"Use of Real-Code in EMT Models for Power System Analysis"

IEEE PES Meeting, Atlanta, Wednesday August 7, 2019

This is a Task Force under:

- ***AMPS Committee (Analytical Methods for Power Systems)***
- ***TASS Working Group (Transient Analysis and Simulation)***

Contacts: Garth Irwin
gdi@electranix.com
1-204-953-1831

Deepak Ramasubramanian
dramasubramanian@epri.com

Meeting Agenda

- Joint IEEE Task Force/Cigre Working Group for a DLL modeling standard
  - Concept
  - Challenges
- Presentations (10 minutes each!):
  - Real-Code Controllers – EMT vs TS Equation Solvers
    - Garth Irwin (Electranix)
  - Experience in EMT Real-Code Controller Interfaces
    - Jean Mahseredjian (Professeur at Polytechnique Montréal)
  - Use of STATCOM Real-Code in EMT and Transient Stability Models
    - Bilgehan Donmez (AMSC)
- Other Topics for Discussion

DLL Standard Modeling Concept

1. Manufacturers compile their device source code with DLL wrapper (the IEEE/Cigre wrapper largely based on IEC DLL standard)
2. Program Developers (PSS/E, PSLF, PSCAD, EMTP/RV etc.) include a DLLImport tool:
   - Creates all interface code, state variable storage etc.. for that program.
3. End User:
   - Get the DLL from the manufacturer
   - Run the DLLImport tool once

DLLImport tool written for PSCAD and PSS/E. Similar IEC standard interfaces available for Matlab, DigSilent and EMTP/RV. Applications for wind, solar, HVDC, SVC/Statcom, BES, exciters/governors/stabilizers, protection relays…

Biggest Challenge:

- Manufacturers must write initialization subroutines for their code (not strictly necessary for EMT, but essential for TS programs).

DLL Standard - Routines

Routines Called from DLLImport Tool to DLL Model:

StaticExtSimEnvCapi Model_GetInfo();

```
typedef struct _StaticExtSimEnvCapi
{
…
   const char_T * const                ModelName;               // Model name
   const char_T * const                ModelVersion;            // Model version
   const char_T * const                ModelDescription;        // Model description
   const char_T * const                VersionControlInfo;      // Version control info
   …
   const real64_T                      FixedStepBaseSampleTime; // Base sample time
   const int32_T                       NumInputPorts;           // Number of inputs
   const StaticESEInputSignal * const  InputPortsInfo;          // input description array
   const int32_T                       NumOutputPorts;          // Number of outputs
   const StaticESEOutputSignal * const OutputPortsInfo;         // output description array
   const int32_T                       NumParameters;           // Number of parameters
   const StaticESEParameter * const    ParametersInfo;          // parameter description array
   const int32_T                       NumContStates;           // Number of continuous states
   const int32_T                       SizeofMiscStates;        // Size of work variables / misc states
   const char_T                        *LastErrorMessage;       // Error string pointer
   const uint8_T                       EMT_RMS_Mode;            // Mode: EMT = 1, RMS = 2, EMT & RMS = 3
   const int32_T                       NumIntStates;            // Number of Integer states
   const int32_T                       NumFloatStates;          // Number of Float states
   const int32_T                       NumDoubleStates;         // Number of Double states
}StaticExtSimEnvCapi;
```

DLL Standard - Routines

Inputs:

```
typedef struct _StaticESEInputSignal
{
   const char_T * const          Name;        // Input signal name
   const char_T * const          BlockPath;   // Path to block in Simulink model
   const int32_T                 Width;       // Signal array dimension
   const char_T * const          Unit;        // Units
   const int32_T                 InputType;   // Vector of Integer Types (see full list)
}StaticESEInputSignal;
```

Outputs:

```
typedef struct _StaticESEOutputSignal
{
   const char_T * const          Name;         // Output signal name
   const char_T * const          BlockPath;    // Path to block in Simulink model
   const int32_T                 Width;        // Signal array dimension
   const char_T * const          Unit;         // Units
   const int32_T                 OutputType;   // Integer Output Signal Type (see full list)
}StaticESEOutputSignal;
```

