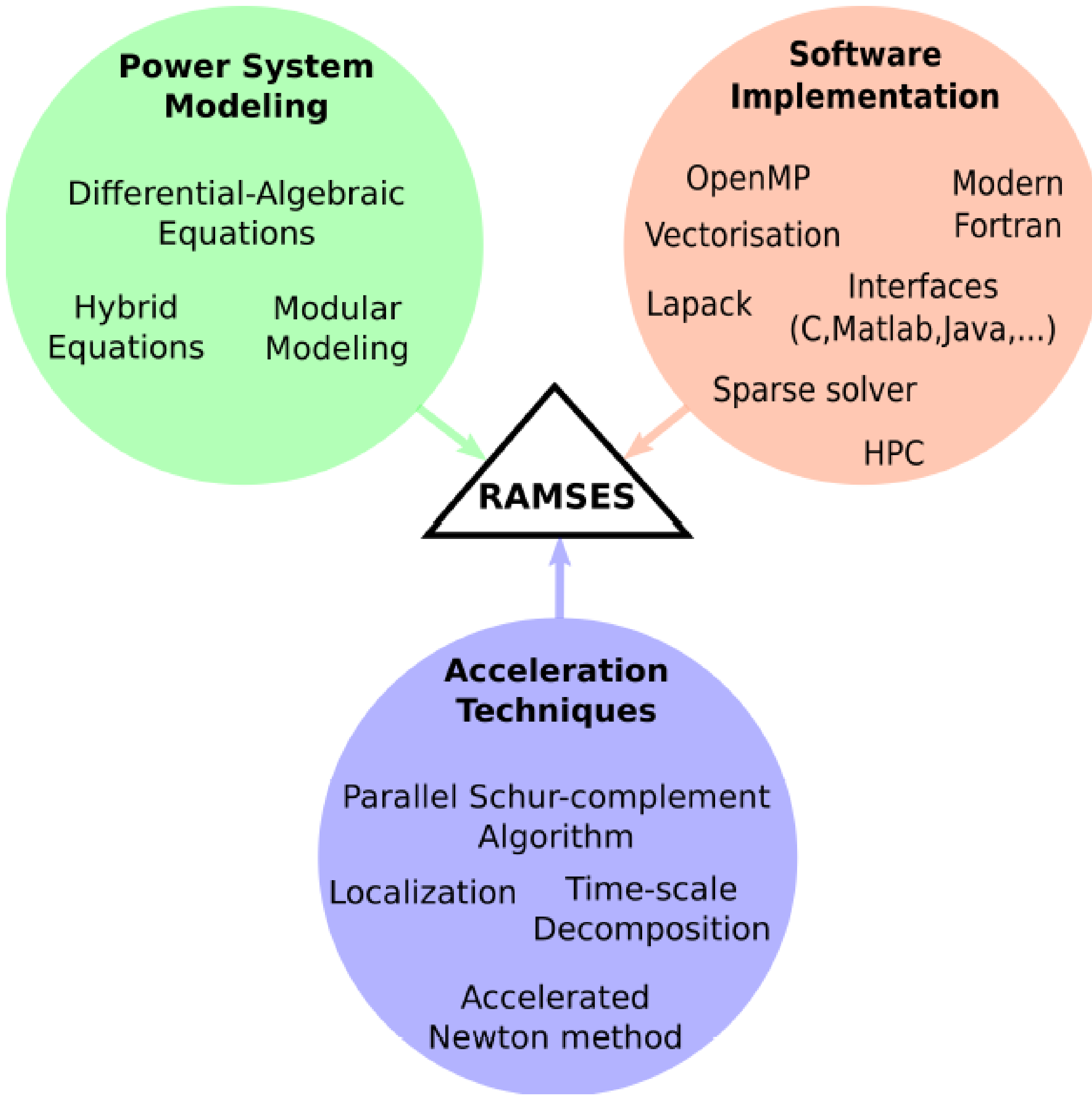
Dynamic simulation needs

- Power systems equipped with more and more controls reacting to disturbances, with either beneficial or detrimental effects
  - requires simulating dynamic responses
  - static security analysis no longer sufficient
- larger and larger models considered
  - simulation of large interconnections
  - incorporation of sub-transmission and distribution levels
    - distribution grids expected to host more and more distributed energy sources
    - active distribution network management impacts overall system response
  - explicit modelling preferred to equivalents
- longer simulated times
  - check response of system up to its return to steady state
    - long-term dynamics : typically several minutes after initiating event
- faster than real-time (simulators, prediction capability, etc.)

Speeding up simulations

- Conventional simulation codes and serial computing platforms have met their limits
- dynamic simulation software must be revisited to cope with large models and take advantage of computer technology
- multi-core computers available widely and at affordable price
  - deployed to overcome the limits of single-core computers
  - parallel computing tools are available
- attempts to parallelize *existing* power system simulation software reveal themselves unsuccessful
- *new solution schemes* must be devised to exploit parallelism
  - RAMSES was developed with this objective in mind, based on system decomposition

