TRNSYS Solar Combined Cycle Plant plus Storage

Paul J Harper*
Light Year CC, Stellenbosch

This report presents a TRNSYS model of a combined cycle solar central receiver plant with thermal storage. A combustion chamber inserted after the solar receiver maintained the input to the gas turbine at a constant temperature, enabling the gas turbine to operate near its optimum continuously. Thermal storage was added between the gas turbine and the steam cycle, enabling power from the steam cycle to be shifted till later in the day when demand in South Africa is highest. Overall efficiencies were calculated and ideas for further work presented.

I. INTRODUCTION

This report is the first of a series of reports for a development contract with dept Mechatronica and Mechanical Engineering, University of Stellenbosch. The emphasis in this report was on learning the TRNSYS modeling system and obtaining a working model that could be optimized for energy efficiency and cost at a later stage.

TRNSYS is a computer program developed by the University of Wisconsin for modeling transient thermal phenomena. Originally developed for modeling solar heating over an annual cycle, it is now widely used in the solar power industry for modeling parabolic trough and central receiver type systems.

A TRNSYS model of this size is quite complicated, the one discussed here had 36 components, 130 connections and 253 parameters. One mistake in any of the connections or parameters could cause the model to not function at all, become numerically unstable or to give completely spurious results. Quite a lot of time spent experimenting with various subsystems of the model in order to understand the components and their interactions.

II. MODEL

A schematic of the model is shown in Fig 1.

This model included 200 heliostats of 100m^2 each focusing onto an air receiver. A weather file for Upington with recorded Direct Normal Insolation was used to drive the power falling onto the solar receiver. This gives peak power thermal of just under 15 MW at the days peak around noon, See Fig 3.

This model used the standard heliostat efficiency matrix supplied with the heliostat field model, which was found to contain an error, where the azimuth and elevation angles were swapped around. A graph of the correct heliostat efficiency is shown in Fig 2 .This was not noticeable at high latitudes (such as Wisconsin) but produces an unexpected dip in radiation at noon at low latitudes like Upington. There are many computer codes to calculate Heliostat field efficiency (such as MIRVAL, FIAT LUX, WinDELSOL, HFLCAL and SolTRACE) [Garcia 2006], but we did not have time to investigate this aspect of the model.

A. Solar Only Approach

Initially we attempted a solar only approach, where we controlled the airflow through the compressor to produce a constant temperature at the exit of the air receiver. We attempted this because this is the approach used by solar trough systems, where they control the flow of the HTF through the system to produce a constant temperature at the output of the solar field. The air receiver has an output (desired mass flow) which we thought might be useful for this, but this output always stays at zero, so we presume that this output was not implemented in this model of the air receiver. We then created a controller using a model 22 controller from the TRNSYS library, and some equations to only switch on the compressor when the solar flux reached a threshold value(10,000 kJ/Hr or roughly 3 kW), and to invert the control signal from the model 22 controller. This worked after a fashion but was numerically unstable, and we could only run the simulation for a maximum of three days at one hour intervals. Any deviation of the step

*Electronic address: `pharper@choicecomp.com`

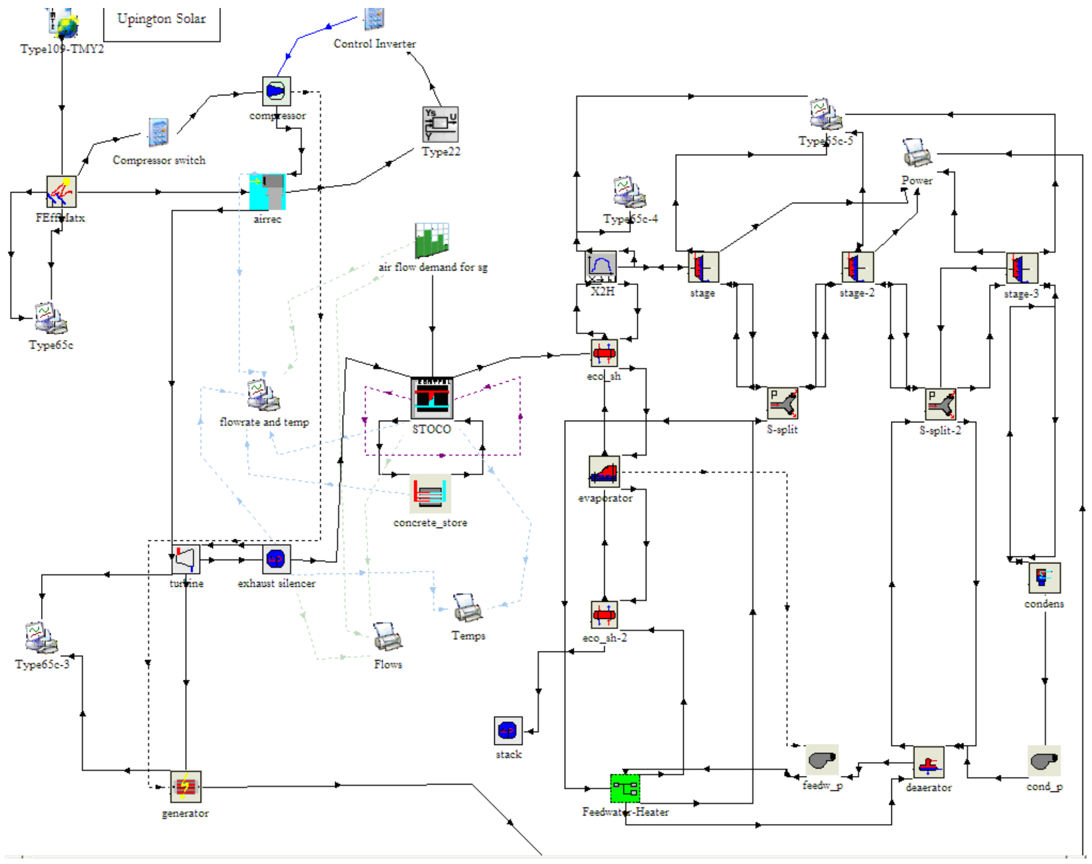

Figure 1: the TRNSYS model

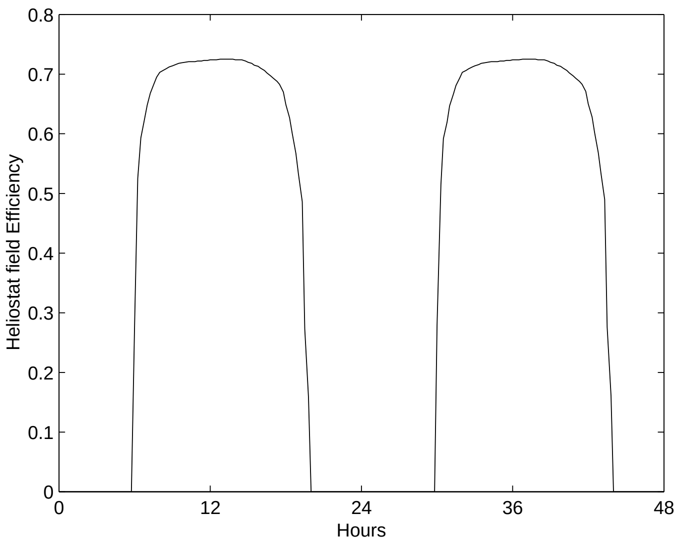

Figure 2: Corrected Heliostat Efficiency

size or attempts at longer runs lead to fatal FORTRAN errors (zero raised to a negative power) while running the simulation.

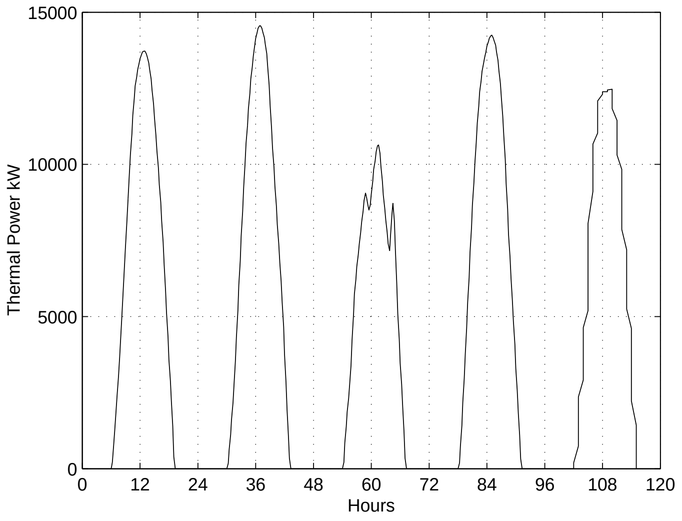

Figure 3: Incident Solar Power onto receiver

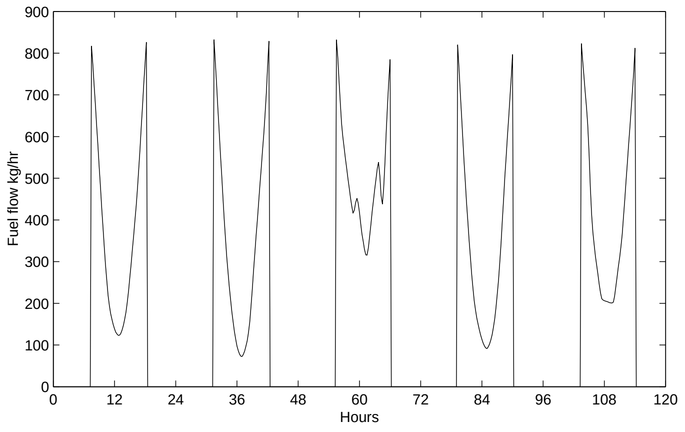

Figure 4: Fuel flow

B. Added Combustion Chamber

Because of the instability of the controllers available, and also because compressors and gas turbines do not operate efficiently over wide ranges of mass flow, we added a combustion chamber after the solar air receiver, and set it to maintain a constant output temperature of 1200 C. This then produced as an output the desired fuel flow in kg/hr (see Fig 4)

After some preliminary initial optimization of the air cycle (see figures ..), the air flow was set at 50 tons/hr, and compression ratio at 12.5. The gas turbine raw power (including power needed to drive the compressor) and the generated electric power are shown in Fig 5. During the day (when we have the compressor running) the raw turbine power was roughly 10 MW and the turbine electrical output about 4 MW. The other 6 MW was used to drive the compressor, and was lost in electrical inefficiency in the generator. The dip in the power output is presumably due to the slightly different thermodynamic properties of the gas (pure air at noon, more combustion products in the morning and evening).