DLL Standard - Routines

Parameters:

```
/* Static parameter information; only scalar parameters are allowed */
typedef struct _StaticESEParameter
{
   const char_T * const       Name;          // Parameter name
   const char_T * const       Description;   // Description
   const char_T * const       Unit;          // Unit
   const real64_T             DefaultValue;  // Default value
   const real64_T             MinValue;      // Minimum value
   const real64_T             MaxValue;      // Maximum value
   const int32_T              Type;          // 0 for parameters which can be modified at any time, 1 for parameters
which need to be defined at T0 but cannot be changed.
}StaticESEParameter;
```

DLL Standard - Routines

Routines Called from Program to DLL Model:

```
pInstanceCapi Model_Instance();
Model_CheckParameters  (*pInstanceCapi);
Model_Initialize       (*pInstanceCapi);
Model_Outputs          (*pInstanceCapi, …);
Model_Terminate        (*pInstanceCapi);
*/
```

DLL Standard - Routines

Data Exchange Object:

/* Instance specific model information */

```
typedef struct _InstanceExtSimEnvCapi
{
  real64_T          *ExternalInputs;       // Input signals, all elements in one long vector
  real64_T          *ExternalOutputs;      // Output signals, all elements in one long vector
  real64_T          *Parameters;           // Parameters as vector
  real64_T          *ContinuousStates;     // We assume a states vector - not used in current version
  real64_T          *StateDerivatives;     // We assume a states derivatives vector - not used
  uint8_T           *MiscStates;           // Work variables / states with unknown content
  const char_T      *LastErrorMessage;     // Error string pointer
  const char_T      *LastGeneralMessage;   // General message
  uint8_T           VerboseLevel;          // Decides how much the code "should talk"
  int32_T           *IntStates;            // Int State vector
  real32_T          *FloatStates;          // Float State vector
  real64_T          *DoubleStates;         // Double State vector
  ESEExtension      Extension;             // Provided for extensions
}InstanceExtSimEnvCapi;
```

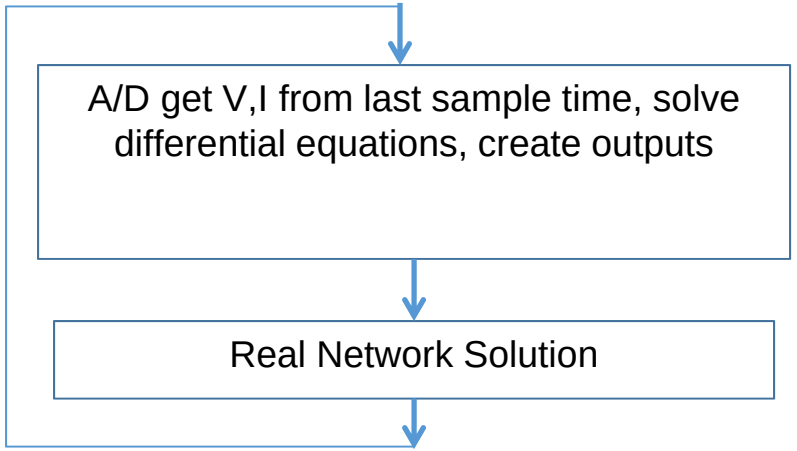
DLL Standard - Key Advantages

- DLL can be used in any program, version, compiler
  - Future proofing
  - This is not “Custom Model Hell” all over again!
- Source code stays protected
- Same results in each program
- High accuracy in models (ie real-code)
- Everyone does what they are good at:
  - Manufacturers write model code
  - Program developers write interface code

“Real-Code Controllers” - EMT vs TS Solvers

Real-Code Controllers:

