

What Is Numerical Propulsion System Simulation (NPSS®)?

Southwest Research Institute® (SwRI®), San Antonio, TX

1/7/2015

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.

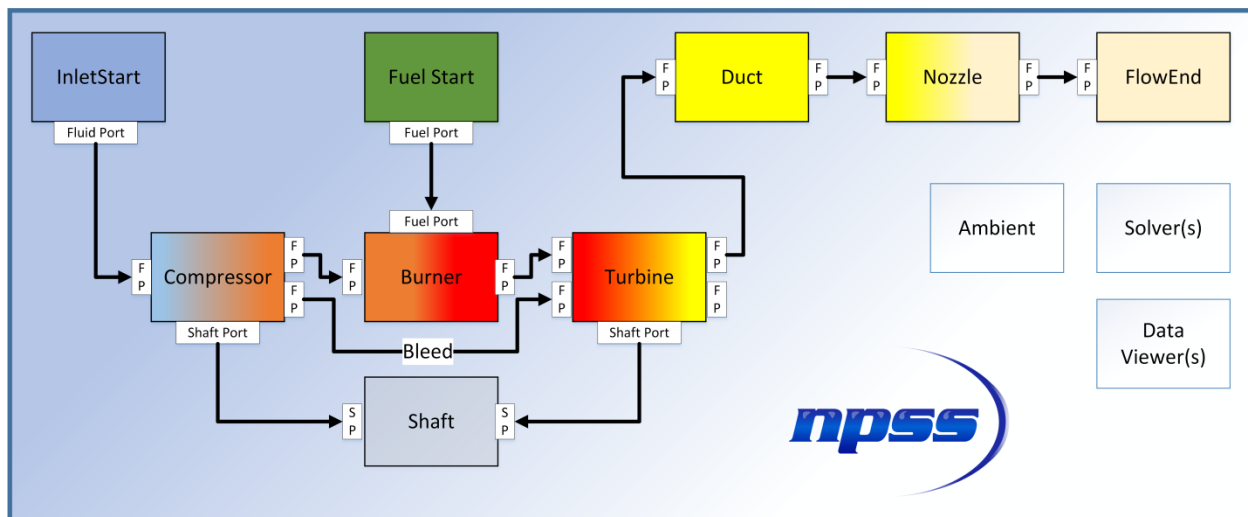


Figure 1. Typical Air Breathing Engine Block Diagram

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.