After the gas turbine stage a storage controller and concrete heat store was implemented, which fed the Heat Recovery Steam generator (HRSG) of a Rankine cycle steam turbine.

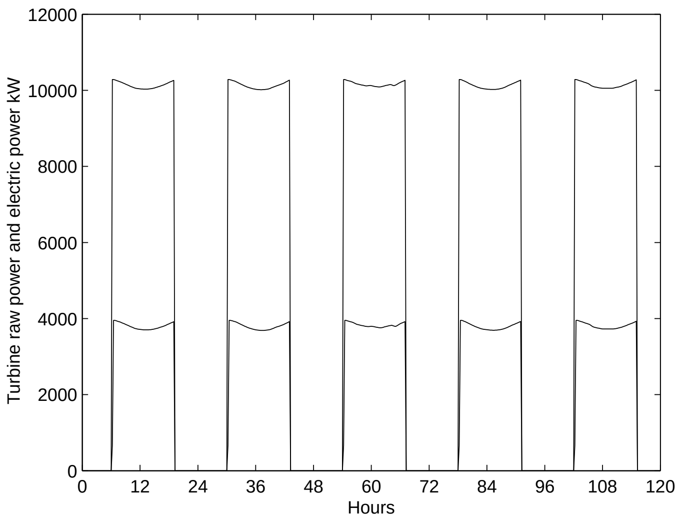

Figure 5: Gas Turbine raw power and electric power

C. Storage

A concrete thermal storage was implemented straight after the exhaust of the gas turbine. This was paired with a controller unit. The controller unit has as inputs HTF (Heat Transfer Fluid) mass flow and temp, as well as Steam generator mass flow demand. It is modeled as a stratified tank, split into a number of nodes along the hot cold axis. The model is simplified and does not consider non-axial flows and temperature gradients, or gradients inside the concrete. The model has a hot end and a cold end, and flows are allowed in one direction only at a time: either hot to cold (charge) or cold to hot (discharge). The controller decides that the store is full when during charge the cold end reaches a set temperature. Likewise the model decides that the storage is empty when the hot end falls below a certain set temperature on discharge. If there is HTF flowing into the model and there is no Steam generator demand, then the storage is charged till full. If there is steam generator demand, the controller decides whether to take mass flow for the steam generator either directly from the HTF, or from discharge from the storage.

If there is no HTF flow, and there is steam generator demand, then the mass flow will be taken from the storage, provided that there is still sufficient thermal energy in the storage.

In the current model we have Steam generator demand set to ramp from 0 to maximum from 12:00 to 14:00, to stay at the maximum till 21:00 then to ramp down till 22:00. This leads to controller behavior of charging till 12:00, then a ramp down of charging till 14:00, and a ramp up of air flow straight from the gas turbine to the steam cycle. When the air cycle is shut down when the sun sets, then the storage discharges (see Fig 6)

1. Size of Concrete Store

We increased the size of the concrete store till it gave meaningful results. In Fig 7 we show the temperatures over a period of 10 days. This was for a concrete store with a mass of 1000 tons. This might a bit much storage, as it can be seen that although the hot end is fully charged after 3 days, the cold end takes about 7 days. This concrete store was divided into 13 thermal levels and had a thermal heat transfer coefficient of 3.0 10^6 kJ/hr.K. This might be a totally unrealistic thermal store size, and should be optimized during further work on the model.

2. Rock Bed Storage Model

We initially attempted to use the rock bed thermal model (type 502) from the STEC library. We connected it to the STOCO controller in a similar manner to the thermal concrete storage. This model has its default number of nodes (thermal stratification levels) set to 1. With one node the TRNSYS model converges and gives results, but as soon as we attempted to increase the number of nodes, the model either gives numeric errors or hangs up. We tried many different numbers of nodes, 3,5,7,13,17 and 23 but it was unstable at all of them except 1. Thus we reverted back to the thermal concrete storage model (type 430).

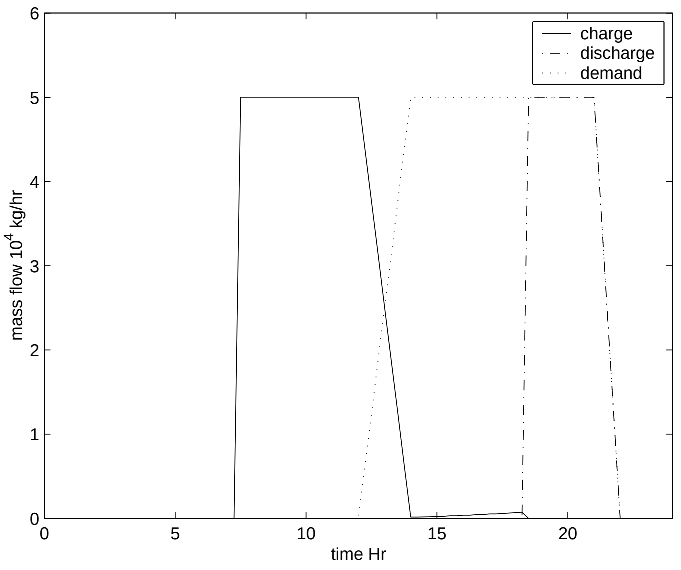

Figure 6: Charge, Discharge and Demand for Steam generator flows

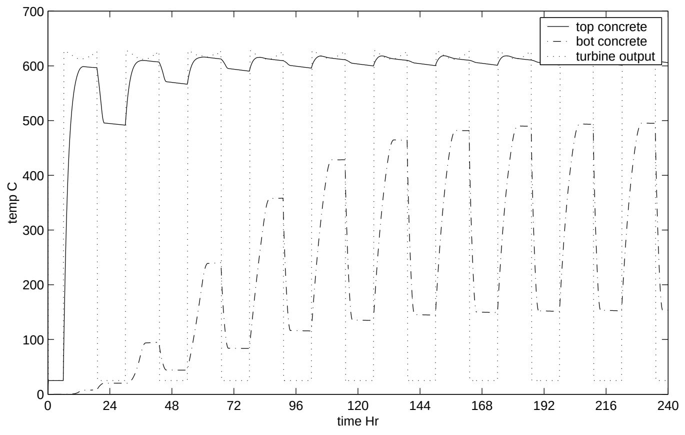

Figure 7: Hot and cold end temperatures of concrete storage

D. Steam Cycle

We used the default Rankine cycle example supplied with the STEC library, as we initially had problems with building a steam cycle from scratch. This is definitely another aspect of the model that also still needs some optimization. We also did some tests on the steam cycle in a stand alone mode, and found that the electrical power generated was only about 60% of that expected from other references [Kehlhofer 1999]. We also wrote some MATLAB code (see VI) to attempt to optimize the steam cycle, by altering the parameters of the heat exchangers in the HRSG. We set a constant flow of input air at a fixed temperature to the HRSG to optimize under steady state conditions. What we learned was that this optimization also needs to have a cost component, as optimizing for power output alone tends to make the heat exchangers very large. In Fig 8 we see the electric power generated by the steam cycle. During the first day when the thermal storage is still charging up, there is a steep shoulder on the right of the steam power curve when power is being drawn from the thermal store which is not yet full, but by the fifth day, that problem has dissipated.

In Fig 9 we see the total combined power output with the current time shifting, this shows the sum of the steam and air cycles electric output. remember that we are switching the fuel on and off with the rising and setting of the sun.