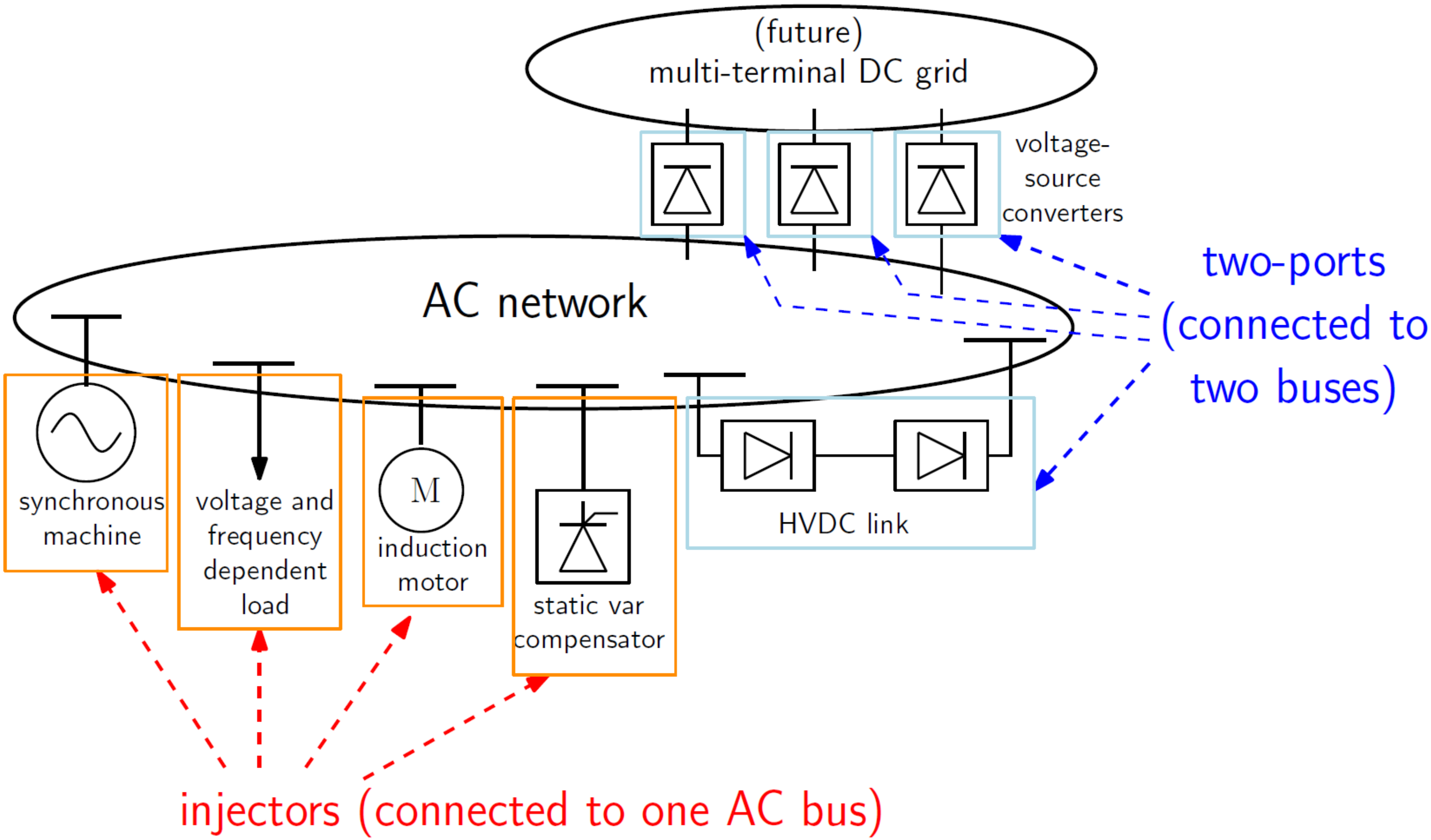
RAMSES ingredients



Power system modelling in RAMSES

Power system =

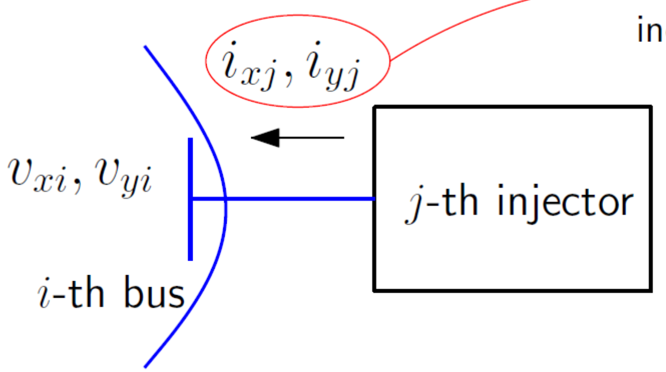
AC network(s) + DC grid(s) + injectors + two-ports



Modelling : modularity and flexibility

Each injector :

- is interfaced to the AC network through the (rectangular components of) bus voltage and injected current
- is modelled with its own Differential-Algebraic (DA) equations
  - algebraic equations yield higher modelling flexibility
  - the solver handles equations changing between differential and algebraic



components of injected current included in the injector state vector

$$\boldsymbol{\Gamma}_j \dot{\mathbf{x}}_j = \Phi(v_{xi}, v_{yi}, \mathbf{x}_j)$$

with $(\boldsymbol{\Gamma}_j)_{\ell\ell} = 0$ if the ℓ-th equation is algebraic
$= 1$ if the ℓ-th equation is differential

