What Is Numerical Propulsion System Simulation (NPSS®)?

Southwest Research Institute® (SwRI®), San Antonio, TX *6/3/2016*

NPSS is an object oriented, multi-physics, engineering design and simulation environment which enables development, collaboration and seamless integration of system models. Primary application areas for NPSS include aerospace systems (i.e. engine performance models for aircraft propulsion), thermodynamic system analysis such as Rankine and Brayton cycles, various rocket propulsion cycles, and industry standardization for model sharing and integration. However, since it is fundamentally a flow-network solver, it has also been applied to a variety of other fluid/thermal subjects such as multi-phase heat transfer systems, refrigeration cycles, variations of common power cycles (i.e. Brayton), and overall vehicle emission analyses.

Figure 1 shows a typical air breathing engine block diagram in which the cycle is represented by the various thermodynamic processes such as compression, combustion and turbine power extraction. When developing a model in the NPSS environment, the engineer specifies the type and order of the necessary components (referred to as "elements") and provides the technical data which describes their individual performance. NPSS comes with a library of thermodynamic property databases and standard elements for use in engine cycle models.

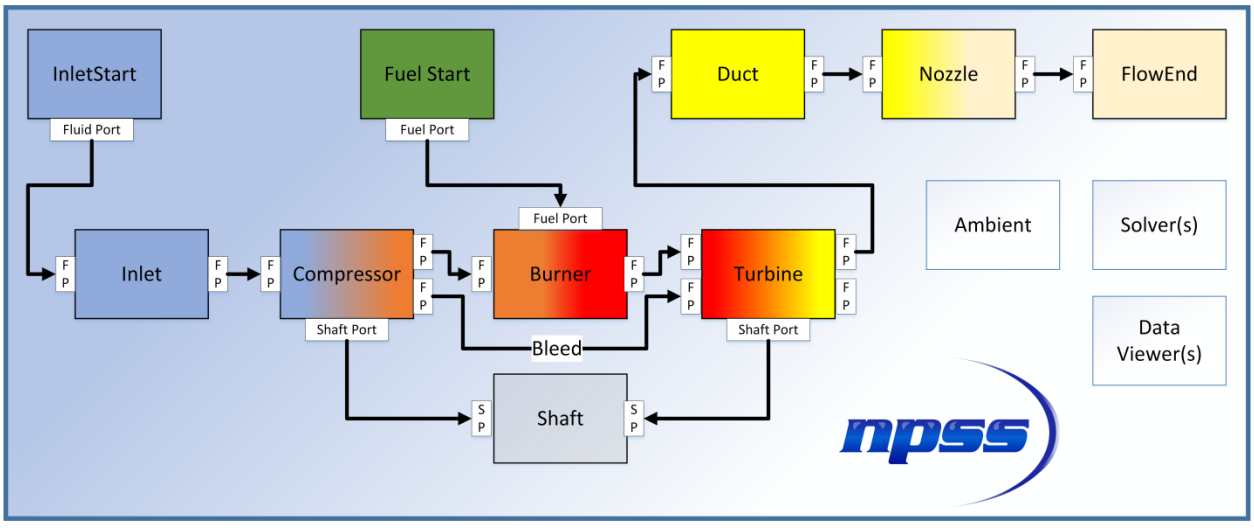

Figure 1. Typical Air Breathing Engine Block Diagram

Model definition is primarily performed through input files and simulations are typically launched via a system command window (Figure 2). Most users select a programming text editor that supports

language specific highlighting, coloring and auto-complete type features. Since NPSS is very much a C++ based tool, the C++ language setting in most text editors provides for excellent readability of the NPSS model and solution definition files.