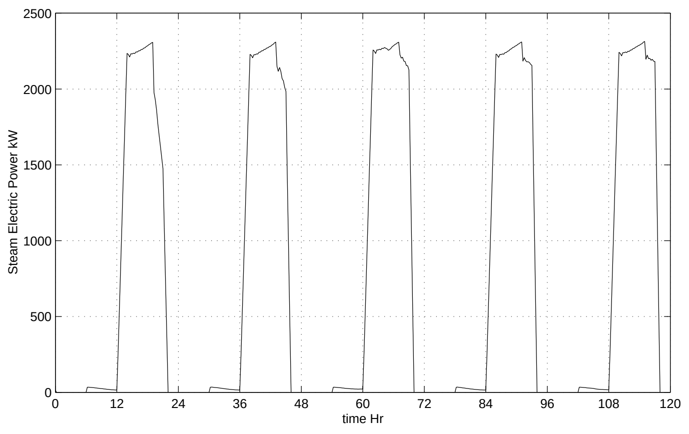

Figure 8: Electric Power Steam Cycle

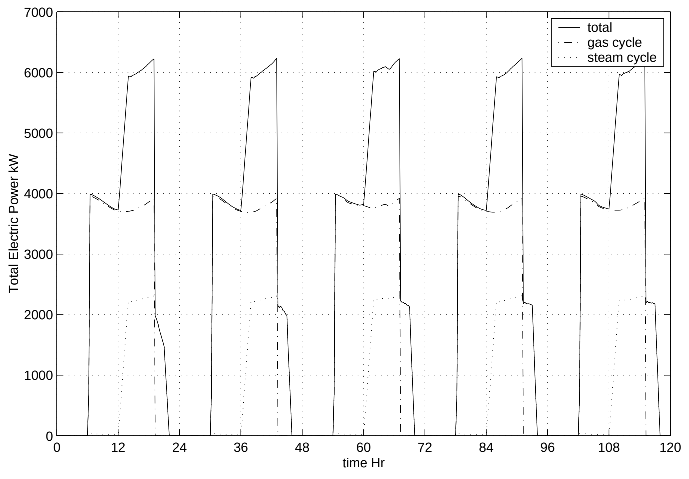

Figure 9: Combined Cycle Electric Output

III. TOTAL EFFICIENCIES

Over a period of 240 hours our calculated efficiencies were as shown in Table I. This was an integration over the whole period, and a substantial amount of energy was still in the thermal store, but probably not enough to raise the efficiency more than a few percentage points.

	*10^5 kW hr	Efficiency
Total Solar Input	5.3386	
Total Fuel input	12.786	
Total Input	18.1246	
Electric Output	6.762	
Efficiency		37.3%
Solar Fraction		29.5%

Table I: Total Efficiencies over 10 day period

IV. PROBLEMS

We encountered several problems in building up models with TRNSYS, and we detail some of them here.

- Inverted solar efficiency matrix(the azimuth and elevation components were swapped around)
- Wrong units on certain components, condenser, rock bed storage. Nearly all units work with energy in units of kJ/hr. The units are listed for the condenser and rock bed storage as kg/sec. We ignored the labeling, and presumed it to be in error, as several samples had the components directly connected together without any conversion equations between them (which would have been necessary if there really was a unit mismatch)
- Numeric instability of controllers. We initially tried to control air flow through the compressor using a type 22 PID controller, which exhibited instability .
- Numeric instability of certain models E.g. rock bed storage with more than one node

V. SUMMARY AND FURTHER WORK

We have constructed a preliminary working TRNSYS model that is starting to give meaningful results. Items that need to be addressed going forward are:

1. The efficiency of the steam cycle presented here is lower than expected, and this needs to be improved
2. Rough Cost estimates of heat exchangers and other major components need to be obtained to place upper limits on components used in the model.
3. A realistic plant size must be chosen. This could either be a pilot plant or an initial small scale industrial plant , and the model optimized around that
4. Check if it is at all possible to go to molten salt storage (air/salt heat exchangers)
5. Have an electricity demand/pricing graph for a period of a day to enable planners to try to get maximum utility from the model and storage.

VI. MATLAB INTERFACE

This section describes the routines created to interface MATLAB (12 and later) with TRNSYS 16. These routines enable MATLAB to run TRNSYS simulations, to optimize models over multiple parameters and to do parametric runs much easier than the built in Trnedit application which is supplied with TRNSYS.

A. Structure

MATLAB has an interface for calling external programs called mex functions. We wrote a mex function named runtrnsys.c that enables MATLAB to run a TRNSYS simulation, passing in certain parameters, and reading out the results of the simulation. A typical call to runtrnsys appears as follows:

B. MATLAB Code

```
deckdir = 'C:\Trnsys16\MyProjects\steamtest';
trnsyspath = 'C:\Trnsys16\Exe\TRNExe.exe';
deckfile = 'STEC_CYC-Steam3.dck';
outputfile = 'power.out';
hesuper = 200000;
heevap = 400000;
outresults = runtrnsys(deckdir, trnsyspath, deckfile, outputfile, 'HESUPER', hesuper,'HEEVAP',heevap);
```

The deckdir, trnsyspath, deckfile and outputfile parameters are paths and file names. After the outputfile parameter, there are a variable number of parameters, each consisting of a pair: of (Parameter name, numeric value). In the above example we have passed in two parameters, the HESuper value and the HEEvap value.

The results that you are interested in from the simulation should be printed to the named outputfile using a TRNSYS unit 25c (output printer). Set the printer to not print a header line, and to overwrite the output file on each run. All the results from the run (parsed from the first line of the output file) will be passed back in a vector outresults.

In typical usage this function would be wrapped in another m-file, which took a vector as input, and which returned a single scalar as output. An example of this is the function steam3.m which takes a vector (typically of heat exchanger coefficients) and returns an output (plant output in MWe). Steam3 can then be passed to any of the MATLAB optimization routines such as fminunc (unconstrained optimization) or fminsearch (Nelder Meade non linear search)

C. Preparing a deck file for MATLAB optimization or parametrized runs

Start off with a model that you have generated in simulation Studio. Ensure that the output that you are interested in is written to a file by a unit 25c. Run the simulation in Simulation Studio. Take the deck file written by Simulation Studio, and rename it to something else. In our case we are working with a sample deck file called STEC_CYC-Steam3.dck Into this new deck file, you must add a second CONSTANTS block just after the first CONSTANTS block inserted by Simulation Studio. See the sample deck file STEC_CYC-Steam3.dck for an example (line 28). Into this second constants block you must place in parametrized form all the variables that you want to alter over the course of an optimization or over the course of a parametric run. These named variables must match in name those passed into the runtrnsys call. You do not have to change all the variables in the call to runtrysys, just the ones you want to change.

D. Sample Code

All the sample code is contained in the files included in the zip file with this document

E. Optimization results

The MATLAB optimization function fminunc (unconstrained optimization, using a modified Newton method) were unsuccessful, probably because of lack of smoothness in first and second derivatives. Fminsearch was successful after we introduced penalty code to avoid any of the Heat exchanger components from going negative. One disadvantage of just optimizing for power over the heat exchanger capacities is that this tends to make very large heat exchangers, and so some cost function must be added to this, or a maximum heat exchanger size could be introduced using the MATLAB fminmax function.

