

Jason Bush

MSFC/EV41/TriVector

September 14, 2010



Integrating Generated Code with NASA's MAVERIC Flight Vehicle Simulation



◆ Aerospace Engineer with TriVector Services, Inc.

- Support the Control Design and Analysis Branch of the Flight Mechanics Division at NASA - Marshall Space Flight Center (MSFC)
 - Majority of work is in Ares I Thrust Vector Control (TVC) time domain analysis in MAVERIC
 - modeling, simulation and analysis
 - implementation of delivered models
 - Beginning to support efforts in Ares I flexible vehicle modeling in MAVERIC
- Education
 - BSE , Aerospace Concentration, University of Alabama in Huntsville , 2009
 - Current – MS, Aerospace (Dynamics & Controls)

◆ Not a developer by any stretch of the word, more of a user becoming a developer

◆ MAVERIC

- history
- current uses
- formatting

◆ Integration of generated code

- custom target file for MAVERIC (maveric.tlc)
- wrapper

◆ Models tested thus far

- actuator
- auxiliary power unit (APU)
- pressure estimator

◆ Observations thus far on Real Time Workshop Embedded Coder generated code

◆ Future work

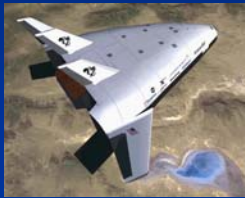
◆ Conclusions



MAVERIC

MARSHALL AEROSPACE VEHICLE REPRESENTATION IN "C"

Existing MAVERIC Simulations



X-33
Lockheed Martin
Skunk Works



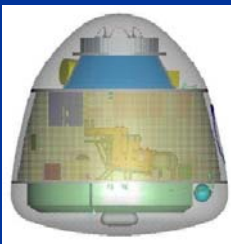
Shuttle-Derived Cargo
MSFC VIPA Activity

Saturn V
MSFC VIPA
Activity



X-43C
Orbital
Sciences

X-37
Boeing

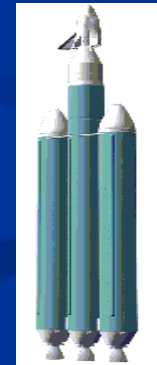


OSP
Lockheed
Martin



Ares I and Ares V

OSP
Boeing



Others: TSTO (LM), UAH 2-stage, Generic 2-stage (ITAGCT)

MAVERIC History



- ◆ An X33 simulation was constructed in 1996 for the X33 program. We named this program MAVERIC (Marshall Aerospace VEHICLE Representation in C) .
- ◆ By the time X33 was cancelled (2000), MAVERIC had developed into a high-fidelity simulation that was used for X33 flight performance assessments.
- ◆ The X33 MAVERIC version – although very useful for X33 - lacked generality which made it difficult to adapt to different vehicle configurations and mission scenarios and add new functionality.
- ◆ When the SLI (Space Launch Initiative) program was beginning, it was realized that MAVERIC would not be an adequate tool to support the many vehicle concept studies that would follow.
- ◆ In 2001, A “clean-sheet” approach was taken to design a 2nd generation version of MAVERIC (MAVERIC-II) that would possess the qualities of generality and modularity that were needed for quickly developing sims for different vehicle configurations and mission scenarios.
- ◆ Although SLI was soon cancelled, MAVERIC-II developed into a simulation tool that was able to support TSTO, OSP and CLV design efforts.
- ◆ Now that the X33 MAVERIC version is obsolete, the “-II” has been dropped from “MAVERIC-II” and this simulation tool is referred to as just “MAVERIC”.
- ◆ MAVERIC’s continued development is driven by the evolving requirements of the ARES I launch vehicle development program.
- ◆ A GUI interface has been development, in house, that greatly enhances MAVERIC’s usability.

- ◆ **The Constellation Program was developed in response to the Vision for Space Exploration laid out by President Bush in 2005 to send astronauts to the International Space Station, the moon and then Mars**
 - Ares I is to be used to launch the crewed Orion spacecraft into orbit
 - Ares V is to be used for heavy lift operations including delivering the Earth Departure Stage and the Altair Lunar Lander to orbit for rendezvous with Orion

- ◆ **MAVERIC is the primary simulation environment for assessment of the Constellation Program's Ares vehicles time domain performance**
- ◆ **The Ares MAVERIC simulations contain all of the flight control (FC) models that simulate the GN&C functions of the vehicle**
 - Any FC design changes are tested in MAVERIC Monte Carlo time domain analysis
- ◆ **The FC design will be taken out of MAVERIC and translated into onboard flight code for Ares**