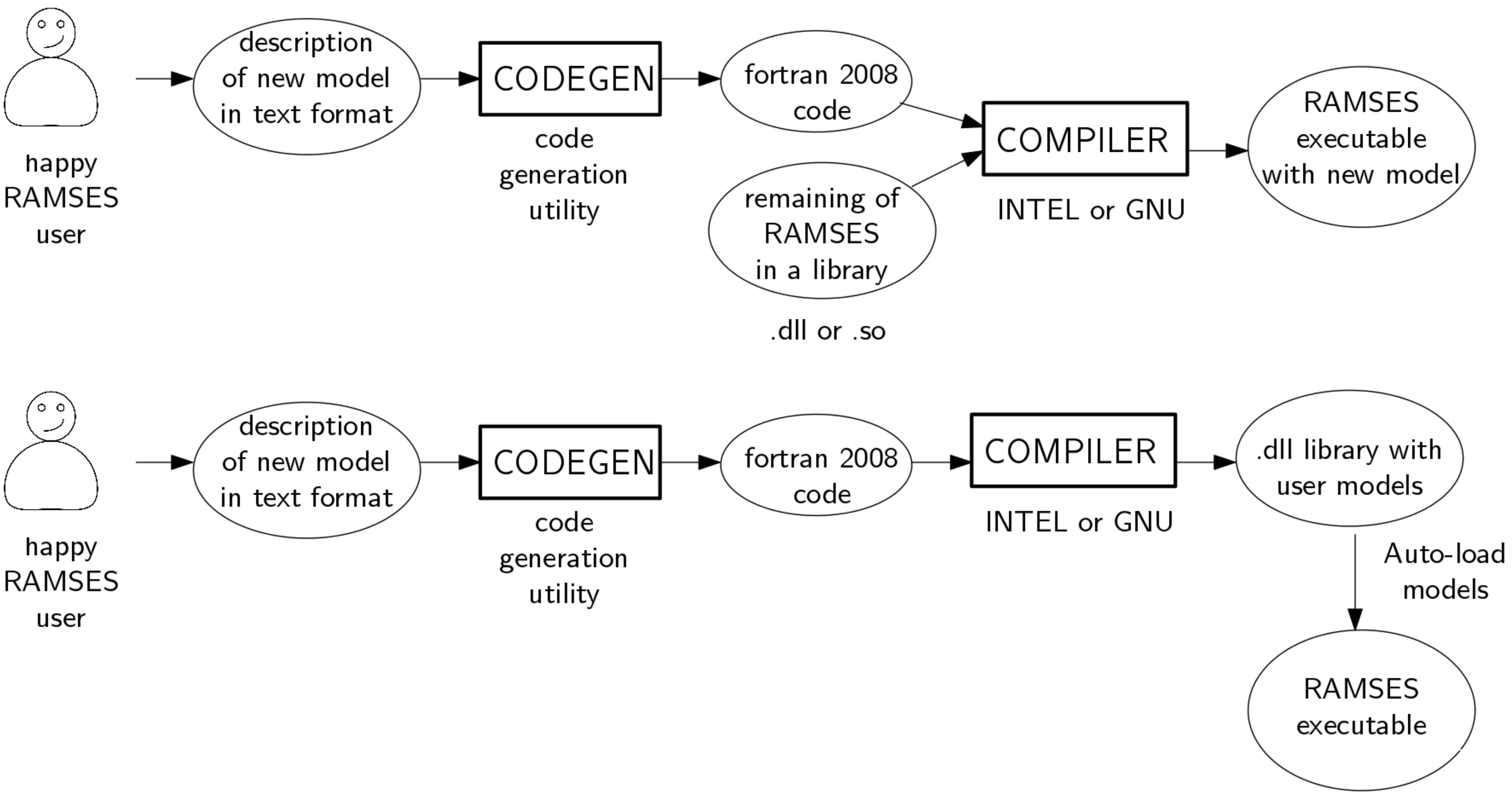
… and similarly for two-ports

Component models (I)

- Hard-coded :
  - synchronous machine, voltage and frequency dependent load
- open-source :
  - induction machine, some IEEE models of excitation and torque controls of synchronous machine, wind generators, etc.
- user-defined :
  - 4 categories : torque control of synchronous machine, excitation control of synchronous machine, injector and two-port
  - compiled and linked to RAMSES for computational efficiency

Component models (II)

- Two ways to include models (second currently only under Windows)



Discrete-time controls

- Monitor some observables from the simulation of the D-A models
- modify some parameters in those models
- act at discrete times
  - when a condition is fulfilled, or at multiple of their internal activation period
  - applied after the simulation time step is completed.

- Examples of applications:
  - distributed controllers
    - under-frequency and under-voltage load shedding
    - generator protections, etc.
  - wide-area monitoring and/or control
    - tracking state estimation : RAMSES used to simulate SCADA and PMUs
    - secondary frequency control
    - secondary voltage control
    - centralized load shedding against voltage instability
    - coordinated control of dispersed generation units in distribution grids, etc.

Acceleration techniques – parallel processing

- Based on decomposition of model according to :

  network(s) + injectors + two-ports

- injectors (and two-ports) solved independently of each other
- same solution as a non-decomposed scheme by resorting to the Schur-complement for the network equations

- tasks pertaining to components assigned to a number of *threads*, e.g.
  - update and factorization of local Jacobian
  - computation of mismatch vector (of Newton method)
  - computation of contribution to Schur-complement matrix
  - solution of local linear systems, etc.

- threads executed each on a separate processor
  - computational load balanced among the available processors
- *shared-memory* parallel programming model
  - through *OpenMP* Application Programming Interface (API)

Acceleration techniques – localization

After a disturbance, the various components of a (large enough) system exhibit different levels of dynamic activity.

This property is exploited at each time step to :

- accelerate Newton scheme
  - thanks to the decomposed solution scheme, Newton iterations are skipped on components that have already converged
- exploit component latency
  - injectors with high (resp. low) dynamic activity are classified as *active*; the others as *latent*
  - active injectors have their original model simulated
  - latent injectors are replaced by automatically calculated, sensitivity-based models to accelerate the simulation
  - a fast to compute metrics is used to classify the injectors
  - injectors seamlessly switch between categories according to their activity