VII. APPENDIX A TRNSYS DECK FILE FOR THE MODEL

```
VERSION 16.1
*******************************************************************************
*** TRNSYS input file (deck) generated by TrnsysStudio
*** on Wednesday, March 25, 2009 at 21:41
*** from TrnsysStudio project:  C:\Trnsys16\MyProjects\paulcombinedcycleStorage\paulsCombinedCycleStorageCombustion.tpf
***
*** If you edit this file, use the File/Import TRNSYS Input File function in
*** TrnsysStudio to update the project.
***
*** If you have problems, questions or suggestions please contact your local
*** TRNSYS distributor or mailto:software@cstb.fr
***
*******************************************************************************
*******************************************************************************
*** Units
*******************************************************************************
*******************************************************************************
*** Control cards
*******************************************************************************
* START, STOP and STEP
CONSTANTS 3
START=0
STOP=120
STEP=.25
* User defined CONSTANTS
SIMULATION START STOP STEP ! Start time End time Time step
```

```
TOLERANCES .01 .01 !  Integration Convergence
LIMITS 30 60 30 !  Max iterations Max warnings Trace limit
DFQ 1 !  TRNSYS numerical integration solver method
WIDTH 80 !  TRNSYS output file width, number of characters
LIST ! NOLIST statement
!  MAP statement
SOLVER 0 1 1 !  Solver statement Minimum relaxation factor Maximum relaxation factor
NAN_CHECK 0 !  Nan DEBUG statement
OVERWRITE_CHECK 0 !  Overwrite DEBUG statement
TIME_REPORT 0 !  disable time report
EQSOLVER 0 !  EQUATION SOLVER statement
* Model "Type109-TMY2" (Type 109)
*
UNIT 2 TYPE 109 Type109-TMY2
*$UNIT_NAME Type109-TMY2
*$MODEL .\Weather Data Reading and Processing\Standard Format\TMY2\Type109-TMY2.tmf
*$POSITION 71 48
*$LAYER Weather - Data Files #
PARAMETERS 4
2 !  1 Data Reader Mode
38 !  2 Logical unit
4 !  3 Sky model for diffuse radiation
1 !  4 Tracking mode
INPUTS 3
0,0 !  [unconnected] Ground reflectance
0,0 !  [unconnected] Slope of surface
0,0 !  [unconnected] Azimuth of surface
*** INITIAL INPUT VALUES
0.2 0 0
*** External files
ASSIGN "C:\Trnsys16\Weather\Meteonorm\Africa\ZA-Upington-684240.tm2" 38
*|?  Weather data file |1000
*------------------------------------------------------------------------------
* Model "air flow demand for sg" (Type 14)
*
UNIT 46 TYPE 14 air flow demand for sg
*$UNIT_NAME air flow demand for sg
*$MODEL .\Utility\Forcing Functions\General\Type14h.tmf
*$POSITION 479 288
*$LAYER Main #
PARAMETERS 12
0 !  1 Initial value of time
0.0 !  2 Initial value of function
12 !  3 Time at point-1
0 !  4 Value at point -1
14 !  5 Time at point-2
50000 !  6 Value at point -2
21 !  7 Time at point-3
50000 !  8 Value at point -3
22 !  9 Time at point-4
0 !  10 Value at point -4
24 !  11 Time at point-5
0 !  12 Value at point -5
*------------------------------------------------------------------------------
* Model "FEffMatx" (Type 394)
*
UNIT 6 TYPE 394 FEffMatx
*$UNIT_NAME FEffMatx
*$MODEL .\STEC Library\STE\CRS\Heliostat Field\FEffMatx.tmf
*$POSITION 70 234
*$LAYER Air Loop #
PARAMETERS 9
10 !  1 Unit no of Input file
7 !  2 No of zenith angle data points
9 !  3 No of azimuth angle data
200 !  4 No of concentrator units
100 !  5 Mirror surface area
0.9 !  6 av reflectivity
0 !  7 startup energy of unit
0 !  8 power to track 1 unit
999.9 !  9 max allowed windspeed
INPUTS 5
2,12 !  Type109-TMY2:total radiation on horizontal ->Direct normal insolation
0,0 !  [unconnected] wind speed
0,0 !  [unconnected] on/off control
2,10 !  Type109-TMY2:solar zenith angle ->solar zenith angle
2,11 !  Type109-TMY2:solar azimuth angle ->solar azimuth angle
*** INITIAL INPUT VALUES
0 0 1 90 0
*** External files
ASSIGN "matr02-reversed.dat" 10
*|?  File name of efficiency file |1000
*------------------------------------------------------------------------------
```

```
* Model "compressor" (Type 424)
*
UNIT 18 TYPE 424 compressor
*$UNIT_NAME compressor
*$MODEL .\STEC Library\Brayton\DLR\compressor.tmf
*$POSITION 308 124
*$LAYER Main #
PARAMETERS 5
12.5 !  1 compression ratio
.99 !  2 mechanical efficiency
100000 !  3 ISO inlet mass flow_design
0 !  4 partial load by mass flow reduction if mode2
2 !  5 operating mode
INPUTS 5
0,0 !  [unconnected] inlet air temperature
0,0 !  [unconnected] inlet pressure
CompressorOn !  Compressor switch:CompressorOn ->inlet mass flow if mode2
0,0 !  [unconnected] isentropic efficiency
0,0 !  [unconnected] cooling air mass ratio
*** INITIAL INPUT VALUES
25 1 0 .88 0
*------------------------------------------------------------------------------
* Model "airrec" (Type 422)
*
UNIT 17 TYPE 422 airrec
*$UNIT_NAME airrec
*$MODEL .\STEC Library\STE\CRS\Tower Receiver\airrec.tmf
*$POSITION 301 234
*$LAYER Air Loop #
*$# Model1:  Simple black body absorber including radiative and convective losses.
PARAMETERS 18
1 !  1 receiver model
0.9 !  2 optical efficiency
1 !  3 emissivity of absorber
25 !  4 receiver aperture
1 !  5 absorber fraction
100 !  6 lower flux limit
1000000 !  7 upper flux limit
11 !  8 operation mode
0.6 !  9 emissivity of hot gas piping
1 !  10 surface area of piping
0 !  11 convective loss coeff.of piping
0 !  12 cooling loss factor
0 !  13 pressure loss factor
10 !  14 design inlet pressure
300 !  15 design inlet temperature
1000 !  16 design outlet temperature
0 !  17 design inlet mass flow
0 !  18 electric parasitics
INPUTS 7
18,1 !  compressor:outlet temperature ->air inlet temperature
18,2 !  compressor:outlet pressure ->air inlet pressure
18,7 !  compressor:outlet enthalpy ->air inlet enthalpy
18,3 !  compressor:outlet mass flow working air ->air inlet mass flow
6,1 !  FEffMatx:Power to receiver ->solar input power
0,0 !  [unconnected] control signal
0,0 !  [unconnected] ambient temperature
*** INITIAL INPUT VALUES
25 1 0 0 0 1 25
*------------------------------------------------------------------------------
* Model "Flows" (Type 25)
*
UNIT 47 TYPE 25 Flows
*$UNIT_NAME Flows
*$MODEL .\Output\Printer\Unformatted\No Units\Type25c.tmf
*$POSITION 433 725
*$LAYER Outputs #
PARAMETERS 10
STEP ! 1 Printing interval
START ! 2 Start time
STOP ! 3 Stop time
46 !  4 Logical unit
0 !  5 Units printing mode
0 !  6 Relative or absolute start time
-1 !  7 Overwrite or Append
-1 !  8 Print header
0 !  9 Delimiter
1 !  10 Print labels
INPUTS 5
20,4 !  exhaust silencer:outlet mass flow ->Input to be printed-1
44,5 !  STOCO:Storage charge flow ->Input to be printed-2
44,7 !  STOCO:Storage discharge flow ->Input to be printed-3
46,2 !  air flow demand for sg:Instantaneous value of function over the timestep ->Input to be printed-4
```

```
44,3 !  STOCO:SG input HTF flow ->Input to be printed-5
*** INITIAL INPUT VALUES
m_Solar m_charge m_discharge m_demand m_use
*** External files
ASSIGN "flows.dat" 46
*|?  Output file for printed results |1000
*------------------------------------------------------------------------------
* Model "STOCO" (Type 431)
*
UNIT 44 TYPE 431 STOCO
*$UNIT_NAME STOCO
*$MODEL .\STEC Library\Storage\STOCO.tmf
*$POSITION 481 462
*$LAYER Main #
*$#
*$#
*$#
PARAMETERS 5
500 !  1 Charge max oil bottom Temp
300 !  2 Discharge min oil top temp
1.04 !  3 cp HTF
10 !  4 min storage flow
50000 !  5 max stoage flow
INPUTS 8
20,4 !  exhaust silencer:outlet mass flow ->Solar HTF flow
20,3 !  exhaust silencer:outlet temperature ->Solar HTF Temperature
46,2 !  air flow demand for sg:Instantaneous value of function over the timestep ->SG desired HTF flow
44,3 !  STOCO:SG input HTF flow ->SG real HTF flow
20,3 !  exhaust silencer:outlet temperature ->SG HTF cold temperature
49,3 !  concrete_store:HTF temp on top ->Storage top HTF Temp
49,1 !  concrete_store:HTF temp at bottom ->Storage bottom HTF Temp
49,6 !  concrete_store:total energy stored since sim started ->Storage total Energy
*** INITIAL INPUT VALUES
40000 400 40000 40000 20 550 500 1
*------------------------------------------------------------------------------
* Model "Type65c" (Type 65)
*
UNIT 7 TYPE 65 Type65c
*$UNIT_NAME Type65c
*$MODEL .\Output\Online Plotter\Online Plotter With File\No Units\Type65c.tmf
*$POSITION 59 373
*$LAYER Outputs #
PARAMETERS 12
1 !  1 Nb.  of left-axis variables
1 !  2 Nb.  of right-axis variables
0 !  3 Left axis minimum
50000000 !  4 Left axis maximum
0 !  5 Right axis minimum
1 !  6 Right axis maximum
1 !  7 Number of plots per simulation
12 !  8 X-axis gridpoints
0 !  9 Shut off Online w/o removing
39 !  10 Logical Unit for output file