◆ **MSFC developed, mostly C language with some C++ and FORTRAN**

- Capable of both 3-DOF and 6-DOF flight simulations of aerospace vehicles
- Executes in a UNIX environment
- Relatively fast
- Monte Carlo analysis is a strength

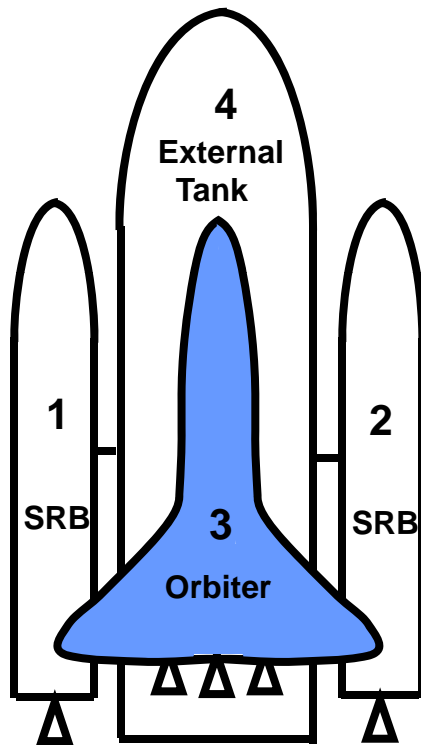
◆ **Mathematical models**

- Natural environments
 - gravity
 - atmosphere
- Vehicle properties
 - vehicle configuration
 - mass properties
 - propulsive devices
 - Thrusters
 - Engines – solid and liquid
 - actuators
 - nonlinear full-envelope aerodynamics
 - thruster interactions
 - static aeroelastic effects
 - slosh
 - structural dynamics
 - flight controls



- ◆ **Plug in feature allows for integration of different models into current architecture without modifications to existing models**
 - this feature has been abundantly helpful with the integration of the generated code
- ◆ **Formatting**
 - structure hierarchy
 - next slide
 - data input
 - data types
 - dispersions

A MAVERIC simulation contains:



Vehicles

An entity whose motion may be propagated consisting of one or more stages.

Stages

Elements that are combined to make up a vehicle to be propagated. A stage can have engines, tanks, sensors, cargo, thrusters, actuators

Cargo

Masses to be moved (retrieved/jettisoned) to or from stages. (ice, crew, payloads, shrouds, etc).

Engines

Propulsive devices.

Tanks

Contain propellant to feed engines and RCS thrusters

Sensors

Located on stages to provide sensed indications of position, velocity, attitude, body rates, etc. (GPS/INS, accelerometers, rate gyros, etc).

RCS Thrusters

Located on stages to provide attitude control for 6-DOF simulations.

Docking ports

Located on stages (under development)

Handling input data in MAVERIC



When a data file contains data such as:

```
engine_location 102.5 2.5 1.2 [N,xyz, sd=.34]
```

and the file is opened with:

```
data_source ( "propMean.dat");
```

The entire file (except for comments) is read into one long string, **input_stream**.

When the following line of code:

```
readn_v ( "engine_location", location, echo );
```

Is finally executed as:

```
location = scan_v ( "engine_location", &location, echo)
```

The scan routine starts searching through the **input_stream** string (token-by-token) until it finds the string

```
engine_location
```

Then the next 3 tokens are read, decoding them into double variables for the vector components.