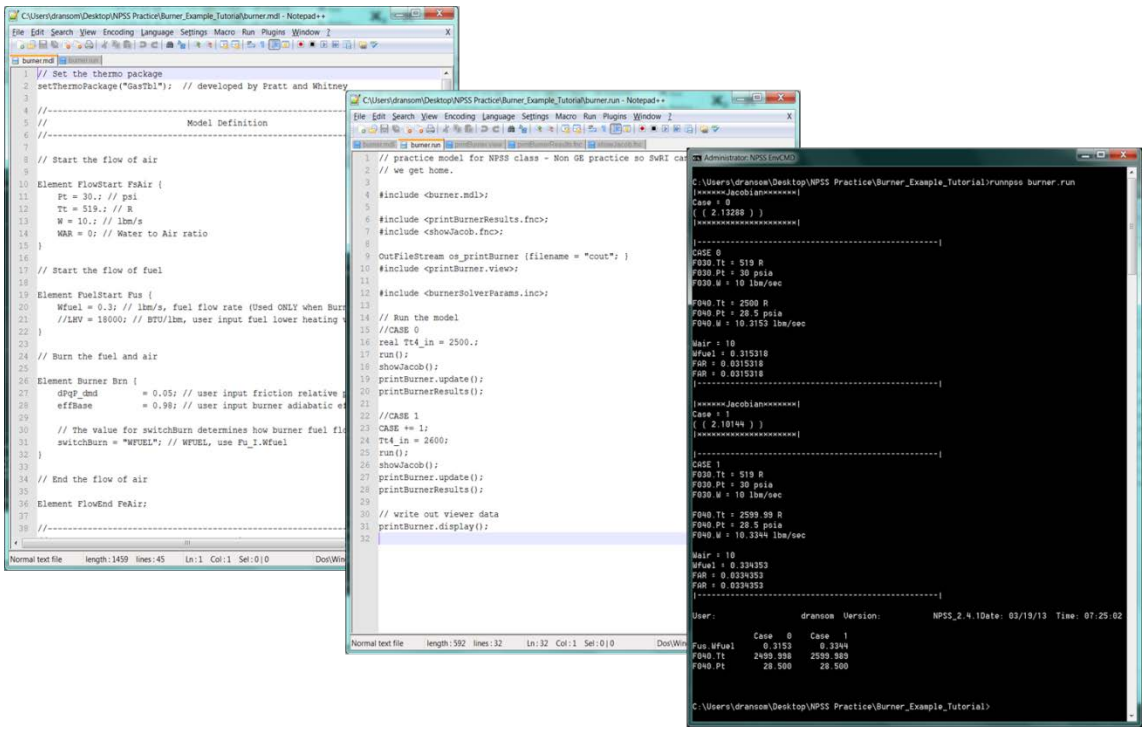

Figure 2. NPSS is File Based and Typically Run through System Command Window

Model Development

Elements are specified and linked together in a user-defined input file similar to that shown in Figure 3. In this example, the compressor, fuel start and burner elements are specified based on the NPSS included elements known as `Compressor`, `FuelStart` and `Burner`. The engineer gives them local names (i.e. `Cmp`, `FusEng` and `BrnPri`) and then assigns the known physical parameters necessary for the problem solution. The various elements are then connected through the use of `linkPorts` commands where the engineer again provides meaningful names to the connections such as `F020`, signifying that this connection is at station number 2.0 of the air breathing engine cycle.

```
Element Compressor Cmp {
    // Set compressor design point values
    PRdes  = 4.9; // design point pressure ratio
    effDes= 0.70; // design point efficiency
    Sh_O.inertia = .0005;
    // Load file that instantiates a subelement and plugs into compre
    // compressor performance map
    #include "hpcE3.map"; // when scoping, refer to the subelement by
}

// Start the flow of fuel
Element FuelStart FusEng {
    Wfuel = 0.004; // lbm/s, fuel flow rate (Used ONLY when Burner sw
    //LHV = 18000; // BTU/lbm, user input fuel lower heating value (L
}

Element Burner BrnPri {
    dPqP_dmd         = 0.05; // user input friction relative pressure
    effBase          = 0.98; // user input burner adiabatic efficien
    // The value for switchBurn determines how burner fuel flow rate
    switchBurn = "WFUEL"; // WFUEL, use Fu_I.Wfuel
}
```

```
linkPorts( "Cmp.Fl_O", "BrnPri.Fl_I", "F020" );
linkPorts( "FusEng.Fu_O", "BrnPri.Fu_I", "FuEng" );
linkPorts( "BrnPri.Fl_O", "Trb.Fl_I", "F030" );
linkPorts( "Trb.Fl_O", "Dexh.Fl_I", "F040" );
linkPorts( "Dexh.Fl_O", "NozPri.Fl_I", "F050" );
linkPorts( "NozPri.Fl_O", "FePri.Fl_I", "F060" );

linkPorts( "Cmp.Sh_O", "Sh.Sh_ICmp", "ShCmp" );
linkPorts( "Trb.Sh_O", "Sh.Sh_ITrb", "ShTrb" );
linkPorts( "Gen.Sh_O", "Sh.Sh_IGen", "ShGen" );
```