0 !  11 Output file units
0 !  12 Output file delimiter
INPUTS 2
6,1 !  FEffMatx:Power to receiver ->Left axis variable
6,4 !  FEffMatx:concentrator field efficiency ->Right axis variable
*** INITIAL INPUT VALUES
Power.kj/hr Efficiency
LABELS 3
"Solar Power"
"Helio efficiency"
"Solar Power"
*** External files
ASSIGN "***.plt" 39
*|?  What file should the online print to?  |1000
*------------------------------------------------------------------------------
* Model "generator" (Type 428)
*
UNIT 21 TYPE 428 generator
*$UNIT_NAME generator
*$MODEL .\STEC Library\Brayton\DLR\generator.tmf
*$POSITION 208 889
*$LAYER Main #
*$# Simple Model to be used with Gas Turbine Components
*$#
*$#
*$#
*$#
PARAMETERS 2
0.99 !  1 Generator efficiency
2 !  2 operating point (mode 1 or 2)
```

```
INPUTS 9
11,3 !  turbine:actual turbine power ->Total turbine power
18,5 !  compressor:actual compressor power ->Compressor shaft work
0,0 !  [unconnected] Fuel Heat flow, Qf
0,0 !  [unconnected] electric output_mode2
0,0 !  [unconnected] relative turbine power_mode2
0,0 !  [unconnected] relative compressor power_mode2
0,0 !  [unconnected] air ratio_mode2
0,0 !  [unconnected] specific minimum air quantity_mode2
0,0 !  [unconnected] cooling air mass ratio
*** INITIAL INPUT VALUES
1 1 1 0 0 0 0 0 0
*------------------------------------------------------------------------------
* Model "flowrate and temp" (Type 65)
*
UNIT 9 TYPE 65 flowrate and temp
*$UNIT_NAME flowrate and temp
*$MODEL .\Output\Online Plotter\Online Plotter With File\No Units\Type65c.tmf
*$POSITION 294 458
*$LAYER Outputs #
PARAMETERS 12
6 !  1 Nb.  of left-axis variables
5 !  2 Nb.  of right-axis variables
0 !  3 Left axis minimum
100000 !  4 Left axis maximum
0 !  5 Right axis minimum
1200 !  6 Right axis maximum
1 !  7 Number of plots per simulation
12 !  8 X-axis gridpoints
0 !  9 Shut off Online w/o removing
40 !  10 Logical Unit for output file
0 !  11 Output file units
0 !  12 Output file delimiter
INPUTS 11
17,4 !  airrec:air outlet mass flow ->Left axis variable-1
44,5 !  STOCO:Storage charge flow ->Left axis variable-2
44,7 !  STOCO:Storage discharge flow ->Left axis variable-3
44,3 !  STOCO:SG input HTF flow ->Left axis variable-4
46,2 !  air flow demand for sg:Instantaneous value of function over the timestep ->Left axis variable-5
FuelX10 !  fuelx10:FuelX10 ->Left axis variable-6
17,1 !  airrec:air outlet temperature ->Right axis variable-1
20,3 !  exhaust silencer:outlet temperature ->Right axis variable-2
49,7 !  concrete_store:concrete temp on top ->Right axis variable-3
49,9 !  concrete_store:concrete temp on bottom ->Right axis variable-4
44,2 !  STOCO:SG input HTF Temp ->Right axis variable-5
*** INITIAL INPUT VALUES
F_air F_charge F_discharge F_SG_use F_demand F_FuelX10 T_RcvrOut T_TurbineOut
T_TopStore T_BotStore T_SG_Use
LABELS 3
"Flow Rates kg hr"
"output temp"
"Flow rates and temp"
*** External files
ASSIGN "flowrateTemp.plt" 40
*|?  What file should the online print to?  |1000
*------------------------------------------------------------------------------
* Model "turbine" (Type 427)
*
UNIT 11 TYPE 427 turbine
*$UNIT_NAME turbine
*$MODEL .\STEC Library\Brayton\DLR\turbine.tmf
*$POSITION 208 639
*$LAYER Main #
PARAMETERS 4
.99 !  1 mechanical efficiency
900 !  2 maximum inlet temperature w/o cooling
1 !  3 ambient pressure
1200 !  4 maximum inlet temperature with cooling
INPUTS 15
42,1 !  combustion chamber:outlet temperature ->temperature combustion air
0,0 !  [unconnected] temperature cooling air
42,4 !  combustion chamber:outlet pressure ->inlet pressure
42,3 !  combustion chamber:outlet mass flow combustion air ->mass flow combustion air
0,0 !  [unconnected] mass flow cooling air
0,0 !  [unconnected] isentropic efficiency
0,0 !  [unconnected] CO2 mass ratio
0,0 !  [unconnected] H2O mass ratio
0,0 !  [unconnected] SO2 mass ratio
0,0 !  [unconnected] air mass ratio
0,0 !  [unconnected] airnitrogen mass ratio
20,1 !  exhaust silencer:relative pressure drop ->relative pressure drop_exhaust silencer
0,0 !  [unconnected] relative pressure drop_heat exchanger_hot side
42,14 !  combustion chamber:outlet enthalpy ->inlet enthalpy working air
```

```
0,0 !  [unconnected] inlet enthalpy cooling air
*** INITIAL INPUT VALUES
1100 350 15 1 0 .9 0 0 0 1 0 .01 0 0 0
*------------------------------------------------------------------------------
* Model "eco_sh" (Type 315)
*
UNIT 33 TYPE 315 eco_sh
*$UNIT_NAME eco_sh
*$MODEL .\STEC Library\Rankine\Steamgen\Eco_sh.tmf
*$POSITION 669 412
*$LAYER Main #
PARAMETERS 6
2 !  1 Counter flow mode
97920 !  2 Overall heat transfer coefficient of exchanger
1 !  3 Reference press loss cold side
18000 !  4 Reference cold side flow
0 !  5 power law exp for UA
0 !  6 power law exp for DP
INPUTS 7
44,2 !  STOCO:SG input HTF Temp ->Hot side inlet temperature
44,3 !  STOCO:SG input HTF flow ->Hot side flow rate
32,3 !  evaporator:Cold side outlet temperature ->Cold side inlet temperature
32,7 !  evaporator:Cold side outlet flow rate ->Cold side flow rate
32,5 !  evaporator:Cold side outlet quality ->Cold side quality
37,2 !  X2H:steam pressure ->cold side outlet pressure
0,0 !  [unconnected] hot side spedific heat
*** INITIAL INPUT VALUES
573.8 129600 20 18000 0.0 1 1.0948
*------------------------------------------------------------------------------
* Model "Temps" (Type 25)
*
UNIT 48 TYPE 25 Temps
*$UNIT_NAME Temps
*$MODEL .\Output\Printer\Unformatted\No Units\Type25c.tmf
*$POSITION 520 693
*$LAYER Outputs #
PARAMETERS 10
STEP ! 1 Printing interval
START ! 2 Start time
STOP ! 3 Stop time
47 !  4 Logical unit
0 !  5 Units printing mode
0 !  6 Relative or absolute start time
-1 !  7 Overwrite or Append
-1 !  8 Print header
0 !  9 Delimiter
1 !  10 Print labels
INPUTS 6
20,3 !  exhaust silencer:outlet temperature ->Input to be printed-1
44,4 !  STOCO:Storage charge Temp ->Input to be printed-2
44,6 !  STOCO:Storage discharge Temp ->Input to be printed-3
0,0 !  [unconnected] Input to be printed-4
0,0 !  [unconnected] Input to be printed-5
44,2 !  STOCO:SG input HTF Temp ->Input to be printed-6
*** INITIAL INPUT VALUES
T_solar_air T_air_top T_air_bottom T_rock_top T_rock_bottom T_use
*** External files
ASSIGN "temps.dat" 47
*|?  Output file for printed results |1000
*------------------------------------------------------------------------------
* Model "Type65c-3" (Type 65)
*
UNIT 13 TYPE 65 Type65c-3
*$UNIT_NAME Type65c-3
*$MODEL .\Output\Online Plotter\Online Plotter With File\No Units\Type65c.tmf
*$POSITION 49 725
*$LAYER Outputs #
PARAMETERS 12
1 !  1 Nb.  of left-axis variables
2 !  2 Nb.  of right-axis variables
0 !  3 Left axis minimum
600 !  4 Left axis maximum
0 !  5 Right axis minimum
20000000 !  6 Right axis maximum
1 !  7 Number of plots per simulation
12 !  8 X-axis gridpoints
0 !  9 Shut off Online w/o removing
41 !  10 Logical Unit for output file
0 !  11 Output file units
0 !  12 Output file delimiter
INPUTS 3
11,1 !  turbine:outlet temperature ->Left axis variable
11,3 !  turbine:actual turbine power ->Right axis variable-1
```

```
21,1 !  generator:Electric output ->Right axis variable-2
*** INITIAL INPUT VALUES
outletTemp TurbinePower ElectricPower
LABELS 3
"Temperatures"
"Power kj/hr"
"Turbine"
*** External files
ASSIGN "***.plt" 41
*|?  What file should the online print to?  |1000
*------------------------------------------------------------------------------
* Model "Power" (Type 25)
*
UNIT 43 TYPE 25 Power
*$UNIT_NAME Power
*$MODEL .\Output\Printer\Unformatted\No Units\Type25c.tmf
*$POSITION 1016 192
*$LAYER Outputs #
PARAMETERS 10
STEP ! 1 Printing interval
START ! 2 Start time
STOP ! 3 Stop time
45 !  4 Logical unit
0 !  5 Units printing mode
0 !  6 Relative or absolute start time
-1 !  7 Overwrite or Append
-1 !  8 Print header
2 !  9 Delimiter
1 !  10 Print labels
INPUTS 4
21,1 !  generator:Electric output ->Input to be printed-1
23,4 !  stage:turbine power ->Input to be printed-2
24,4 !  stage-2:turbine power ->Input to be printed-3
25,4 !  stage-3:turbine power ->Input to be printed-4
*** INITIAL INPUT VALUES
gasTurbPowr SteamTurb1 SteamTurb2 SteamTurb3
*** External files
ASSIGN "power.out" 45
*|?  Output file for printed results |1000
*------------------------------------------------------------------------------
* Model "exhaust silencer" (Type 429)
*
UNIT 20 TYPE 429 exhaust silencer
*$UNIT_NAME exhaust silencer
*$MODEL .\STEC Library\Brayton\DLR\pressure drop.tmf
*$POSITION 308 639
*$LAYER Air Loop #
PARAMETERS 5
.01 !  1 relative pressure drop_design
400 !  2 inlet temperature_design
1.1 !  3 inlet pressure_design