Acceleration techniques – time-scale decomposition

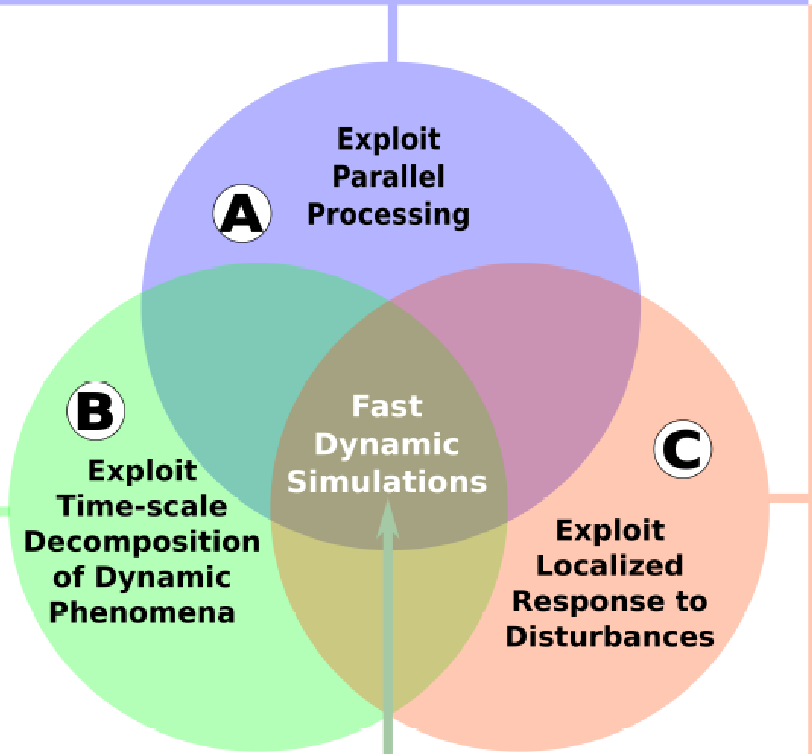
- When only the long-term behaviour of the system is of interest, RAMSES can provide a faster to compute *time-averaged* response
  - unlike with the "quasi steady-state" approximation, no model simplification is performed
  - instead, the original system model is simulated with larger time-steps to "filter out" the fast dynamics
- a *stiff-decay* (or L_1*-stable*) integration scheme is used to this purpose
- with a dedicated treatment of the discrete part of the model (limits, switchings, etc.) by the solver.

- Example of application:
  - 5-10 seconds after a fault : simulation with time steps of ¼ to ½ cycle
  - from $t \approx$ 5-10 seconds until $t \approx$ several minutes : simulation with time steps of 2 to 6 cycles

Acceleration techniques

- use shared-memory parallel processing techniques to accelerate the solution of the decomposed DAE system
- up to **4.5x** faster execution

- use time-averaging to "filter" out fast dynamics and concentrate on average evolution
- use for long-term dynamics
- use "stiff-decay" (L-stable) integration scheme
- use "large" time-steps
- use proper, ex-post, treatment of discrete events



- many disturbances affect only a subset of injectors
- converged injector models stop being solved
- during the simulation, injectors showing high dynamic activity are classified as ***active*** and the original DAE model is simulated. Injectors showing low dynamic activity are classified as ***latent*** and are replaced by simple, linear, automatically calculated models.
- up to **4x** faster execution
- simulation can stop early if all injectors become latent

- accurate simulations with up to **11x** faster execution
- look-ahead, faster than real-time dynamic simulations for systems up to 8000 buses (approx. 75 000 dynamic states) on 24-core, shared-memory computer

Software implementation

- Written in modern (2008) FORTRAN using the *OpenMP* API for shared-memory parallel programming

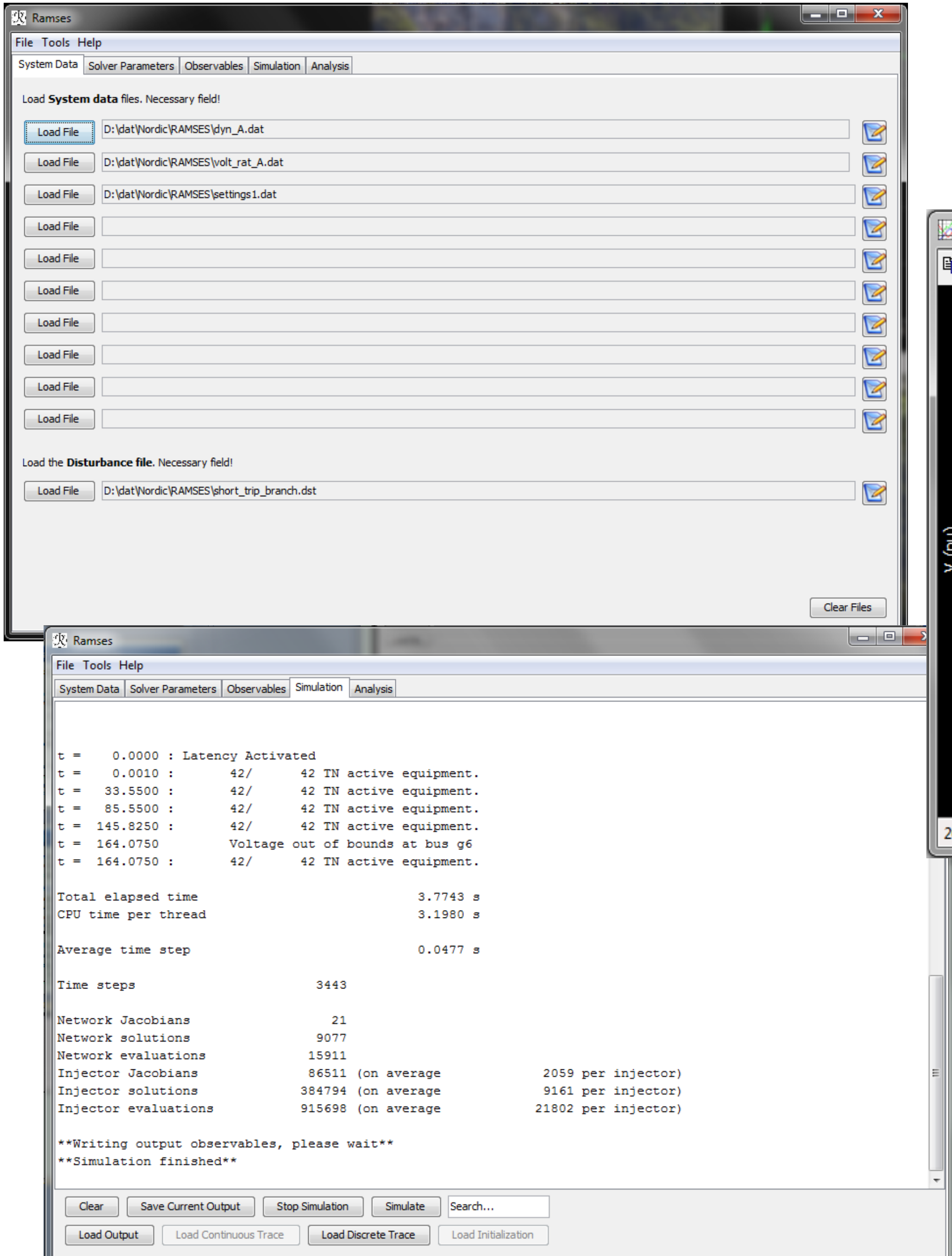
- general implementation : no “hand-crafted” optimization particular to the computer system, the power system or the disturbance