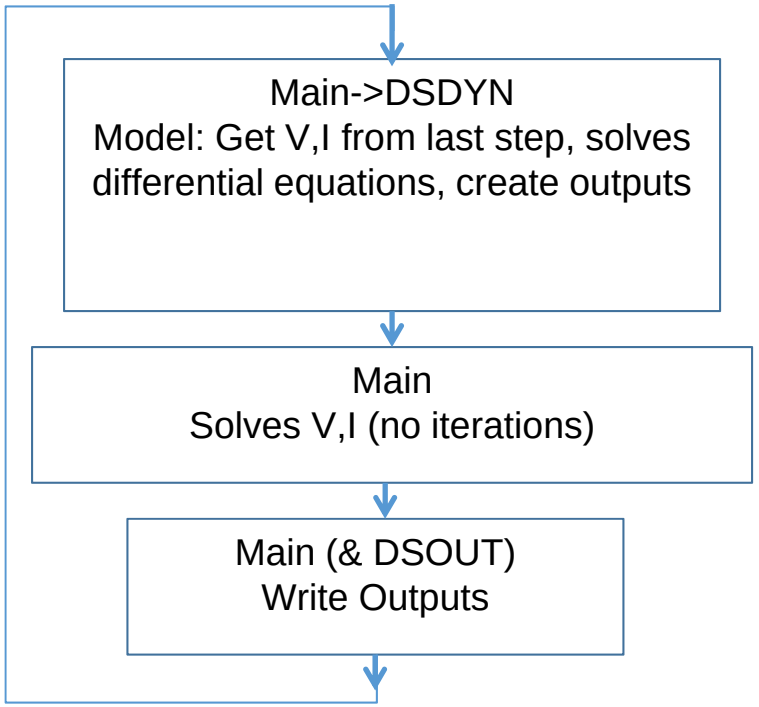
- CT/PTs->A/D
- Control sampling time
- Code solves differential equations
- Outputs



EMT Programs

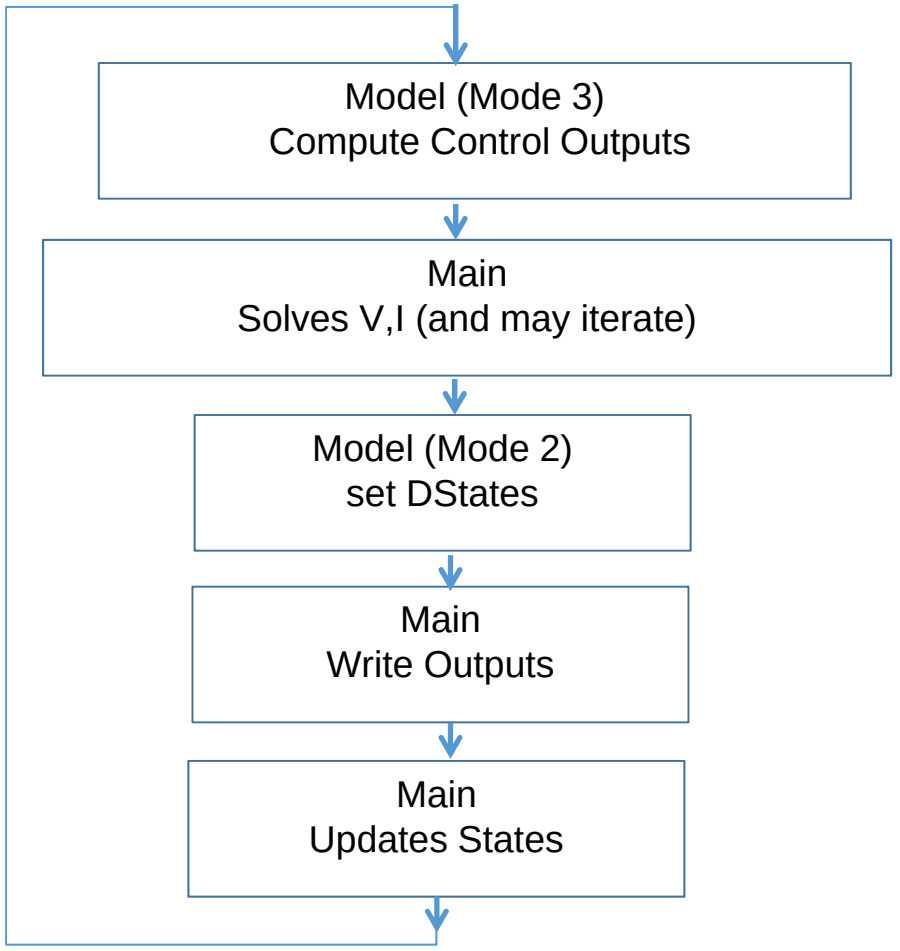
- Measurements of V/I (from last time step)
- Controller sampling time (may differ from EMT simulation step)
- Each model solves its own differential equations directly
- Create Outputs