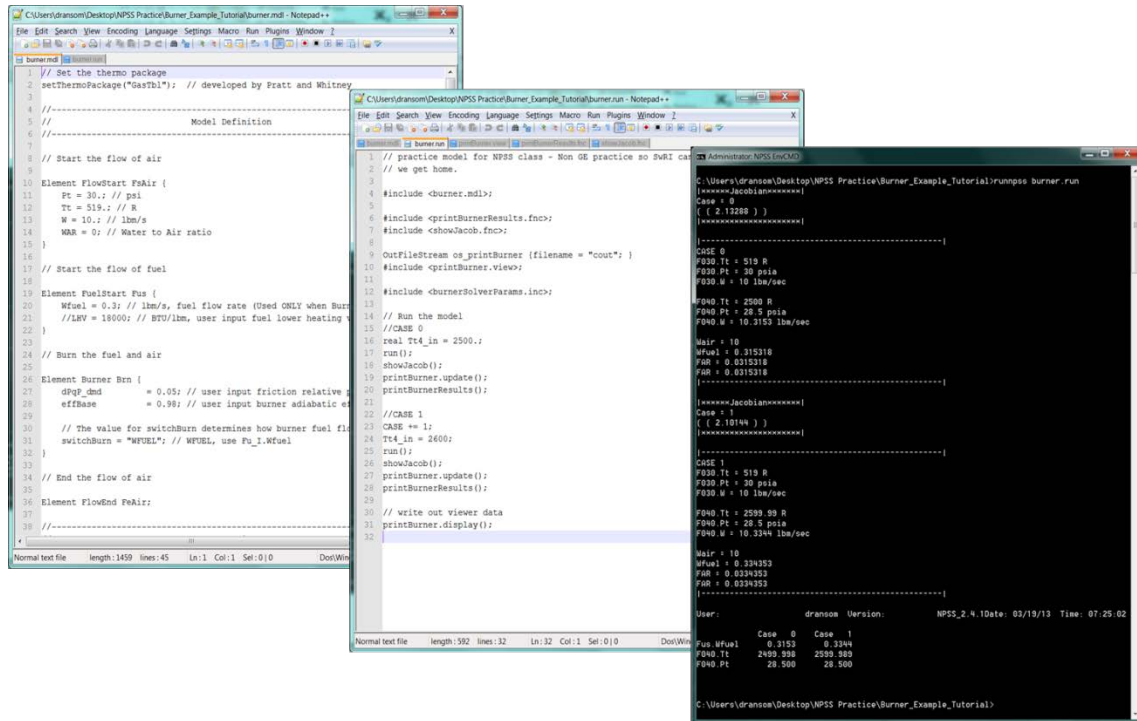


Figure 2. NPSS is File Based and Typically Run through System Command Window

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.

```

84 Element Compressor Cmp {
85     // Set compressor design point values
86     PRdes = 4.9; // design point pressure ratio
87     effDes= 0.70; // design point efficiency
88     Sh_O.inertia = .0005;
89     // Load file that instantiates a subelement and plugs into compr
90     // compressor performance map
91     #include "hpcE3.map"; // when scoping, refer to the subelement by
92 }
93
94 // Start the flow of fuel
95 Element FuelStart FusEng {
96     Wfuel = 0.004; // lbm/s, fuel flow rate (Used ONLY when Burner sw
97     // LHV = 18000; // BTU/lbm, user input fuel lower heating value (L
98 }
99
100 Element Burner BrnPri {
101     dPq_dmd = 0.05; // user input friction relative pressure
102     effBase = 0.98; // user input burner adiabatic efficien
103     // The value for switchBurn determines how burner fuel flow rate
104     switchBurn = "WFUEL"; // WFUEL, use Fu_I.Wfuel
105 }
106
107 Element Turbine Trb /
108
109
110
111
112
113
114
115
116
117
118 linkPorts( "Cmp.Fl_O", "BrnPri.Fl_I", "FO20" );
119 linkPorts( "FusEng.Fu_O", "BrnPri.Fl_I", "FuEng" );
120 linkPorts( "BrnPri.Fl_O", "Trb.Fl_I", "FO30" );
121 linkPorts( "Trb.Fl_O", "Dexh.Fl_I", "FO40" );
122 linkPorts( "Dexh.Fl_O", "NozPri.Fl_I", "FO50" );
123 linkPorts( "NozPri.Fl_O", "FePri.Fl_I", "FO60" );
124
125
126
127
128 linkPorts( "Cmp.Sh_O", "Sh.Sh_ICmp", "ShCmp" );
129 linkPorts( "Trb.Sh_O", "Sh.Sh_ITrb", "ShTrb" );
130 linkPorts( "Gen.Sh_O", "Sh.Sh_IGen", "ShGen" );
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

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.