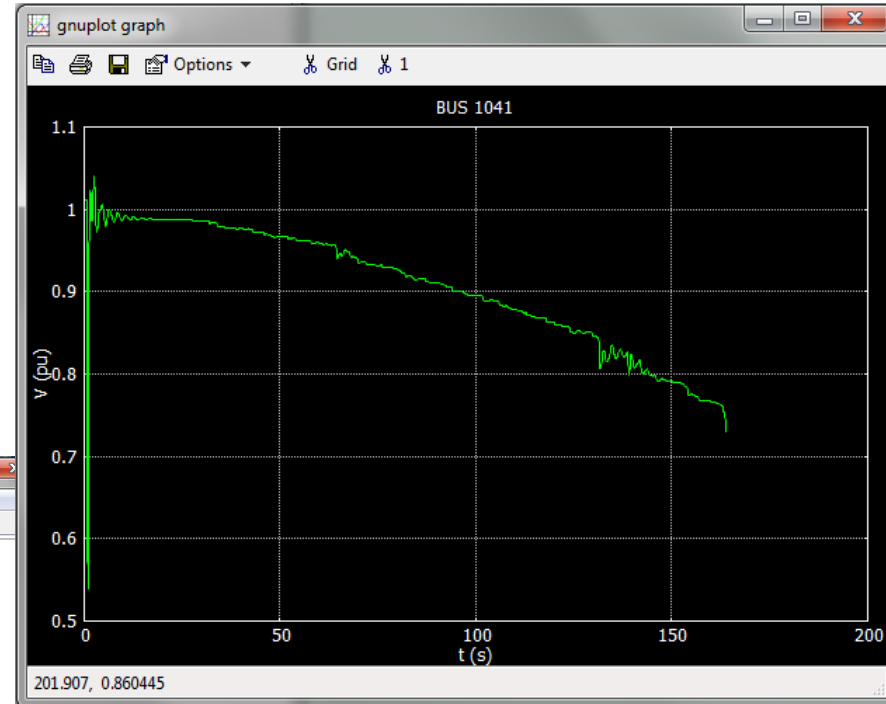
- wide range of platforms : from personal laptops to multi-core scientific computing equipment
  - Microsoft Windows OS : tested on Windows 7 and 10
  - Linux OS : tested on Debian, Ubuntu, Redhat

- interface with MATLAB
  - for faster prototyping of discrete-time controls (see slide # 9)
  - through the “MATLAB engine”
  - during the simulation RAMSES passes information to MATLAB workspace and receives control actions

Possible execution modes

- As a standalone program executed from command line terminal
  - for remote execution on systems without graphic interfaces
  - embedded in scripts as part of more complex procedures
- with a JAVA-based Graphic User Interface
  - for an easy-to-use and standalone execution
  - JAVA for compatibility with multiple platforms
  - single JAVA archive (.jar file) including all necessary executables and libraries to perform simulations and visualize results
    $\rightarrow$ software ready to be used with no installation !
- a dynamic library (.dll or .so)
  - to be linked to other software (e.g., written in C)
  - can be loaded within interpreted languages (e.g., MATLAB or Python)





screenshots of the
JAVA-based Graphic User Interface

Example : Hydro-Québec system - 30% motor loads (I)

Characteristics

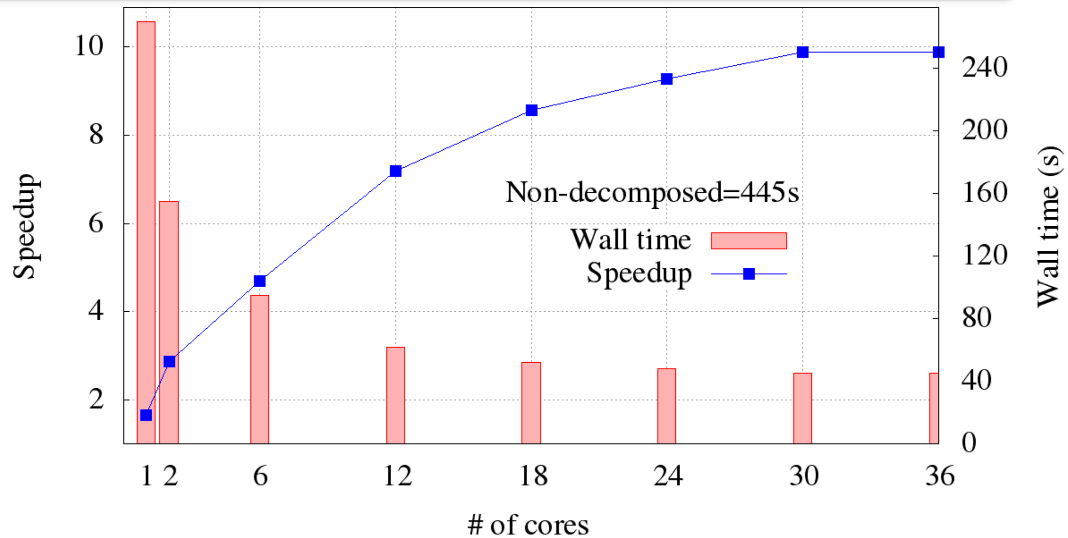
- 2565 buses
- 3225 branches
- 290 power plants
- 4311 dynamically modeled loads
- 506 impedance loads
- 1136 discrete devices
- 35559 differential-algebraic states