5000 !  4 inlet mass flow_design
1 !  5 design or off-design (1 or 2)
INPUTS 3
11,1 !  turbine:outlet temperature ->inlet temperature
11,7 !  turbine:outlet pressure ->inlet pressure
11,2 !  turbine:outlet mass flow ->inlet mass flow
*** INITIAL INPUT VALUES
20 1 5000
*------------------------------------------------------------------------------
* Model "evaporator" (Type 316)
*
UNIT 32 TYPE 316 evaporator
*$UNIT_NAME evaporator
*$MODEL .\STEC Library\Rankine\Steamgen\Evaporator.tmf
*$POSITION 669 542
*$LAYER Main #
PARAMETERS 6
495000 !  1 overall heat transfer factor
0.0 !  2 blowdown fraction
1 !  3 reference pressure loss
18000 !  4 reference flow rate
0 !  5 power lax exp for UA
0 !  6 power law exp for dp
INPUTS 6
33,1 !  eco_sh:Hot-side outlet temperature ->Hot side inlet temperature
33,2 !  eco_sh:Hot-side flow rate ->Hot side flow rate
35,3 !  eco_sh-2:Cold-side outlet temperature ->Cold side inlet temperature
33,8 !  eco_sh:Cold Side Inlet pressure ->Cold side outlet pressure
35,7 !  eco_sh-2:Cold side Outlet quality ->Cold side inlet quality
0,0 !  [unconnected] Hot side specific heat capacity
*** INITIAL INPUT VALUES
300 130000 100 1 0.0 1.0663
```

```
*------------------------------------------------------------------------------
* Model "X2H" (Type 391)
*
UNIT 37 TYPE 391 X2H
*$UNIT_NAME X2H
*$MODEL .\STEC Library\Rankine\Utility\X2H.TMF
*$POSITION 669 317
*$LAYER Main #
INPUTS 4
33,3 !  eco_sh:Cold-side outlet temperature ->Steam temperture
23,1 !  stage:turbine inlet pressure ->Steam pressure
33,7 !  eco_sh:Cold side Outlet quality ->Steam quality
33,4 !  eco_sh:Cold-side flow rate ->Steam flow rate
*** INITIAL INPUT VALUES
300 20 1 18000
*------------------------------------------------------------------------------
* Model "eco_sh-2" (Type 315)
*
UNIT 35 TYPE 315 eco_sh-2
*$UNIT_NAME eco_sh-2
*$MODEL .\STEC Library\Rankine\Steamgen\Eco_sh.tmf
*$POSITION 669 700
*$LAYER Main #
PARAMETERS 6
2 !  1 Counter flow mode
389880 !  2 Overall heat transfer coefficient of exchanger
1 !  3 Reference press loss cold side
18000 !  4 Reference cold side flow
0 !  5 power law exp for UA
0 !  6 power law exp for DP
INPUTS 7
32,1 !  evaporator:Hot side outlet temperture ->Hot side inlet temperature
32,2 !  evaporator:Hot side outlet flow rate ->Hot side flow rate
39,2 !  preheater:Cold side outlet temperture ->Cold side inlet temperature
39,3 !  preheater:Cold side outlet flow rate ->Cold side flow rate
0,0 !  [unconnected] Cold side quality
32,4 !  evaporator:Cold side inlet pressure ->cold side outlet pressure
0,0 !  [unconnected] hot side spedific heat
*** INITIAL INPUT VALUES
20 100 208 18000 0.0 1 1.0522
*------------------------------------------------------------------------------
* Model "feedw_p" (Type 300)
*
UNIT 31 TYPE 300 feedw_p
*$UNIT_NAME feedw_p
*$MODEL .\STEC Library\Ste\Utility\Saltpu_1.tmf
*$POSITION 909 860
*$LAYER Main #
PARAMETERS 5
21600 !  1 Maximum flow rate
4.19 !  2 Fluid specific heat
1000 !  3 Maximum power
0.0 !  4 Conversion coefficient
0.5 !  5 Power coefficient
INPUTS 3
36,1 !  deaerator:Feed water out temp ->Inlet fluid temperature
36,3 !  deaerator:Feed Water out flow rate ->Inlet mass flow rate
32,6 !  evaporator:Cold side flow rate demand ->desired mass flow rate
*** INITIAL INPUT VALUES
20 100 18000
*------------------------------------------------------------------------------
* Model "Type65c-5" (Type 65)
*
UNIT 41 TYPE 65 Type65c-5
*$UNIT_NAME Type65c-5
*$MODEL .\Output\Online Plotter\Online Plotter With File\No Units\Type65c.tmf
*$POSITION 913 149
*$LAYER Outputs #
PARAMETERS 12
4 !  1 Nb.  of left-axis variables
1 !  2 Nb.  of right-axis variables
0.0 !  3 Left axis minimum
15000000 !  4 Left axis maximum
0.0 !  5 Right axis minimum
100000 !  6 Right axis maximum
1 !  7 Number of plots per simulation
12 !  8 X-axis gridpoints
0 !  9 Shut off Online w/o removing
44 !  10 Logical Unit for output file
0 !  11 Output file units
0 !  12 Output file delimiter
INPUTS 5
23,4 !  stage:turbine power ->Left axis variable-1
```

```
24,4 !  stage-2:turbine power ->Left axis variable-2
25,4 !  stage-3:turbine power ->Left axis variable-3
21,1 !  generator:Electric output ->Left axis variable-4
37,3 !  X2H:steam flow rate ->Right axis variable
*** INITIAL INPUT VALUES
SteamturbPow1 SteamturbPow2 SteamturbPow3 GasturbPow steamFlow
LABELS 3
"Power Kj/Hr"
"SteamFlow kg/hr"
"Turbine"
*** External files
ASSIGN "***.plt" 44
*|?  What file should the online print to?  |1000
*------------------------------------------------------------------------------
* Model "Type65c-4" (Type 65)
*
UNIT 40 TYPE 65 Type65c-4
*$UNIT_NAME Type65c-4
*$MODEL .\Output\Online Plotter\Online Plotter With File\No Units\Type65c.tmf
*$POSITION 697 234
*$LAYER Outputs #
PARAMETERS 12
1 !  1 Nb.  of left-axis variables
1 !  2 Nb.  of right-axis variables
0.0 !  3 Left axis minimum
10000.0 !  4 Left axis maximum
0.0 !  5 Right axis minimum
1000.0 !  6 Right axis maximum
1 !  7 Number of plots per simulation
12 !  8 X-axis gridpoints
0 !  9 Shut off Online w/o removing
43 !  10 Logical Unit for output file
0 !  11 Output file units
0 !  12 Output file delimiter
INPUTS 2
37,1 !  X2H:steam enthalpy ->Left axis variable
37,2 !  X2H:steam pressure ->Right axis variable
*** INITIAL INPUT VALUES
stmEnthal stmPressure
LABELS 3
"Enthalpy kJ/Kg"
"Pressure"
"SteamIntoTurbine"
*** External files
ASSIGN "steam.plt" 43
*|?  What file should the online print to?  |1000
*------------------------------------------------------------------------------
* Model "stage" (Type 318)
*
UNIT 23 TYPE 318 stage
*$UNIT_NAME stage
*$MODEL .\STEC Library\Rankine\Turbine\Stage.tmf
*$POSITION 780 317
*$LAYER Air Loop #
PARAMETERS 8
100 !  1 design inlet pressure
20 !  2 design outlet pressure
17992.8 !  3 design flow rate
0.8 !  4 design inner efficiency
1 !  5 generator efficiency
0.0 !  6 coef.  for inner eff eq
0.0 !  7 b coeff for inner eff
0.0 !  8 c coeff for inner eff
INPUTS 4
26,4 !  S-split:inlet pressure ->Turbine outlet pressure
37,3 !  X2H:steam flow rate ->Turbine inlet flow rate
37,1 !  X2H:steam enthalpy ->Turbine inlet enthalpy
0,0 !  [unconnected] Bypass indicator
*** INITIAL INPUT VALUES
1 18000 2000 1
*------------------------------------------------------------------------------
* Model "stack" (Type 429)
*
UNIT 34 TYPE 429 stack
*$UNIT_NAME stack
*$MODEL .\STEC Library\Brayton\DLR\pressure drop.tmf
*$POSITION 564 828
*$LAYER Air Loop #
PARAMETERS 5
.01 !  1 relative pressure drop_design
0 !  2 inlet temperature_design
0 !  3 inlet pressure_design
0 !  4 inlet mass flow_design
```

```
1 !  5 design or off-design (1 or 2)
INPUTS 3
35,1 !  eco_sh-2:Hot-side outlet temperature ->inlet temperature
0,0 !  [unconnected] inlet pressure
35,2 !  eco_sh-2:Hot-side flow rate ->inlet mass flow
*** INITIAL INPUT VALUES
0 0 0
*------------------------------------------------------------------------------
* Model "type 5b" (Type 5)
*
UNIT 38 TYPE 5 type 5b
*$UNIT_NAME type 5b
*$MODEL .\Heat Exchangers\Counter Flow\TYPE5b.tmf
*$POSITION 286 690
*$LAYER Main # # #
PARAMETERS 4
2 !  1 Counter flow mode
4.18 !  2 Specific heat of hot side fluid
4.18 !  3 Specific heat of cold side fluid
0 !  4 Not used
INPUTS 5
39,4 !  preheater:Hot side outlet temperature ->Hot side inlet temperature
39,5 !  preheater:Hot side outlet flow rate ->Hot side flow rate
31,1 !  feedw_p:Outlet fluid temperature ->Cold side inlet temperature
31,2 !  feedw_p:Outlet flow rate ->Cold side flow rate
0,0 !  [unconnected] Overall heat transfer coefficient of exchanger
*** INITIAL INPUT VALUES
20 0 137 18000 21239.999437
*------------------------------------------------------------------------------
* Model "preheater" (Type 317)
*
UNIT 39 TYPE 317 preheater
*$UNIT_NAME preheater
*$MODEL .\STEC Library\Rankine\preheating\Preheater.tmf
*$POSITION 152 646
*$LAYER Main # # #
PARAMETERS 4
4.18 !  1 cold fluid spef.  heat capacity
186840 !  2 overall heat transfer factor
18000 !  3 cold sid ref flow rate
0 !  4 power law exp for UA
INPUTS 8
26,5 !  S-split:outlet enthalpy 1 ->Hot side inlet enthalpy
26,2 !  S-split:  outlet pressure 1 ->Hot side inlet pressure
38,3 !  type 5b:Cold-side outlet temperature ->Cold side inlet temperature
38,4 !  type 5b:Cold-side flow rate ->Cold side inlet flow rate
0,0 !  [unconnected] Condensate inlet temperature
0,0 !  [unconnected] Condensate inlet flow rate
0,0 !  [unconnected] Condensate inlet quality
0,0 !  [unconnected] on/off
*** INITIAL INPUT VALUES
3000 1 20 18000 100 1 0 1
*------------------------------------------------------------------------------
