“IEEE/Cigre Power System DLL Models/Standard”

WECC Meeting
April 5, 2019

Contact: Garth Irwin
gdi@electranix.com
1-204-953-1831
Presentation Outline

• Methods Currently Used to Develop Models (EMT and TS)
  • Generic Models
  • Custom Models
  • Real-Code/DLL Models
• Joint IEEE Task Force/Cigre Working Group for a DLL modeling standard
  • Concept
  • IEC coordination
  • Challenges
• Applications, Examples & Experience in EMT
• Possible Extension from EMT to Transient Stability Models
  • Differences from real-code EMT to TS models
  • Take-home items for suppliers
• Other Topics for Discussion
Methods Used to Develop Models

Generic Models
- Less accurate (inherently generic)
  - Testing only performed under select conditions
  - Are generic models useful/valid for weak systems, multiple devices nearby, series capacitors, ride-through testing...?
- Time consuming - every program developer (PSS/E, PSLF, PSCAD etc.) has to write/test models
- Suppliers are not happy:
  - Control concepts can be different from the real controls
- Often no protection models (or very simplified)
- Many cases have been identified where these models incorrectly predict stability, some of them resulting in serious events (eg. Blue Cut fire event)

- Should be easy to use and reliable
- Only settings should be required (does not require manufacturers to write models for N different programs)
Methods Used to Develop Models

Custom Models
- More accurate (but still not real-code so somewhat generic)
  - Testing only performed under select conditions
  - Algorithms often simplified
  - Time consuming/complex to develop - manufacturers have to write/test models for each device.
- Have a reputation for being slow, complex and not reliable
  - Often developed by suppliers, not program experts
- Binary compatibility and NDA/Intellectual Property concerns
  - Have to be re-compiled for new/major program releases (source code needed?)
- Higher confidence level from suppliers (ie better models)
Methods Used to Develop Models

Real-Code Models (EMT)
- Common for wind, solar, HVDC, VSC, SVC, Statcom, BES etc.
- Most accurate (actual code in the field running on your PC)
  - Minimal testing/validation required
- Developed with direct supplier support
- Fast to develop
- NDA/Intellectual Property concerns resolved (no source code of models needs to be supplied – DLLs only)

- Interfaces to DLLs may need to be re-compiled for new/major program releases (interface source code needed?)

- In the EMT world, these are now standard, and in many cases required around the world.
New Joint IEEE Task Force/Cigre Working Group

- IEEE Task Force "Use of Real-Code in EMT Models for Power System Analysis"
  - First meeting IEEE PES Meeting, Portland, Monday, August 6, 2018
  - Under AMPS Committee (Analytical Methods for Power Systems) and TASS Working Group (Transient Analysis and Simulation)
  - >50 people attended (in a room that seats 25) – popular!
  - Feedback says this should become an IEEE standard!
- Cigre B4.82 (Guidelines for Use of Real-Code in EMT Models for HVDC, FACTS and Inverter based generators in Power Systems Analysis)
- Coordination with IEC-61400-27-1
- Objectives:
  - Prepare guidelines/white-paper (standard?) on the techniques and methods for developing models.
  - This paper will provide guidance to manufacturers, utilities, consultants and system operators on the development and use of these models.
New DLL Modeling Concept

• Manufacturers compile their device source code into a DLL (conforming to the standard). The DLL standard includes:
  • Exported functions which can be called (on the initial time step and for each sample time)
  • Definition of all Inputs, Outputs and Parameters (including variable types, units, array dimensions, min and max allowable settings etc.)
  • Sample time step (at which to call the controls each step)
  • This DLL only needs to be updated every time the code is changed/released (version control!)
• Program Developers (PSS/E, PSLF, PSCAD etc.) include a DLLImport tool:
  • Run once by the end user for major program versions
  • Tool first opens/queries the DLL (to get Inputs, Outputs, Parameters, sample time etc...)
  • Tool creates any interfacing code for that particular program/version.
  • Tool may need to be re-run for each version update (but model source code not needed!). DLL can be used for all future program versions.
• End User:
  • Get the DLL from the manufacturer
  • Run the DLLImport tool once
Model Settings

• Getting the code is not enough – settings are equally as important!
• Settings (control and protection gains, settings, constants etc…) are often updated in the field.
• Identical settings in the field should be used in studies (and vice-versa)

• Each utility requires a database of all devices in operation:
  • Contains code name and revision number used in the field
  • End user can verify correct model and code revision is used
  • Contains parameter settings revision #:
    • CRC checks?

• End User gets the DLL/model name and settings version from the database
• The field code settings are directly used in simulation models
  • Same settings used in RMS and in EMT models
  • Studies engineers get notified if site settings are changed
    • Can decide if the setting change is important and can trigger re-studies
  • Real device gets study-qualified settings
Real-Code - Advantages

- Models use actual code – ie highest possible accuracy:
  - All control code and protection included
  - Integrations, time step delays in feedbacks etc... in models is the same as in the field
  - Same DLL used in all programs (EMT and TS)
- Manufacturer’s do not need to understand how to write complicated EMT or TS models. No interfacing knowledge required.
- DLLs will work with every current or future version of any program:
  - DLLImport tools maintained by the program developers
  - DLL will work with any current or future Fortran/C compiler
- Confidential Information is hidden (no source code released to others):
  - Possible concerns regarding reverse-engineering?
- Speed?
  - Possibly slower due to more “lines of code” and complex models
  - Possibly faster due to local solvers that can be threaded/parallelized
- Data/Model Parameters in the field same as in models
Real-Code - Challenges