Figure 3. Example of Element Specification and Element Linking

In addition to the standard library of elements provided, the NPSS environment also allows for the definition of new elements or even modification/customization of NPSS provided elements. The NPSS user has access to the NPSS code used to define each element class (i.e. `Compressor`) and can then make their own version of the class with their own new name (i.e. `myCompressor`). This is a powerful feature of NPSS that provides significant flexibility in the types of elements used in a NPSS model. Figure 4 is an example taken from the `EngPerf` element which is included in the NPSS element library. On the left is part of the standard calculations intended to quantify the specific fuel consumption (SFC) for a thrust producing engine. For power generation applications, this element may be modified to calculate SFC based on shaft power. A possible user-modified solution is on the right which can then be saved as a new element, stored in the users own personal element library.

```
    //--------------------------------------------------------------
    // Sum up nozzle gross thrusts
    //--------------------------------------------------------------
    Fg = 0;
    for( i=0; i < _ptrFg.entries(); i+=1 ){
        Fg = Fg + _ptrFg[i]->value;
    }


    //--------------------------------------------------------------
    // Calculate overall values
    //--------------------------------------------------------------
    Fn = Fg - Fram;

    // If the static pressure string is empty, don't try to get its v
    if (_ptrPs != ""){
    Fnc = Fn * C_PSTD / _ptrPs->value;
    }

    Wfuel = Wfuel * 3600.0;      // LBM/SEC -> LBM/hr
    SFC = Wfuel / Fn;
  } // end calculate function
} // end EngPerf class
```

```
        Fg = Fg + _ptrFg[i]->value;
    }

    // DLR: sum of loads
    //--------------------------------------------------------------
    // Sum up torque loads
    //--------------------------------------------------------------
    tPwr = 0;
    for( i=0; i < _ptrLoad.entries(); i+=1 ){
        tPwr = tPwr + _ptrLoad[i]->value;
    }

    //--------------------------------------------------------------
    // Calculate overall values
    //--------------------------------------------------------------
    Fn = Fg - Fram;
    Fnc = Fn * C_PSTD / _ptrPs->value;
    Wfuel = Wfuel * 3600.0;      // LBM/SEC -> LBM/hr
    // DLR: modify SFC calc to use load as power basis instead of thr
    SFC = Wfuel / -tPwr;
    SFCkW = SFC*1.34; //convert to lb/kW/hr
  }
}
```

Figure 4. NPSS Provided `EngPerf` Sample and User Defined Modification of `EngPerf` Element

Problem Setup and Solution

Once the model is developed, the engineer must then setup the problem and define the solution goals and constraints. The problem setup consists mainly of gathering all of the inputs required for the solution, defining the type of outputs desired and where to send them, and definition of the solution cases to be run. NPSS is organized through the use of various input files although there are no formal rules regarding the number, type or organization of the input files. In the simplest models, the engineer can completely define the entire simulation in a single file. However, for larger models and more complicated systems, it is useful to organize the simulation using a variety of file types which are then linked together to form a complete simulation. A typical simulation contains the following types of files:

.*run*: typically the master file for the problem setup and points to all of the other files, specifies the desired thermodynamic package to be used for fluid properties, and defines/initializes data output files

.*mdl*: contains all of the elements and connectivity required to represent the fluid/thermal system

.*int* or .*dll*: contains the engineering methods for each of the elements used in the model (i.e. `Compressor` and `EngPerf`)

.*inc*: contains other items as needed for organization of the overall simulation such as an include file for all solver settings

.*fnc*: contains user defined functions to perform custom operations

.*view*: defines viewers used to extract data and send to formatted output file(s)

Figure 5 shows a very simple example of a .*run* file. In this case, the file is used to specify the thermodynamic package (`GasTbl`), includes the model with all of the elements and `linkPorts` (*burner.mdl*), includes some user-defined functions stored in .*fnc* files, identifies a data viewer file for output data (*printBurner.view*) and includes a user-defined file with solver parameters (*burnerSolverParameters.inc*). The right image shows a section of the file which defines a nested loop for evaluating the model over a range of lower heating value (LHV) and station number 4 total temperature (`Tt4_in`).

```
// practice model for Intro to NPSS class -

#define THERMO GasTbl
// #define THERMO allFuel

#include <burner.mdl>

#include <printBurnerResults.fnc>;
#include <showJacob.fnc>;

//OutFileStream os_printBurner {filename = "cout"; }
OutFileStream os_printBurner {filename = "burner.out"; }
#include <printBurner.view>

// OutFileStream os_dumpBurner {filename = "burnerDump.out"; }
// #include <dumpBurner.view>

#include <burnerSolverParams.inc>;
```

```
*/
real Tt4_in { value = 2500.; units = "R"; };
real Tt4Arr[] = (2300., 2400., 2500., 2600. );
real LHVRef[] = (18000., 18400., 18800. );

int iT4 = 0;
int iLHV = 0;
for (iLHV=0; iLHV<LHVRef.entries(); iLHV++) {
    Fus.LHV = LHVRef[iLHV];
    cout << "LHV Value = " << Fus.LHV << endl;
    for (iT4=0; iT4<Tt4Arr.entries(); iT4++) {
        Tt4_in = Tt4Arr[iT4];
        run();
        printBurner.update();
        CASE++;
    }
}

// write out viewer data
```

Figure 5. Typical *.run* File Organization

One important aspect of the problem definition is the specification of the solver settings. NPSS is unique because of the sophistication and configurability of the solver which enables the engineer to use the same model to solve a multitude of problems. For each problem to be solved, the engineer has the ability to define independent variables, dependent variables and solver constraints to direct the problem solution. An example of a solver independent/dependent pair definition is provided in Figure 6. In this case, the engineer is interested in knowing the fuel flow rate (`Fus.Wfuel`) required to achieve a specific burner temperature (`F040.Tt`). Therefore, the fuel flow is defined as an independent variable with some user-defined limits on the solver. There are many parameters which can be set to control the values for fuel flow rate that the solver will try. The burner temperature is identified as a dependent variable with a model parameter to monitor (`F040.Tt`) and a target temperature identified as a declared variable (`Tt4_in`). As shown in Figure 5 above, `Tt4_in` is evaluated over a range of values. As instructed, the solver determines the fuel flow rate associated with each `Tt4_in` value. Executing the solution requires a very simple command line entry shown in the right pane of Figure 6, telling NPSS to evaluate the *burner.run* file.

```
Independent ind_Wfuel {
    varName = "Fus.Wfuel";
    indepRef = "0.5";
    dxLimit = 0.2;
    dxLimitType = "FRACTIONAL";
    perturbation = 0.01;
    perturbationType = "FRACTIONAL";
    description = "vary the fuel flow to achieve the desired burner t
}

Dependent dep_F040Tt {
    eq_rhs = "Tt4_in";
    eq_lhs = "F040.Tt";
    eq_Ref = "Tt4_in";
    description = "desired burner temperature";
}
```

```
NPSS V27R2
C:\Users\dransom\Desktop\NPSS Practice\BurnerModel>runnpss burner.run
```