```

382
383 //-----
384 // Sum up nozzle gross thrusts
385 //-----
386 Fg = 0;
387 for( i=0; i < _ptrFg.entries(); i+=1 ){
388     Fg = Fg + _ptrFg[i]->value;
389 }
390
391 //-----
392 // Calculate overall values
393 //-----
394 Fn = Fg - Fram;
395
396 // If the static pressure string is empty, don't try to get its v
397 if ( _ptrPs != "" ){
398     Fnc = Fn * C_PSTD / _ptrPs->value;
399 }
400
401 Wfuel = Wfuel * 3600.0; // LBM/SEC -> LBM/hr
402 SFC = Wfuel / Fn;
403 } // end calculate function
404 // end EngPerf class
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

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*).

```

1 // practice model for Intro to NPSS class -
2
3 #define THERMO GasTbl
4 // #define THERMO allFuel
5
6 #include <burner.mdl>
7
8 #include <printBurnerResults.fnc>;
9 #include <showJacob.fnc>;
10
11 //OutFileStream os_printBurner {filename = "cout"; }
12 OutFileStream os_printBurner {filename = "burner.out"; }
13 #include <printBurner.view>
14
15 // OutFileStream os_dumpBurner {filename = "burnerDump.out"; }
16 // #include <dumpBurner.view>
17
18 #include <burnerSolverParams.inc>;
19
40 L*
41 real Tt4_in { value = 2500.; units = "R"; };
42 real Tt4Arr[] = {2300., 2400., 2500., 2600. };
43 real LHVRef[] = {18000., 18400., 18800. };
44
45 int iT4 = 0;
46 int iLHV = 0;
47 for (iLHV=0; iLHV<LHVRef.entries(); iLHV++) {
48     Fus.LHV = LHVRef[iLHV];
49     cout << "LHV Value = " << Fus.LHV << endl;
50     for (iT4=0; iT4<Tt4Arr.entries(); iT4++) {
51         Tt4_in = Tt4Arr[iT4];
52         run();
53         printBurner.update();
54         CASE++;
55     }
56 }
57
58 // 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 ($F_{us} \cdot W_{fuel}$) required to achieve a specific burner temperature ($F_{040} \cdot Tt$). 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 ($F_{040} \cdot Tt$) 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.

```

4
5 Independent ind_Wfuel {
6     varName = "Fus.Wfuel";
7     indepRef = "0.5";
8     dxLimit = 0.2;
9     dxLimitType = "FRACTIONAL";
10    perturbation = 0.01;
11    perturbationType = "FRACTIONAL";
12    description = "vary the fuel flow to achieve the desired burner t
13 }
14
15 Dependent dep_F040Tt {
16     eq_rhs = "Tt4_in";
17     eq_lhs = "F040.Tt";
18     eq_Ref = "Tt4_in";
19     description = "desired burner temperature";
20 }
21
    