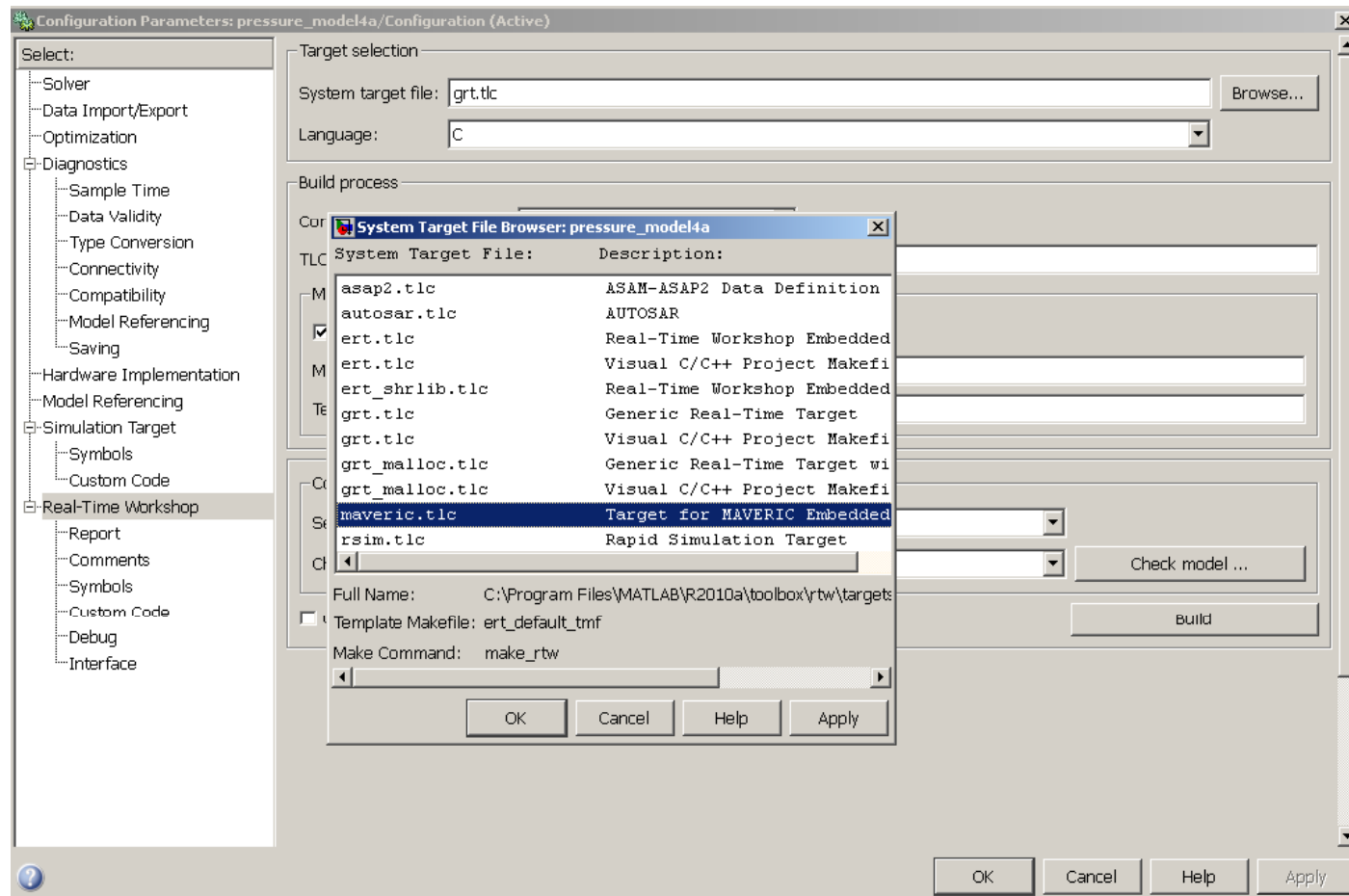
If the next token contains a "[", then this token is a Dispersion Specification which is decoded and saved in a dispersion data structure. If white space was put between the parts of the dispersion specification then, the scan routine continues to read tokens and concatenating them until a token containing a "]" is found.

- ◆ **Custom target file configures a wrapper for the generated code to match the format used in MAVERIC**
 - Utilizes existing MAVERIC structure hierarchy
 - Pointers to proper structures for hardware interactions
 - Defines the correct data files related to the components
 - Formats functions into MAVERIC general format
 - Follows the executive format
 - read()
 - init()
 - update()
 - uses #define that are needed to read data and apply Monte Carlo dispersions to variables
 - Contains common components from MAVERIC as well as an option for custom components that can be configured by the user
 - Actuator, Engine, Thruster, Prop tank , Sensor, FSW, etc.
- ◆ **Creates a section for custom code to be entered if needed in the update function**

Custom target file for MAVERIC cont.



- ◆ The MAVERIC custom target file 'maveric.tlc' is selectable from a drop-down menu in the configuration properties dialog box



Custom target file for MAVERIC cont.



- ◆ The 'maveric.tlc' custom target file allows the user to select the proper MAVERIC component for the model

Configuration Parameters: EM001408_1APU/Configuration (Active)

Select:

- Solver
- Data Import/Export
- Optimization
- Diagnostics
 - Sample Time
 - Data Validity
 - Type Conversion
 - Connectivity
 - Compatibility
 - Model Referencing
 - Saving
- Hardware Implementation
- Model Referencing
- Simulation Target
 - Symbols
 - Custom Code
- Real-Time Workshop**
 - Report
 - Comments
 - Symbols
 - Custom Code
 - Debug
 - Interface
 - SIL and PIL Verification
 - Code Style
 - Templates
 - Code Placement
 - Data Type Replacement
 - Memory Sections
 - MAVERIC Target Code ge...