Figure 6. Example Solver Independent/Dependent Pair Definition and Command Window

This is a very basic example of the overall problem setup and solution control. There are many more options available for controlling the simulation and the solver can handle many independent/dependent pairs in a single solution. Multiple solvers can be used to improve the overall solution performance for models with sub-system assemblies. Various solution modes include Design, Off-Design and Transient. Design is used to determine element performance characteristics required to meet the design objective; Off-Design is used to determine how the selected design will perform away from the design point; and Transient is used to study system response to time varying conditions such as changing power level, sudden load application, or even a pressure vessel blow down.

Viewing Output Data

Once the solution sequence is complete, there are a variety of ways to view the output data. The data can be sent straight to the screen or to an output file designated in the problem setup. The data can be presented in a column view (`CaseColumnViewer`) or a row view (`CaseRowViewer`). In Figure 7, the data from the previous sweep of burner temperature is presented in a sequence of columns. Each column contains rows of data that were specifically identified for output in the problem setup. In this case, it is clear that for increasing firing temperature there is an associated increase in fuel flow rate, and this is true for two different values of `Fus.LHV`. It is also possible to have the data sent straight to the command window for immediate feedback to the engineer.

```
User:                dransom  Version:          NPSS_2.7R2Date: 11/24/14  T

                     Case  0    Case  1    Case  2    Case  3
Tt4_in                2300.0     2400.0     2300.0     2400.0
Tt4_in.units               R          R          R          R
Fus.LHV                18000      18000      18400      18400
Fus.LHV.units        Btu/lbm    Btu/lbm    Btu/lbm    Btu/lbm
Air Flow (kg/sec)     4.5359     4.5359     4.5359     4.5359
Air Flow (lbm/sec)   10.0000    10.0000    10.0000    10.0000
Fus.Wfuel             0.2848     0.3037     0.2782     0.2966
Fus.Wfuel.units      lbm/sec    lbm/sec    lbm/sec    lbm/sec
F040.Tt               2300.0     2399.8     2300.0     2400.0
F040.Tt.units              R          R          R          R
F040.Pt                14.0       14.0       14.0       14.0
F040.Pt.units          psia       psia       psia       psia
Brn.dPqP              0.050      0.050      0.050      0.050
Brn.EINOx             0.174      0.181      0.174      0.181
```

```
NPSS V27R2
C:\Users\dransom\Desktop\NPSS Practice\BurnerModel>runnpss burner.run
User:                dransom  Version:          NPSS_2.7R2Date: 11/24/14  Time: 13:27:27

                     Case  0    Case  1    Case  2    Case  3
Tt4_in                2300.0     2400.0     2300.0     2400.0
Tt4_in.units               R          R          R          R
Fus.LHV                18000      18000      18400      18400
Fus.LHV.units        Btu/lbm    Btu/lbm    Btu/lbm    Btu/lbm
Air Flow (kg/sec)     4.5359     4.5359     4.5359     4.5359
Air Flow (lbm/sec)   10.0000    10.0000    10.0000    10.0000
Fus.Wfuel             0.2848     0.3037     0.2782     0.2966
Fus.Wfuel.units      lbm/sec    lbm/sec    lbm/sec    lbm/sec
F040.Tt               2300.0     2399.8     2300.0     2400.0
F040.Tt.units              R          R          R          R
F040.Pt                14.0       14.0       14.0       14.0
F040.Pt.units          psia       psia       psia       psia
Brn.dPqP              0.050      0.050      0.050      0.050
Brn.EINOx             0.174      0.181      0.174      0.181

C:\Users\dransom\Desktop\NPSS Practice\BurnerModel>
```

Figure 7. Example Case Column Viewer Output (File and Command Window)

The data in the column or row viewers can be imported into other plotting tools to generate graphs as needed by the engineer. The format is fixed so updated data can be used to update plots without difficulty. In addition to the column and row viewers, there is significant flexibility to define other data outputs such as an overall performance summary, messages to the command line for solution progress monitoring, etc.