* Model "S-split" (Type 389)
*
UNIT 26 TYPE 389 S-split
*$UNIT_NAME S-split
*$MODEL .\STEC Library\Rankine\Turbine\S-split.tmf
*$POSITION 866 467
*$LAYER Main #
INPUTS 4
39,1 !  preheater:Demanded hot inlet flow rate ->Demanded Flow Out 1
23,2 !  stage:turbine outlet flowrate ->inlet flow rate
24,1 !  stage-2:turbine inlet pressure ->outlet pressure 2
23,3 !  stage:turbine outlet enthalpy ->inlet enthalpy
*** INITIAL INPUT VALUES
1 1 1 2000
*------------------------------------------------------------------------------
* Model "deaerator" (Type 384)
*
UNIT 36 TYPE 384 deaerator
*$UNIT_NAME deaerator
*$MODEL .\STEC Library\Rankine\preheating\Deaerator.tmf
*$POSITION 1026 862
*$LAYER Main #
INPUTS 8
30,1 !  cond_p:Outlet fluid temperature ->Feed Water in tmp
30,2 !  cond_p:Outlet flow rate ->Feed Water flow rate
27,5 !  S-split-2:outlet enthalpy 1 ->steam inlet enthalpy
27,2 !  S-split-2:  outlet pressure 1 ->steam inlet pressure
38,1 !  type 5b:Hot-side outlet temperature ->condesate inlet temperture
38,2 !  type 5b:Hot-side flow rate ->Condesate inlet flow rate
0,0 !  [unconnected] Condensate inlet quality
```

```
0,0 !  [unconnected] Dearator on/off
*** INITIAL INPUT VALUES
20 18000 3000 1 100 1 0.0 1.0
*------------------------------------------------------------------------------
* Model "stage-2" (Type 318)
*
UNIT 24 TYPE 318 stage-2
*$UNIT_NAME stage-2
*$MODEL .\STEC Library\Rankine\Turbine\Stage.tmf
*$POSITION 948 317
*$LAYER Main #
PARAMETERS 8
20 !  1 design inlet pressure
5 !  2 design outlet pressure
16198.9 !  3 design flow rate
0.8 !  4 design inner efficiency
1 !  5 generator efficiency
0.0 !  6 coef.  for inner eff eq
0.0 !  7 b coeff for inner eff
0.0 !  8 c coeff for inner eff
INPUTS 4
27,4 !  S-split-2:inlet pressure ->Turbine outlet pressure
26,3 !  S-split:outlet flow rate 2 ->Turbine inlet flow rate
26,6 !  S-split:outlet enthalpy 2 ->Turbine inlet enthalpy
0,0 !  [unconnected] Bypass indicator
*** INITIAL INPUT VALUES
1 18000 2000 1
*------------------------------------------------------------------------------
* Model "S-split-2" (Type 389)
*
UNIT 27 TYPE 389 S-split-2
*$UNIT_NAME S-split-2
*$MODEL .\STEC Library\Rankine\Turbine\S-split.tmf
*$POSITION 1039 468
*$LAYER Air Loop #
INPUTS 4
36,2 !  deaerator:required steam flow rate ->Demanded Flow Out 1
24,2 !  stage-2:turbine outlet flowrate ->inlet flow rate
25,1 !  stage-3:turbine inlet pressure ->outlet pressure 2
24,3 !  stage-2:turbine outlet enthalpy ->inlet enthalpy
*** INITIAL INPUT VALUES
1 1 1 2000
*------------------------------------------------------------------------------
* Model "stage-3" (Type 318)
*
UNIT 25 TYPE 318 stage-3
*$UNIT_NAME stage-3
*$MODEL .\STEC Library\Rankine\Turbine\Stage.tmf
*$POSITION 1129 318
*$LAYER Air Loop #
PARAMETERS 8
5 !  1 design inlet pressure
0.056 !  2 design outlet pressure
13197.6 !  3 design flow rate
0.8 !  4 design inner efficiency
0.98 !  5 generator efficiency
0.0 !  6 coef.  for inner eff eq
0.0 !  7 b coeff for inner eff
0.0 !  8 c coeff for inner eff
INPUTS 4
29,2 !  condens:Condensing pressure ->Turbine outlet pressure
27,3 !  S-split-2:outlet flow rate 2 ->Turbine inlet flow rate
27,6 !  S-split-2:outlet enthalpy 2 ->Turbine inlet enthalpy
0,0 !  [unconnected] Bypass indicator
*** INITIAL INPUT VALUES
1 18000 2000 1
*------------------------------------------------------------------------------
* Model "condens" (Type 383)
*
UNIT 29 TYPE 383 condens
*$UNIT_NAME condens
*$MODEL .\STEC Library\Rankine\Condenser\Condens.tmf
*$POSITION 1154 659
*$LAYER Main #
PARAMETERS 2
5.0 !  1 dT Cool water out+condensing temp
10 !  2 temp increase in cool.  water
INPUTS 6
0,0 !  [unconnected] Cooling water inlet temp
25,3 !  stage-3:turbine outlet enthalpy ->steam enthalpy inlet
25,2 !  stage-3:turbine outlet flowrate ->steam mass flow rate
0,0 !  [unconnected] Condensate inlet flow rate
0,0 !  [unconnected] Condensate inlet temperture
```

```
0,0 !  [unconnected] Condensate inlet quality
*** INITIAL INPUT VALUES
20 3000 18000 1000 1 0
*------------------------------------------------------------------------------
* Model "cond_p" (Type 300)
*
UNIT 30 TYPE 300 cond_p
*$UNIT_NAME cond_p
*$MODEL .\STEC Library\Ste\Utility\Saltpu_1.tmf
*$POSITION 1154 861
*$LAYER Air Loop #
PARAMETERS 5
21600 !  1 Maximum flow rate
4.19 !  2 Fluid specific heat
1000 !  3 Maximum power
0.0 !  4 Conversion coefficient
0.5 !  5 Power coefficient
INPUTS 3
29,1 !  condens:Condesing Temperature ->Inlet fluid temperature
29,6 !  condens:Condesate flow rate ->Inlet mass flow rate
0,0 !  [unconnected] desired mass flow rate
*** INITIAL INPUT VALUES
20 18000 -1
*------------------------------------------------------------------------------
* Model "concrete_store" (Type 430)
*
UNIT 49 TYPE 430 concrete_store
*$UNIT_NAME concrete_store
*$MODEL .\STEC Library\Storage\concrete_store.tmf
*$POSITION 477 567
*$LAYER Air Loop #
*$# Parallel tubes in concrete block for thermal storage in solar plants driven by HTF (oil, water).
*$# Based on Standard TRNSYS Type10 Rockbed Thermal Storage by JWT.
*$#
*$#
*$#
*$#
PARAMETERS 16
1.04 !  1 HTF specific heat
1 !  2 HTF density
2 !  3 total cross sec area of pipes
40 !  4 Length of storage
.9638 !  5 concrete specific heat
1000000 !  6 concrete total mass
3000000 !  7 overall heat transfer coefficient at reference flow rate
1000 !  8 overall loss coefficient
13 !  9 number of nodes
50000 !  10 reference flow rate
0.6454905 !  11 ak0 parameter for scaling of heat transfer coeff.
2.255832 !  12 ak1
-6.842885 !  13 ak2
10.86112 !  14 ak3
-8.5377 !  15 ak4
2.618488 !  16 ak5
INPUTS 5
44,4 !  STOCO:Storage charge Temp ->HTF temp entering on top
44,5 !  STOCO:Storage charge flow ->HTF flow rate entering on top
44,6 !  STOCO:Storage discharge Temp ->HTF temp entering at bottom
44,7 !  STOCO:Storage discharge flow ->HTF flow rate entering at bottom
0,0 !  [unconnected] ambient Temperatur
*** INITIAL INPUT VALUES
15 500 15 500 25
DERIVATIVES 1
25 !  1 initial Temperatures in segment
*------------------------------------------------------------------------------
* Model "Type22" (Type 22)
*
UNIT 50 TYPE 22 Type22
*$UNIT_NAME Type22
*$MODEL .\Controllers\Iterative Feedback Controller\Type22.tmf
*$POSITION 489 160
*$LAYER Air Loop # Controls #
PARAMETERS 2
0 !  1 mode
0 !  2 Maximum number of oscillations
INPUTS 7
0,0 !  [unconnected] Setpoint
17,1 !  airrec:air outlet temperature ->Controlled variable
0,0 !  [unconnected] On / Off signal
0,0 !  [unconnected] Minimum control signal
0,0 !  [unconnected] Maximum control signal
0,0 !  [unconnected] Threshold for non-zero output
0,0 !  [unconnected] Tolerance on tracking error
```

```
*** INITIAL INPUT VALUES
1000 0 1 -50000 0 0 0
*------------------------------------------------------------------------------
* EQUATIONS "Compressor switch"
*
EQUATIONS 1
CompressorOn = gt ( [6,1], 10000000)*50000
*$UNIT_NAME Compressor switch
*$LAYER Main
*$POSITION 178 170
*------------------------------------------------------------------------------
* EQUATIONS "Control Inverter"
*
EQUATIONS 1
InvertedSigOut = -1*[50,1]
*$UNIT_NAME Control Inverter
*$LAYER Main
*$POSITION 416 42
*------------------------------------------------------------------------------
* Model "combustion chamber" (Type 426)
*
UNIT 42 TYPE 426 combustion chamber
*$UNIT_NAME combustion chamber
*$MODEL .\STEC Library\Brayton\DLR\combustion chamber.tmf
*$POSITION 151 358
*$LAYER Main # Air Loop #
PARAMETERS 14
2 !  1 operating mode (mode 1 or 2)
47600 !  2 lower calorific value
0.7318 !  3 C mass ratio
0.2341 !  4 H2 mass ratio
0 !  5 S mass ratio
0.0159 !  6 N2 mass ratio
0.0182 !  7 O2 mass ratio
0 !  8 H2O mass ratio_in
0 !  9 ashes mass ratio
0.04 !  10 relative pressure drop_design
3 !  11 design or off-design (mode 3 or 4)
0 !  12 inlet temperature_off-design if mode 4
0 !  13 inlet pressure_off-design if mode 4
0 !  14 inlet mass flow_off-design if mode 4
INPUTS 6
17,1 !  airrec:air outlet temperature ->inlet air temperature
17,4 !  airrec:air outlet mass flow ->inlet air flow rate
0,0 !  [unconnected] fuel flow rate if mode 1
0,0 !  [unconnected] outlet temperature if mode 2
17,2 !  airrec:air outlet pressure ->inlet pressure
17,3 !  airrec:air outlet enthalpy ->inlet enthalpy
*** INITIAL INPUT VALUES
600 1980000 0 1200 12 0
*------------------------------------------------------------------------------
* EQUATIONS "fuelx10"
*
EQUATIONS 1
FuelX10 = [42,2]*10
*$UNIT_NAME fuelx10
*$LAYER Main
*$POSITION 217 405
*------------------------------------------------------------------------------
END
```

