Standardized NPSS Propulsion Model integration into Simulink process using FMI

MATLAB EXPO 2023

Keith Pattison, Associate Fellow Propulsion Modeling and Simulation
Lockheed Martin Aeronautics – Fort Worth Tx

© 2023 Lockheed Martin Corporation, All Rights Reserved
Lockheed Martin Aeronautics: Supporting Legacy Aircraft and ADP: Advanced Development Programs

- Next Gen Mobility
- Future Fighter Force
- Derivatives
- Quiet Super Sonic Demonstrator
- High Speed Systems
- Next Gen UAS
- Hybrid
- High Energy Weapons
- Compact Fusion
- High Energy Weapons
**Introduction**

- Integrated Flight and Propulsion Controls has become much more important as aircraft systems become more complex.

- NPSS is a NASA/Industry developed tool that provides an object orientated simulation environment for propulsion performance modeling.

- Utilization of High-Fidelity plant and engine control models, from our suppliers, packaged in NPSS environment is standard practice on almost all Lockheed Martin programs.

- NPSS Engine models are full physics plant model of the engine system, coupled with a representative Engine Control model FADEC(Full Authority Digital Engine Control) to provide installed engine flight performance through out the flight envelope.

- Engine Simulations are utilized at all stages program development from Conceptual design to end of life. They are frequently updated as the program matures, or the control logic is updated.
Background

• Legacy Engine model integration used to be done on a case by case basis with custom interface code developed for each program.

• Usually needed custom interfacing software (Middleware) to hook into Commercial tools like MATLAB/Simulink

• Over the last 10 years utilized Simulink with legacy interface with numerous issues. Required a number of tools to be aligned, NPSS version, Compilers, MATLAB versions. Many stability issues in the interface resulting in numerous hours of trouble shooting.

• FMI / FMU approach removes the middleware layer support from the model interface and places it on the host tool to accommodate.

FMI / FMU approach saves integration time and provides more system stability and common interface to many tools
NPSS FMU Export Development & implementation challenges

- In 2022 Lockheed Martin worked with the NPSS Consortium development team to test and refine the FMI version 2 FMU export capability. Lockheed Martin also worked with MathWorks SMEs to evaluate the Simulink interface to the new NPSS FMUs to develop several features of the NPSS FMUs.

- NPSS will have FMI 2 (CS & ME) import/export capabilities in upcoming version 3.3 being released later this year.
  
  - Released as EMI version 15.0 as part of the 3.3 commercial release.
  - Most features shown in this presentation available.

- For NPSS model Export Currently using a Python3 script to package an NPSS model into an FMU, it uses standard python packages with no dependencies. Desire was to have a simple wrapping process without any other tool or package dependencies.

  - Take advantage of FMU Multi OS Capability with Windows/Linux.
  - Eliminate Software compilation as part of the wrapping process.
  - Build Interface to be real time fast, no I/O slowdowns.

FMUs will feed real time simulation environments and need to be as fast as possible.
NPSS FMU Capability implementation

- NPSS FMU packaging script maps variables to 2 files. This is setup in a .config file as shown below:

```
# Fields: FM name, NPSS name (optional), Data type, causality, variability, units, Description
# These are the inputs
variable: time, real, independent, continuous, sec, Simulation time
variable: SimMode, real, input, continuous, none, Steady State = 1 Transient = 3
variable: DataRec, real, input, continuous, none, Off = 0 on
# AtmoIn Input Data
variable: AtmoIn.Wind.alt_in, real, input, continuous, ft, Pressure Altitude
variable: AtmoIn.Wind.MM_in, real, input, continuous, none, Mach Number
variable: AtmoIn.Wind.Ps_in, real, input, continuous, psia, Ambient pressure input
variable: AtmoIn.Wind.Ts_in, real, input, continuous, °F, Ambient temperature
variable: AtmoIn.Wind.switchDay.intValue, real, input, continuous, none, AS210 Day Type 0: Std 1: Trop -1
variable: EI.PLA_in, real, input, continuous, deg, Power 10=GI 15=FT 10=Nil 15=MaxAB
variable: WBI_in, real, input, continuous, lbm/sec, Mid Compressor Bleed input
variable: WRO_in, real, input, continuous, lbm/sec, Comp Discharge Bleed input
variable: HPX_in, real, input, continuous, hp, HP Extraction input
# Basic Global outputs
variable: FG, real, output, continuous, lbf, Gross Thrust
variable: TM00, real, output, continuous, lbf, Net Propulsion Force
variable: WFT, real, output, continuous, lbm/sec, Fuel Flow pound mass per sec
variable: PLA, FDMK, real, output, continuous, deg, PLA Feedback
variable: ChlECs.W, real, output, continuous, lbm/sec, Bleed Flow Output
variable: ChlECs.Pt, real, output, continuous, lbf, Bleed Pressure
variable: ChlECs.Tt, real, output, continuous, °F, Bleed Temperature
variable: SHI, HPX, real, output, continuous, hp, High Compressor Shaft Extraction
```