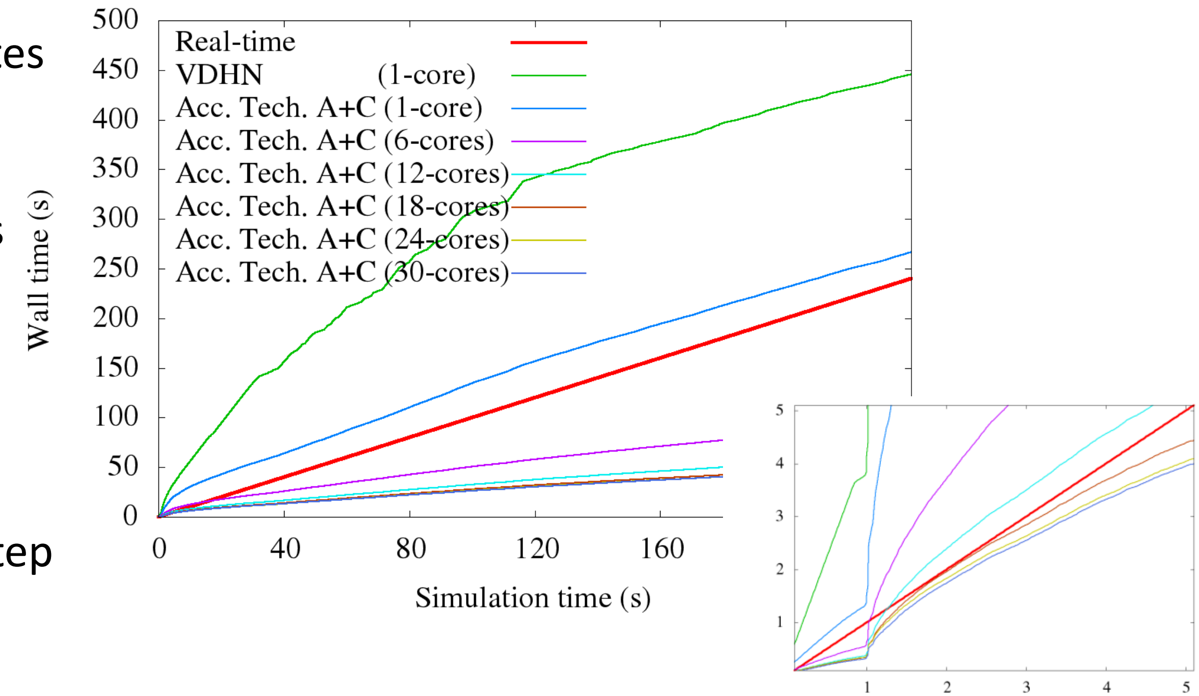
Disturbance

- short circuit lasting seven cycles
- cleared by opening one line

Simulation

- over 240 s
- with one-cycle (16.6 ms) time step
- ϵ_L=0.1MW/MVAr, T_{obs}=5 s





Example : Hydro-Québec system - 30% motor loads (II)