Target selection

System target file:

Language:

Description: Target for MAVERIC Embedded Coder

Build process

TLC options:

Makefile configuration

Generate makefile

Make command:

Template makefile:

Data specification override

Ignore custom storage classes

Code Generation Advisor

Prioritized objectives: Unspecified

Check model before generating code

Generate code only

Configuration Parameters: EM001408_1APU/Configuration (Active)

Select:

- Solver
- Data Import/Export
- Optimization
- Diagnostics
 - Sample Time
 - Data Validity
 - Type Conversion
 - Connectivity
 - Compatibility
 - Model Referencing
 - Saving
- Hardware Implementation
- Model Referencing
- Simulation Target
 - Symbols
 - Custom Code
- Real-Time Workshop**
 - Report
 - Comments
 - Symbols
 - Custom Code
 - Debug
 - Interface
 - SIL and PIL Verification
 - Code Style
 - Templates
 - Code Placement
 - Data Type Replacement
 - Memory Sections
 - MAVERIC Target Code ge...

Component Type

- Actuator
- Actuator**
- APU
- AeroSurface
- Engine
- FCS
- Prop_Tank
- RCS_Thruster
- Sensor
- Slosh
- Stage
- Vehicle

Custom target file for MAVERIC cont.



- ◆ **The wrapper generated is easily readable and looks like all the other files in MAVERIC**
- ◆ **This file actually provides some insight into the generated code and how the structures are used to identify groups of data**