- State variable storage (real-code does not need this)
  - Required for multiple instances (although DLL copying can avoid this) and snapshots (although new memory copy methods can avoid this).
- Initialization
  - Often not needed for real-code or in EMT analysis, but flat-start is required for TS
  - Requires a routine that can accept powerflow terminal conditions, and can compute all required state variables
- Different time frames for models
  - EMT programs often use all code (fast and slow) and have ABC quantities.
  - RMS programs cannot call fast code (constant current inner loops) and use RMS quantities only.
  - Multiple subroutine entry points (with different inputs and outputs) required.
- Speed
  - Each model is called once per step, taking inputs of V,I from last step
  - Internal solvers allow “threads” (parallel processing)
  - Use of real-code should make TS and EMT programs faster (not slower)
Applications

• Almost all devices now use digital code and processors:
  • Power Electronics:
    • Wind turbines, solar inverter, HVDC, SVC, Statcom, machine drives, battery energy storage inverters
  • Protection relays
  • Generator Controls?
    • Exciters, governors, stabilizers
  • RAS/SPS, wide-area controllers, PMU etc.
Example EMT Experience

• Interfacing to real-code done >40 times in EMT programs:
  • Wind farms
  • Solar/PV
  • SVC, Statcom
  • HVDC VSC/LCC
  • BES (battery energy storage)
  • Protection Relays
  • Real-Code is the most common type of model used!
• IEC-61400-27-1 (used by at least one wind turbine suppliers)
  • Similar concept to what is proposed (IEC/IEEE/Cigre collaboration is encouraged)
  • DLLImport tool written to automate the controller interface
  • Program interface/structures → Demo (with permission from IEC 61400)
EMT Lessons Learned

- Real-code DLLs are a fast and reliable method to develop complete models.
- Most accurate modeling possible (including protection etc).
- Program interface requires assistance by the program developers (to solve reliability/quality/usability problems).
- Source code stays with the manufacturer - can resolve NDA/IP concerns.
- Real-code studies often showing device tripping for normal faults.
- Many examples of instabilities (or ride-through failures) found that do not occur in RMS or simple/generic/manually constructed models.
- EMT Snapshot support:
  - Often on the “nice to have” list (and sometimes required) but not easily achievable
  - Can now be done without real-code changes (memory copying)
- Multiple Instances:
  - Can be done if state variables are grouped (rare)
  - Can be done via simple DLL copying method
- Manufacturers should work toward state variable grouping in their real code.
Real-Code – EMT vs TS

- Real-code models are common in EMT/PSCAD
  - Small electrical time steps
  - ABC measured V and I (similar to real controls – not RMS)
  - Inner current and fast controls (PWM firing, PLL, IGBTs etc.)
  - Identical code from the field, running on your PC

- Transient Stability Modeling?
  - Normally custom models are written specifically for an RMS program:
    - State/DState syntax, ABCD constant methods, Integrations/solver done centrally by the main algorithm
    - Not how real controller works! Time step delays and feedbacks should be modeled...
  - Real-Code in RMS Tools – Yes!
    - Internal solvers for all integrations (ie solved in the code, just like in real life)
    - More accurate!
Real-Code in TS Models

- Example of a TS Custom Program Written 5 ways (PSS/E example):
  1. Traditional method using formal State and DState PSS/E solvers
  2. Internal solver (backward Euler or other integration methods as required) - no formal STATE/DState variables • Uses VARS to store state variable information
  3. Same as above (internal solver) but code in a separate Fortran subroutine
  4. Same as above (internal solver) but code in a separate C subroutine (static linking)
  5. Same as above (internal solver) but code in a separate C subroutine (with a DLL loaded from Fortran)
# Real-Code in TS Models

<table>
<thead>
<tr>
<th>Feature</th>
<th>EMT</th>
<th>TS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming Interface</td>
<td>Fortran/C</td>
<td>Fortran/C (API coming soon in PSLF?)</td>
</tr>
<tr>
<td>Snapshots</td>
<td>Needs state storage (or new memory copying can be used).</td>
<td>Needs state storage.</td>
</tr>
<tr>
<td>Multiple Instances</td>
<td>Needs state storage (or a DLL copying to a unique name).</td>
<td>Needs state storage.</td>
</tr>
<tr>
<td>Input Entry Point</td>
<td>ABC domain (same DQ or rms computations as real-code).</td>
<td>RMS domain (so different input entry points needed directly into rms or DQ domain).</td>
</tr>
<tr>
<td>Connection to Other Models</td>
<td>Manually connected.</td>
<td>Needs to be automatically connected to other models.**</td>
</tr>
</tbody>
</table>
Real-Code in TS Models - Other

- **Challenges for TS DLL Standard:**
  - Need to define standard model types and inputs/outputs/connectivity?
  - Parallel Processing (threading)
    - All dynamic models are solved using internal model code (not solved by the main program) so can be easily threaded onto multiple cores to speed up
  - Expectations:
    - More complex modeling (a fact of life with wind, solar, BES, HVDC, SVC, Statcoms, weak systems, series capacitors etc...)

- Take-Home to Manufacturers and Program Developers
  - Modify real-code:
    - Identify and group state variables
    - Add initialization routines for all state variables
    - Modularize code to support both RMS/DQ and ABC input quantities

- Why is this not going to be “user defined model hell” all over again?
Other Topics for Discussion

- Hybrid Simulation Tools
  - PSS/E to PSCAD (and PSS/E to PSS/E)
  - PSLF to PSCAD (in progress – now working for N single port)
    - Also provides PSLF to PSS/E?
- Parallel Processing
- Grid-Forming (or Voltage Source) VSC Controls
"Use of Real-Code in EMT Models for Power System Analysis"

Contact Information

Please contact me below to volunteer specific contributions towards the guidelines paper.


Contact:  Garth Irwin
Electranix Corporation
1-204-953-1831
[ gdi@electranix.com](mailto:gdi@electranix.com)