ModelDescription.XML

```
<Variables>
  <ScalarVariable name="time" valueReference="1" description="Simulation time" variability="continuous" causality="independent">
    <Real units="sec"/>
  </ScalarVariable>
  <ScalarVariable name="SimMode" valueReference="2" description="Steady State = 1 Transient = 3" variability="continuous" causality="input">
    <Real units="none" start="0"/>
  </ScalarVariable>
  <ScalarVariable name="DataRec" valueReference="3" description="Off = 0 on" variability="continuous" causality="input">
    <Real units="none" start="0"/>
  </ScalarVariable>
  <ScalarVariable name="AtmoIn.Wind.alt_in" valueReference="4" description="Pressure Altitude" variability="continuous" causality="input">
    <Real units="ft" start="0"/>
  </ScalarVariable>
  <ScalarVariable name="AtmoIn.Wind.MM_in" valueReference="5" description="Mach Number" variability="continuous" causality="input">
    <Real units="none" start="0"/>
  </ScalarVariable>
  <ScalarVariable name="AtmoIn.Wind.Ps_in" valueReference="6" description="Ambient pressure input" variability="continuous" causality="input">
    <Real units="psia" start="0"/>
  </ScalarVariable>
  <ScalarVariable name="AtmoIn.Wind.Ts_in" valueReference="7" description="Ambient temperature" variability="continuous" causality="input">
    <Real units="°F" start="0"/>
  </ScalarVariable>
  <ScalarVariable name="AtmoIn.Wind.switchDay.intValue" valueReference="8" description="AS210 Day Type 0: Std 1: Trop" variability="continuous" causality="input">
    <Real units="none" start="0"/>
  </ScalarVariable>
  <ScalarVariable name="EI.PLA_in" valueReference="9" description="Power 10=GI 15=FT 10=Nil 15=MaxAB" variability="continuous" causality="input">
    <Real units="lbf" start="0"/>
  </ScalarVariable>
  <ScalarVariable name="WBI_in" valueReference="10" description="Mid Compressor Bleed input" variability="continuous" causality="input">
    <Real units="lbm/sec" start="0"/>
  </ScalarVariable>
  <ScalarVariable name="WRO_in" valueReference="11" description="Comp Discharge Bleed input" variability="continuous" causality="input">
    <Real units="lbm/sec" start="0"/>
  </ScalarVariable>
  <ScalarVariable name="HPX_in" valueReference="12" description="HP Extraction input" variability="continuous" causality="input">
    <Real units="hp" start="0"/>
  </ScalarVariable>
</Variables>
```

-variable naming can be adapted to meet requirements of both the FMI standard and NPSS

Sim Host Calling FMU

Handles Naming to the Calling Host per FMI standard

ModelDescription.XML

npss_fmi.json

Handles Naming to the NPSS model thru parallel I/O arrays

Using 4868 Datalist I/O to NPSS model for Speed

ARP 4868 C API
NPSS FMU Capability implementation

- Initialization and loading of the NPSS model needed to be scheduled and accommodated several ways due to the way various host programs set initial values and run at time 0.

- NPSS model needs to be loaded (c4868init) in initialization mode prior to setReal() being called.

- Prefer setting input start values at runtime, not using FMU variable StartValue input.

- FMU logging and ARP 4868 Debug dump switchable on/off in FMU config file. Provides debugging capabilities for the FMU execution, but slows down runtime.
NPSS FMU Capability implementation

- Variable naming and issues with NPSS and FMI were first issue
  - NPSS is object oriented names can have C++ style object scope: Amb.alt_in, EI.PL_A.in, TrbH.S_map.effDes
  - NPSS objects Contain both Inputs / Output variables
  - Various host programs treat the . Differently for FMU I/O variables
    - FMPy treats . Blocks as an array
    - Simulink treats as a Bus Object with the dot separating multi level bus signals
    - Bus feature is good however need to map Inputs to one bus and outputs to another so NPSS for example:
      - Amb.alt_in (Input) Amb.alt(Output) need to be in separate Bus blocks

A Common approach was developed to handle issue in a number of tools allowing the Simulink Bus object to be utilized for large I/O blocks
NPSS FMU Execution Demo
NPSS FMU in Simulink Benefits and Features

- Integration time of NPSS model greatly reduced
- MathWorks provided preview of Simulink feature that auto codes the interface to the FMU using the Embedded Coder. Was able to autogenerate the interface code about the TF02 FMU.
- FMU tracks other information about the variables,
  - Units attribute can be set, but some differences in the string syntax as shown to the right.
  - FMI / NPSS / Simulink have some minor differences in unit string definitions, but can be handled in the Model description.XML interface.
  - Description also a useful attribute that can transfer over from the NPSS model.
Closing remarks

- FMI seems to be making a big impact in the simulation interoperability world. Numerical simulations like NPSS are easily packaged into FMUs and provide interoperability with a number of Toolsets.
- Need to evaluate FMI 3 standard and its impacts on all interfaces.
- Desire for variable timestep capability supposed supported in FMI 3.
- FMU packaged model provides an easier package to configuration manage, deploy and track with PLM tools.
- FMUs also easy to execute thru python scripts allowing automated regression testing and PLM requirements testing.
- For non Consortium members Contact the NPSS Consortium for further information.