MAVERIC generated code wrapper excerpt



MAVERIC struct pointers for other components

Readable wrapper makes for easy unit testing

MAVERIC data struct to generated code struct

MAVERIC struct to generated code

Generated code data to MAVERIC struct

```

Function: init_Auto1205C
=====*/
int init_Auto1205C(int n)
{
    int i;

    /* pointer to this Maveric component */
    actuator *ActPtr = actuator_addr(n);

    /* pointers to other components that this component may access */
    vehicle *VehPtr = vehicle_addr(ActPtr->vehicle_ID);
    stage *StagePtr = stage_addr(ActPtr->stage_ID);
    engine *EnginePtr = ActPtr->eng;
    aero_surface *AeroSurfacePtr = ActPtr->sur;
    // fcs_outputs *FCSPtr = VehPtr->fsw.td->fcs;
    fcs_outputs *FCSPtr = &VehPtr->fsw.gnc.fcs.out;

    /* initialize RealTime vector */
    Auto1205C_M[n] = &Auto1205C_M[n];

    /* copy input signal data from Maveric to Simulink structs */
    // Auto1205C_U.ActuatorCmd = FCSPtr->ActuatorCmd[n];
    Auto1205C_U.ActuatorCmd = intlc ( ActPtr->actDeflTable, 2, ActPtr->time );

    /* copy parameter data from Maveric to Simulink structs */
    Auto1205C_P.Ap = ActPtr->Ap;
    Auto1205C_P.Be = ActPtr->Be;
    Auto1205C_P.Bias_torque = ActPtr->Bias_torque;
    Auto1205C_P.Closs = ActPtr->Closs;

    .
    .

    /* copy BlockIO signal data from Maveric to Simulink */
    /* BlockIO sometimes holds state information */
    Auto1205C_B[n].deflection = ActPtr->deflection;
    Auto1205C_B[n].deflectionDeg = ActPtr->deflectionDeg;

    .
    .

    /* Initialize model */
    Auto1205C_initialize(1, Auto1205C_M[n], &Auto1205C_B[n], &Auto1205C_X[n],
        &Auto1205C_U, &Auto1205C_Y);

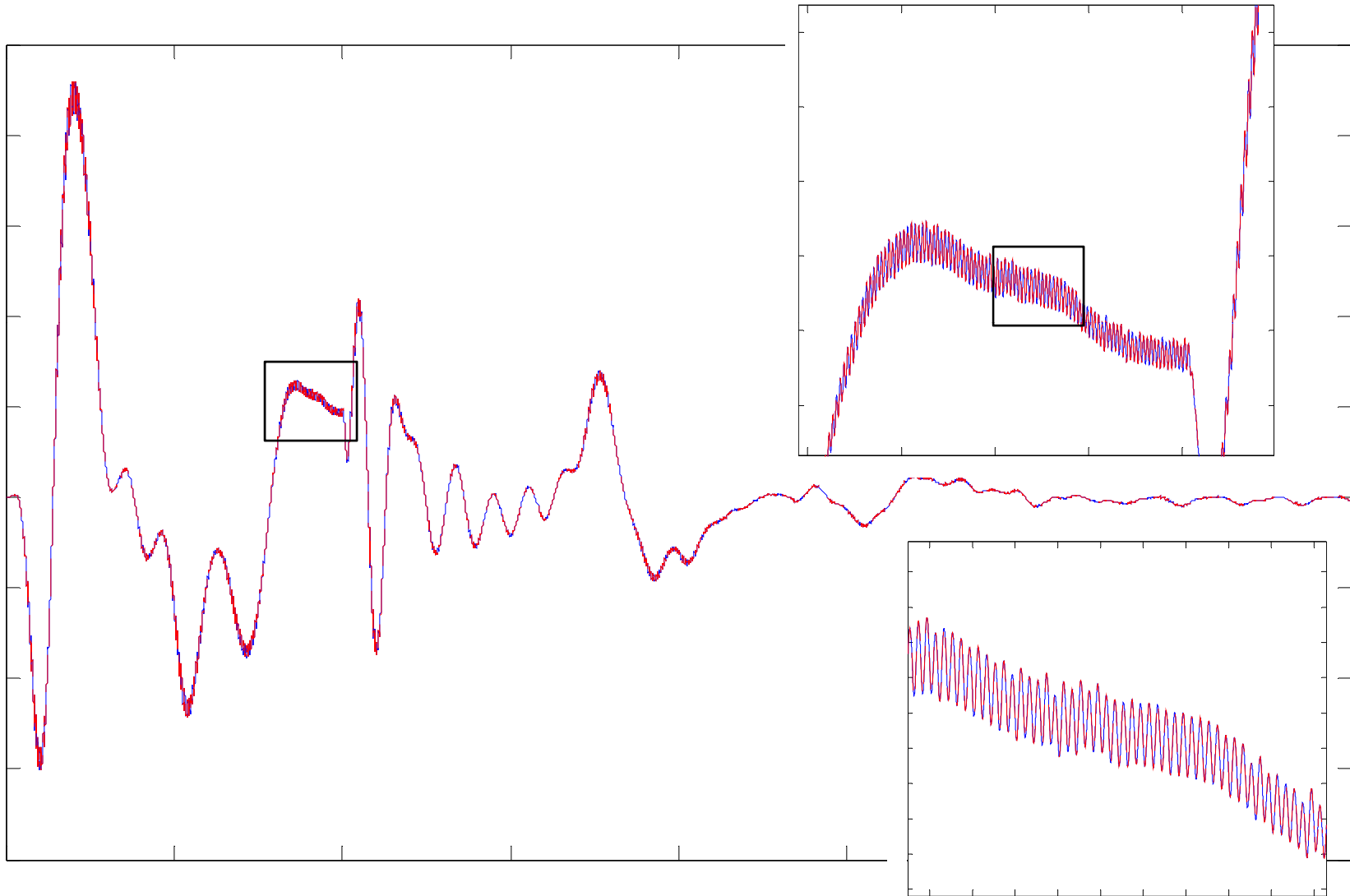
    /* copy BlockIO signal data from Simulink to Maveric structs */
    ActPtr->deflection = Auto1205C_B[n].deflection;
    ActPtr->deflectionDeg = Auto1205C_B[n].deflectionDeg;
    ActPtr->deflection_rate = Auto1205C_B[n].deflection_rate;

    .
    .

    /* initialize time for this component to the current sim time */
    ActPtr->time = t();
    
```

- ◆ **Chosen as the test bed for auto generated code in MAVERIC**
- ◆ **Not a trivial task to transform model into C code from Simulink**
 - Dynamics
 - servo valve (second-order w/ saturation)
 - nozzle (second-order load effects)
 - differential pressure feedback (single-order load control)
 - spring stiffness of all actuator attach points (back-up structure and nozzle)
 - drive electronics
 - smoothing filter (single-order)
 - avionics latencies (transport lag)
 - Non-linearities
 - square root term in the flow calculation path
 - servo valve spool position saturation
 - High integration rate recommended due to friction terms and high frequency dynamics
 - 10 kHz
- ◆ **Continuous model**

Spool Valve Position Plot (~50 Hz Dynamics)