```

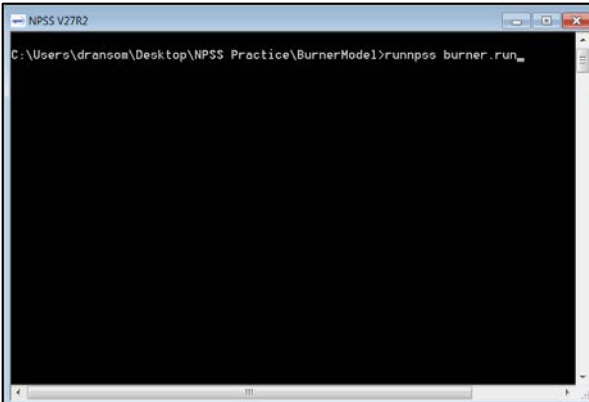


Figure 6. Example Solver Independent/Dependent Pair Definition and Command Window

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 $F_{us} \cdot LHV$. It is also possible to have the data sent straight to the command window for immediate feedback to the engineer.

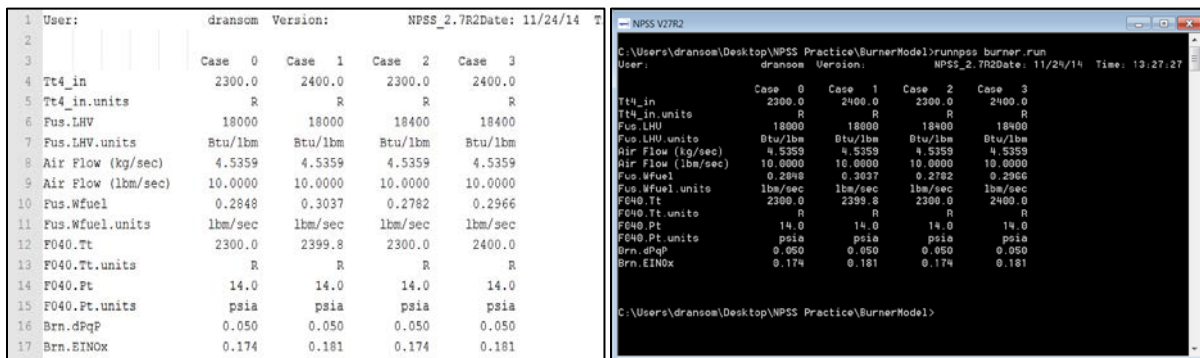


Figure 7. Example Case Column Viewer Output (File and Command Window)

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 through Wolverine Ventures Inc. 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://www.wolverine-ventures.com/>

Special pricing for academic institutions is available. For more information on academic institutional licensing contact Elliott Bryner, NPSS University Relations, by email (elliott.bryner@swri.org) or phone, (210) 522-5277.

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

WATE Elements: Used to model gas turbine engine systems for the purpose of estimating the overall system size, weight and arrangement.

WATEaccessories WATEballBearing WATEballBearingSimple WATEbearingBase

WATERollerBearing WATERollerBearingSimple WATEaugmentor WATEburnerBase

WATEductBurner WATEprimaryBurner WATEaxialCompressor WATEaxialHPC

WATEaxialLPC WATEcentrifCompressor WATEcompressorBase WATEfan WATEhiBypassFan

WATERotSplitterFan WATEsplitterFan WATEdiskBase WATEdiskMTC WATEdiskNew
WATEdiskOld WATEbypassDuct WATEcrossoverDuct WATEdiffuserDuct WATEductBase
WATEductNew WATEengineMount WATEframeBase WATEframeCustom
WATEframeIntermediate WATEframeSingleBearing WATEframeSingleBearingNoHPX
WATEframeSmallEngine WATEframeTurbineExit WATEgearbox WATEgearboxNew
WATEinlet2D WATEinlet2Dextern WATEinlet2DfixedRamp WATEinlet2Dmixed
WATEinletAxiExternExpCenter WATEinletAxiExternTransCenter WATEinletAxiFixedCenter
WATEinletAxiMixedExpCenter WATEinletAxiMixedTransCenter WATEinletBase
WATEinletBoeingTransCenter WATEinletMDALowDrag WATEinletMDAmixedVarCenterA
WATEinletMDAmixedVarCenterB WATEcoannularMixer WATEforcedMixer
WATEmixerBase WATE2DcdNozzle WATEacousticLiner WATEaxisymCDnozzle
WATEaxisymConvNozzle WATEfanAxiConvNozzle WATEdeployableMixerEjector
WATEmixerEjectorBase WATEplugMixerEjector WATEvariableMixerEjector
WATEvoricalChuteMixer WATEnozzleBase WATEnozzle2Dplug WATEnozzlePlug
WATEvarNozzleGeometry WATEhpShaft WATElpShaft WATEshaftBase WATEsplitter
WATEtester WATEfanThrustReverser WATEmda2DdoorThrustReverser
WATEmdaFanRevThrustReverser WATEmdaMixedFlowThrustReverser
WATEmdaPrimaryThrustReverser WATEmdaSepFlowThrustReverser
WATEprimaryThrustReverser WATEthrustReverserBase WATEtowerShaft
WATEaxialHPT WATEaxialLPT WATEaxialTurbine WATEradialHPT WATEradialLPT
WATERadialTurbine WATEturbineBase

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

Element ElementBase VariableContainer VariableOnlyContainer VCInterface Subelement