How to Get NPSS

For commercial and individual academic users, NPSS may be obtained from the Consortium website. Interested parties can download a trial version or purchase an annual license either as a university person (reduced rate) or a full commercial license for those in industry. NPSS is export controlled such that sales of NPSS are restricted to countries not currently listed on the U.S. Department of Commerce Anti-Terrorism watch list. For more details about purchasing NPSS, follow the link below:

http://npssconsortium.org

For users/companies with significant experience with NPSS and/or a large user base, it may be of interest to consider Consortium membership. SwRI manages the NPSS Consortium on behalf of the Consortium members. Members enjoy the additional benefits of access to the full source code (a significant value for building the source to fit your own computing resources), a large number of licenses for use by your company, input on the direction of development efforts for the future of NPSS, and the legal right to sub-license NPSS to your vendors for the purpose of model sharing. For more information about joining the consortium, please contact the Consortium Manager, David Ransom, by email (david.ransom@swri.org) or by phone, (210) 522-5281, or you can visit our Consortium web page for more information (www.npssconsortium.org).

List of Standard NPSS Thermodynamic Packages

The NPSS environment comes with a standard library of thermodynamic property packages that have been developed over the years by the Consortium and/or their member companies. They are all available for use in NPSS and, with the addition of the FPT property package, there is infinite ability for user-defined property definitions.

Keyword	Description
CEA	Implementation of the NASA chemical equilibrium code.
Janaf	Implementation of the National Institute of Standards and Technology gas properties prepared by Honeywell.
GasTbl	Package developed by Pratt & Whitney based on Therm, but adding humidity calculations and some chemical equilibrium capabilities.
AllFuel	Package developed by General Electric that contains gas properties and fuel properties.
FPT	Package used to define NPSS tables and/or functions that describe the thermodynamic properties of the fluid
REFPROP	REFPROP is not included in NPSS, but there is built in functionality to get properties from REFPROP if the user has the REFPROP code in the working directory (.dll and data files)

List of Standard NPSS Elements

The NPSS environment comes with a standard library of elements that have been developed over the years by the Consortium and/or their member companies. They are continually improved and developed as needed by the Consortium and are available for use and modification by users of NPSS. They are generally divided into categories based on their intended use.

Air Breathing Elements and Subelements: Typically used to model air breathing engine cycles

Elements: Ambient Bleed Burner Compressor Cycle Duct EngPerf FlowEnd FlowStart FuelStart Inlet InletStart Instrument InstrumentDuct Load Nozzle Shaft Splitter Turbine BleedOut BleedOutInterstage ControlVolume CrossOverValve Diffuser Emissions FuelSplitter FlowDuplicator HeatExchanger InverterValve Mixer Propeller ShaftSpring Slinger Wall

Subelements: CompressorRlineMap ThermalMass TurbinePRmap BurnEfficiency dPdiffuser dPqPMach dPqP FlightEnvelope PropCT RecoveryFactor RecoveryRatio TDay Valve WireCorrection

Fluid Network Elements: Typically used to model generic fluid/thermal systems and liquid propulsion cycles

FN_Duct FN_DuctInertial FN_FlowEnd FN_FlowStart FN_Leak FN_Pipe FN_Pump FN_Resistance FN_ResistanceInert FN_TMass FN_Turbine FN_Valve FN_Venturi FN_Volume FN_PumpMap FN_TurbMap FN_ValveHeadLoss FN_PumpMap FN_TurbMap FN_ValveHeadLoss FN_ReactedFlowStation FN_UnReactedFlowStation ReactedFlowStation UnReactedFlowStation

Infrastructure Elements: Typically used to support user-defined element development

Element ElementBase VariableContainer VariableOnlyContainer VCInterface Subelement