Characteristics

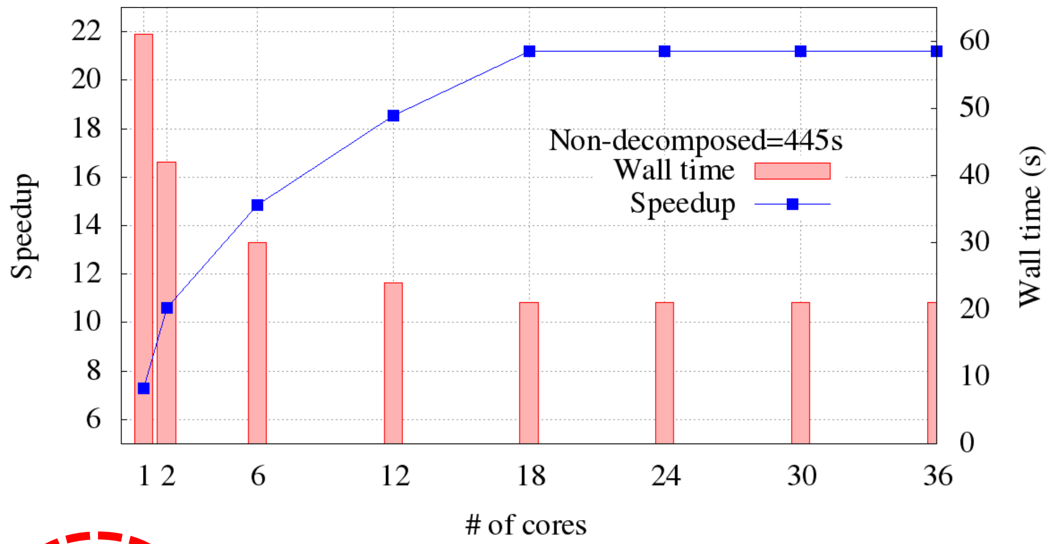
- same as previous slide

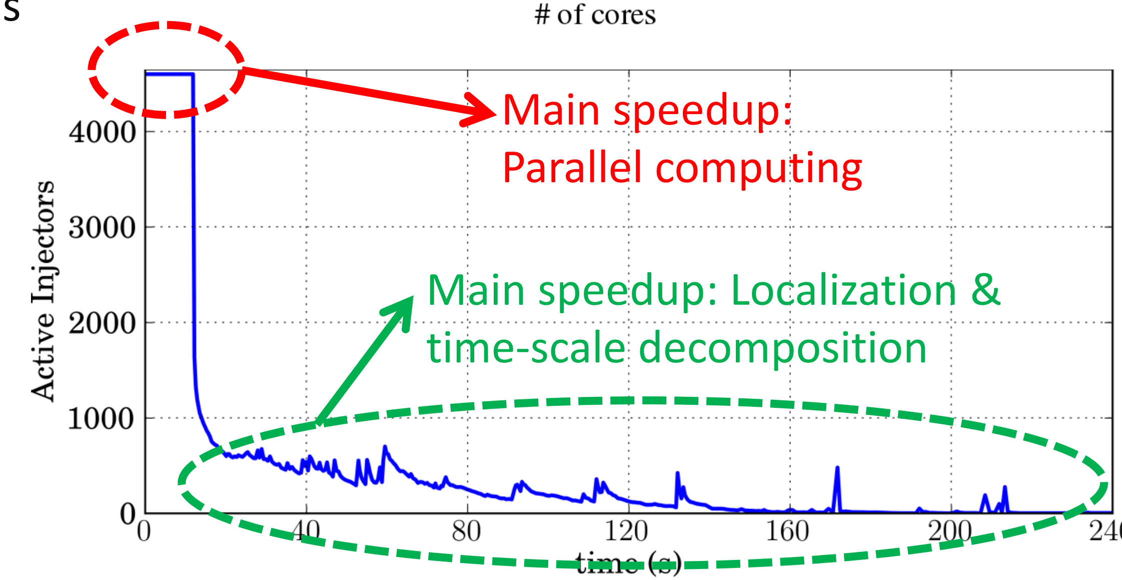
Disturbance

- same as previous slide

Simulation

- one-cycle time step for the first 15 s
- then 0.05 s for the remaining
- ϵ_L=0.1MW/MVAr, T_{obs}=5 s

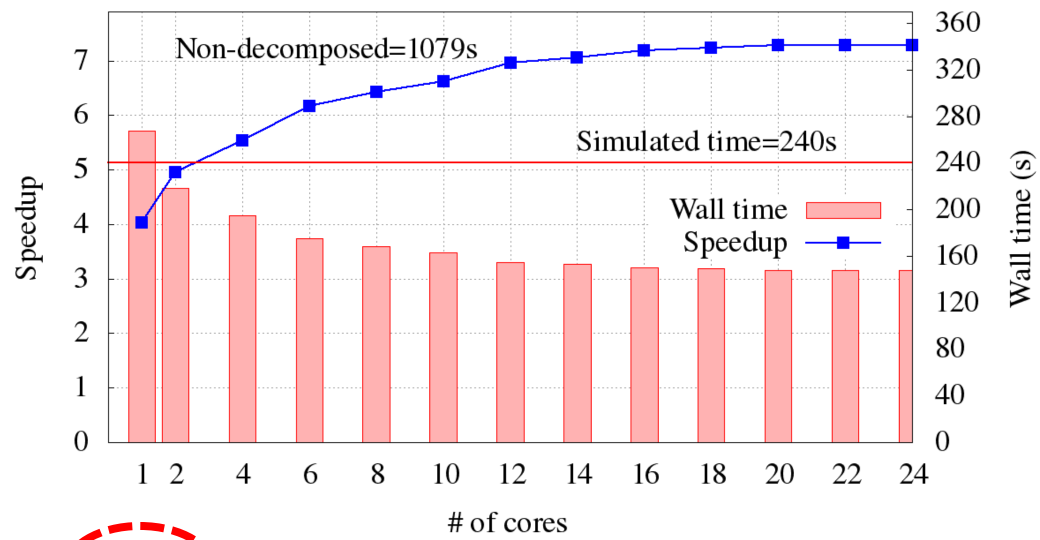




Example : PEGASE test system (I)

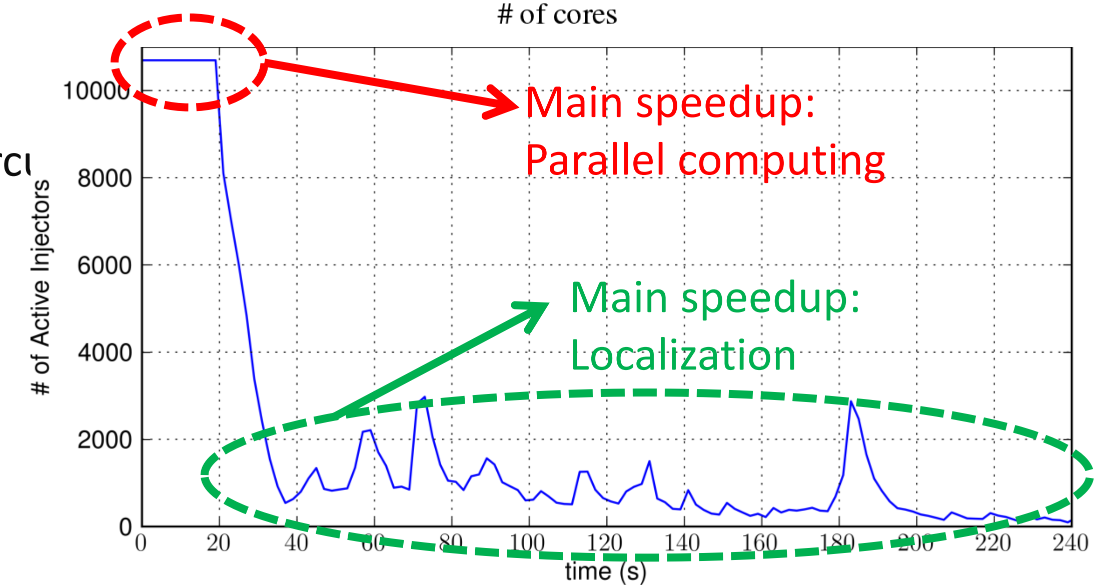
Characteristics

- 15226 buses
- 21765 branches
- 3483 power plants
- 7211 dynamically modeled loads
- 2945 discrete devices
- 146239 differential-algebraic states