- No central control solvers included in EMT



Transient Stability Programs

- Measurements of V/I with a fixed controller sampling time (same as the simulation step)
- Code identifying States and sets DStates
- Main program solves for States
- Back to code/model to update outputs

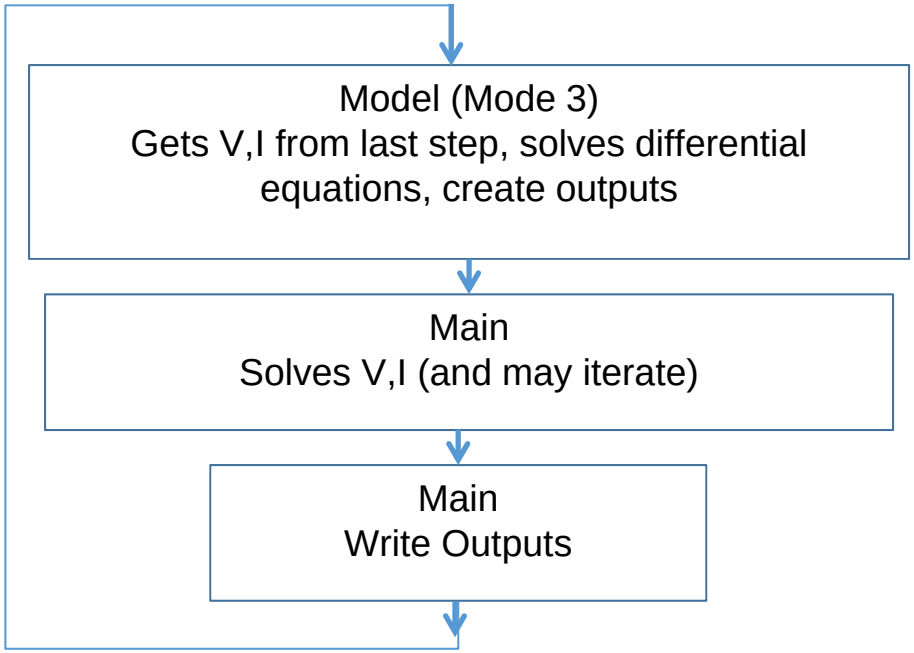


“Transient Stability – proposed method using real-code with control equations solved in the model code”

Propose models solve their own differential equations in model code directly.

Results are 100.0% identical to traditional TS models (if the same integration method is used, and if the controller sample time is the same as the simulation step).

Note: Dual time step can average States during a switch can result in a difference for 1 step.



Why (solve control equations in the model code)?

- Absolutely necessary for Real-code
  - Real-code solves differential equations in its code
- Same answer on all programs
- Can be achieved in all programs (EMT and TS programs)

Is this task force trying to develop a standard open-systems simulation platform? Or is it a standard to interface to real-code controllers?

- Model_Derivatives (set DStates) in standard?
- Model_Outputs vs Model_Update in standard?

Next Presentation:

"Use of Real-Code in EMT Models for Power System Analysis"

Project www site: www.electranix.com/IEEE-PES-TASS-realcodewg

Other topics:

- FMI starting point (alternative to DLL method)
- Group message/collaboration service? File Repository? IEEE Imeet?
- DLL Support in other simulation programs?
- Can same code be compiled to a shared DLL (for real-time simulators/Unix)?

Next Meeting: Cigre B4.82, Johannesburg – September 28/29
Webex meetings?

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

Deepak Ramasubramanian
dramasubramanian@epri.com