- ◆ **Simplified model of hydraulic system providing pressure to the TVC actuators**
- ◆ **Simulates failure of one hydraulic system and two actuators powered from one system**
- ◆ **Couples the actuators where they are normally uncoupled systems**
- ◆ **Complex model to transform into C code**
 - Dynamics
 - accumulator (second-order)
 - valving (second-order pressure effects)
 - flow demand feed forward (single-order)
 - source pressure differential equation (single-order)
 - Non-linearities
 - pressure saturation
 - pump piston position limits
 - accumulator piston position limits
 - valve position limits
 - High integration rate recommended due coupling with actuator model
 - 10 kHz
- ◆ **Continuous model**

- ◆ **Approached about testing generated code with a model for an pressure estimator that was developed by another contractor**
 - Multiple large look up tables with interpolation
 - First order integrator for mass flow
 - First order integral controller
- ◆ **Generated code, installed into MAVERIC and unit tested in about one day**
- ◆ **Integrated generated code into flight controller and performed closed loop flight control testing next day**
- ◆ **This is a case where the auto coding provided a quick turnaround on a test case that may prove to be valuable in future flight control algorithm development**
- ◆ **Discrete model**

◆ Development time

- APU out model
 - ~8 weeks to Hand code and test
 - MAVERIC order of operation for TVC actuators had to be changed
 - ~2 weeks generate code using Real Time Workshop Embedded Coder and test
 - This was during early beta versions of maveric.tlc and issues were still being ironed out
 - Suspect that the effort would take < 1 week now
- Pressure Estimator
 - Implemented and kicked off Monte Carlo Run in ~2.5 working days

◆ Monte Carlo Analysis

- The generated code has worked well for MAVERIC Monte Carlo Analysis
 - APU out
 - 2 cases (2k each = 4k runs)
 - Pressure Estimator
 - 1 case (2k runs)

◆ ‘Buy-in’ from MAVERIC developers

- Resistant at first
- Interest raised based on readability, reproducibility, integration capability and speed
- Buy in and focus on other areas of work
 - design
 - analysis

- ◆ A comparison was made between the generated code simulation time and the hand written code simulation time for the APU out model implementation
- ◆ The results show an improvement when using the generated code versus the hand written code

Timing Comparison APU out Model:

Hand Code Nominal Case

real 8m33.265s
user 6m35.320s
sys 0m33.756s

Generated code Nominal Case

real 8m22.104s
user 6m32.287s
sys 0m29.830s

Opinions on the support from MathWorks



- ◆ **The support from The MathWorks Inc. has been outstanding!**
 - Long email chains about problems and trouble shooting implementations
 - Working on the most efficient way to integrate generated code into MAVERIC
 - Weekly then bi-Weekly meeting to discuss progress and future plans
 - The MAVERIC custom target file has been built to our specifications with inputs from the MathWorks
 - They gave us what we wanted and are ready to provide inputs for improving processes
- ◆ **Have to thank Mark McBroom in particular for providing all the support**
- ◆ **Thanks also to Brian Lallensack**

- ◆ **We are still testing and really getting a feel for the generated code in MAVERIC**
 - Testing situations include
 - simulation time
 - model complexity
 - modularity
- ◆ **Install generated code from multiple models into MAVERIC that interact**
 - Currently all generated coded models are independent and the interaction between two sets of generated code has not been tested
- ◆ **Generate code for other Simulink models – a limited amount of models in MAVERIC are delivered as Simulink models**
 - Challenge is changing delivery method of models
- ◆ **Develop standards/guidelines for model delivery and code generation for inclusion into MAVERIC**

Conclusions



- ◆ **The generated code provided the same output as the Simulink model that was used to create it!**
- ◆ **The generated code can be utilized in MAVERIC regardless of the very specific format required and is compatible with Monte Carlo time domain analysis**
- ◆ **The generated code is as fast if not faster than any hand code that I have created**
- ◆ **Code can be quickly generated and incorporated into MAVERIC**

References



- ◆ **McCarter, Jim, DEESE Research; “MAVERIC Training Sessions”, April 2010**
- ◆ **McCarter, Jim, DEESE Research; “MAVERIC User’s Guide”, 2010**
- ◆ **Hipp, Brent ,MSFC EV41; Bush, Jason, TriVector; “GNC8 FS TVC Modeling, Simulation, and Analysis”, September 2010**
- ◆ **Hannan, Mike, MSFC EV41; “MAVERIC Tool Summary”, 2010**