Disturbance

- busbar fault lasting five cycles
- cleared by opening two double-circu



Simulation

- over 240 s
- with one-cycle (20 ms) time step
- ϵ_L=0.1MW/MVAr, T_{obs}=20 s

Example : PEGASE test system (II)

Characteristics

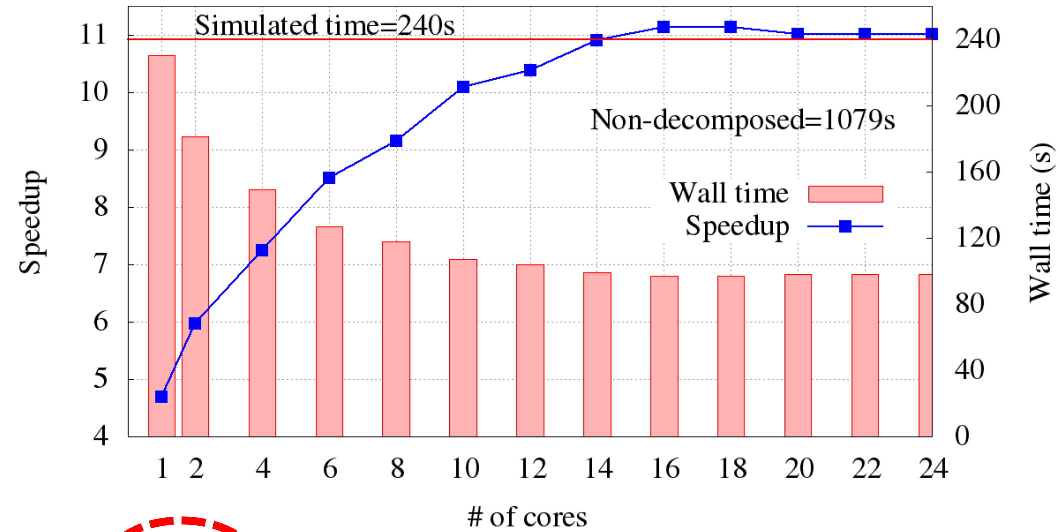
- same as previous slide

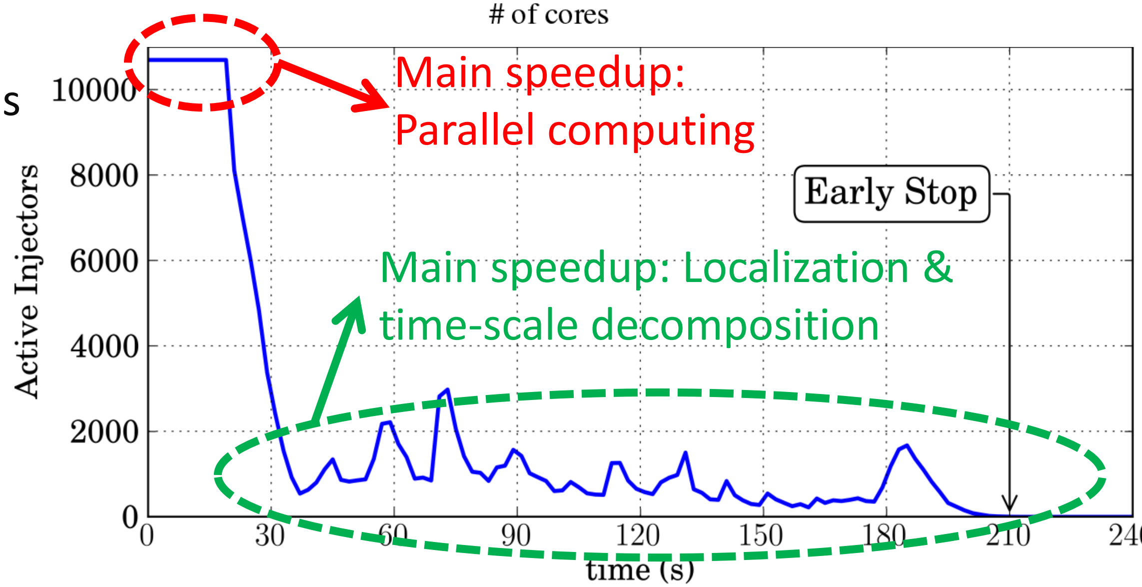
Disturbance

- same as previous slide

Simulation

- over 240 s
- one-cycle time step for the first 15 s
- then 0.05 s for the remaining
- ϵ_L=0.1MW/MVAr, T_{obs}=20 s





References

- D. Fabozzi, A. Chieh, B. Haut, and T. Van Cutsem, “Accelerated and localized newton schemes for faster dynamic simulation of large power systems,” IEEE Trans. on Power Systems, 2013.
- P. Aristidou, D. Fabozzi, and T. Van Cutsem, “Dynamic simulation of large-scale power systems using a parallel Schur-complement-based decomposition method,” IEEE Trans. on Parallel and Distributed Systems, 2014.
- P. Aristidou, S. Lebeau, T. Van Cutsem. Power System Dynamic Simulations Using a Parallel Two-Level Schur-Complement Decomposition. IEEE Transactions on Power Systems, 2016.
- P. Aristidou, S. Lebeau, L. Loud, T. Van Cutsem. Prospects of a new dynamic simulation software for real-time applications on the Hydro-Quebec system. CIGRE Science & Engineering, 2016.