VIII. APPENDIX B CALLTRNSYS MEX FUNCTION

```
/*==================================================================
*
* RUNTRNSYS.C program to run trnsys , to insert certain constants into a deck
* file, and to retrieve results of run from output file
*
* The calling syntax is:
*
* [outputvector] = runtrnsys(deck filepath, trnsyspath, deckfilename,
* outputfilename, NamedPairsOfInputs)
*
sample call is
outresults = runtrnsys('C:\Trnsys16\MyProjects\TestProject','C:\Trnsys16\Exe\TRNExe.exe','junkTestDeck.dck','STEC_CYC.out','AIRFLOW',123,'GASTEM
NOTE: move trnsys to a path with no spaces, because of windows command shell limits of handling multiple quotes in a command line
The named pairs of inputs come in the order, Name1, numeric value1, Name2, Numeric Value 2.. etc
There is no limit on how many inputs you can have
There is a limit of 100 outputs.  the outputs are all the numeric values from the first line of the output file
---
```

```
*
* This is a MEX-file for MATLAB.
* Copyright 2009 Paul Harper Light Year CC
*
*=============================================================*/
/* $Revision:  1.10 $ */
#include <math.h>
#include "mex.h"
/* Input Arguments */
#define NUMSTRINGPATHS 4 // first 4 args are string paths
#define MAXOUTPUTS 100 // max number that we can handle from trnsys output file
// define this to get debug printing
// #define DEBUGP
#ifdef DEBUGP
#define DPRN(x) x
#else
#define DPRN(x)
#endif
#if !defined(MAX)
#define MAX(A, B) ((A) > (B) ? (A) : (B))
#endif
#if !defined(MIN)
#define MIN(A, B) ((A) < (B) ? (A) : (B))
#endif
//-----------------------------------------------------------------------------------
// since lcc's strchr seems to be broken (return type is int)
char *mystrchr(const char *str, int ch)
{
char *ret = NULL;
char *pch = str;
while(*pch) {
if(*pch == ch) {
ret= pch;
break;
}
pch++;
}
return ret;
}
//-----------------------------------------------------------------------------------
// appends the trailing zero
char *mystrncpy(char *strout, const char *str, int numchr)
{
strncpy(strout, str, numchr);
strout[numchr] = '\0';
return strout;
}
//-----------------------------------------------------------------------------------
// append a path separator char if path does not have one
char *appendsep(char *path)
{
if(path[strlen(path)-1] != '\\') {
strcat(path, "\\");
}
return path;
}
//-----------------------------------------------------------------------------------
// replace the constants in the deck file with these ones
char * insertconst(char *deckpath, char *deckfile, int numconst, char ** constnames,
double *constvalues)
{
char linebuf[1024];
char fulldeckfile[256];
static char fulltmpdeckfile[256];
FILE *fh, *fhtmp;
int constfound = 0;
char *equals = NULL;
char *retval = NULL;
// open temp deck file
strcpy(fulltmpdeckfile, deckpath);
appendsep(fulltmpdeckfile);
strcat(fulltmpdeckfile, deckfile);
strcat(fulltmpdeckfile, "zap");
fhtmp = fopen(fulltmpdeckfile,"w");
if(fhtmp == NULL) {
sprintf(linebuf, "Unable to open temp deck file for writing %s", fulltmpdeckfile);
mexErrMsgTxt(linebuf);
return retval;
}
// now open the deck file, read it line by line, and copy to an output file, replacing the constants as we go
// we replace the second occurrance of the CONSTANTS statement
strcpy(fulldeckfile, deckpath);
appendsep(fulldeckfile);
```

```
strcat(fulldeckfile, deckfile);
fh = fopen(fulldeckfile,"r");
if(fh == NULL) {
sprintf(linebuf, "Unable to open deck file %s", fulldeckfile);
mexErrMsgTxt(linebuf);
return retval;
} else {
while(fgets(linebuf, sizeof(linebuf), fh)) {
// mexPrintf("line from deck file------>%s", linebuf);
if(strncmp(linebuf, "CONSTANTS ", 10) == 0) {
constfound++;
}
if(constfound == 2) {
// we might be dealing with constants
if(strncmp(linebuf, "*", 1) == 0) {
// a comment ..end of constants ..  just copy rest of input file across
constfound = 99;
} else {
equals = mystrchr(linebuf, '=');
// mexPrintf("linebuf is %d equals is %d\n", linebuf, equals);
if(equals != NULL) {
char constname[100];
int constnamelen = equals - linebuf;
int j;
mystrncpy(constname, linebuf,MIN(constnamelen, sizeof(constname)));
// check if this constant matches one in our list
for(j = 0; j < numconst; j++) {
// mexPrintf("comparing %s to incoming:%s\n", constname, constnames[j]);
if(strcmp(constname, constnames[j]) == 0) {
// we have a match
// create a new constant line
sprintf(linebuf, "%s=%d\n", constname, (int)(constvalues[j]));
DPRN(mexPrintf("creating new line in const file:%s", linebuf));
}
}
}
}
}
// copy the linebuf(maybe altered) across to the temp file
fputs(linebuf, fhtmp);
}
fclose(fh);
fclose(fhtmp);
retval = fulltmpdeckfile;
// replace the original deck file with the temp file,
// no ..  now preserve the original deck, and run with the new tmp deck
// unlink(fulldeckfile);
// rename(fulltmpdeckfile, fulldeckfile);
}
return retval;
}
//-----------------------------------------------------------------------------------------
// execute a systems command to change to the deck directory, then to execute trnsys with the given deck
// Note that because of windows limitations trnsys should be on a path with no spaces, i.e.  'program files' is a no-no
int runtrn(char *deckpath, char *trnpath, char *fulltempdeckfile, char *outputfile, double *outputs)
{
char syscmd[512];
char linebuf[1024];
char fulloutputfile[256];
char msgbuf[256];
FILE *fh;
int numoutputs = 0;
// char fulldeckfile[256];
// strcpy(fulldeckfile, deckpath);
// appendsep(fulldeckfile);
// strcat(fulldeckfile, deckfile);
// change directory to trnsys and run the program
//sprintf(syscmd, "cmd /C \" cd %s && %s %s /h \" ", deckpath, trnpath, deckfile);
// try simpler exec, without starting up a cmd shell
sprintf(syscmd, "%s %s /h ", trnpath, fulltempdeckfile);
DPRN(mexPrintf("------> System command=%s\n", syscmd));
system(syscmd);
// exec(syscmd);
// now open the output file and read it line by line
// assume we have just one line in the output file, and it just consists of a number of floating point variables
// i.e we use type 25 (printer) and dontprinter headers, and dont append to the log file
strcpy(fulloutputfile, deckpath);
appendsep(fulloutputfile);
strcat(fulloutputfile, outputfile);
fh = fopen(fulloutputfile,"r");
if(fh == NULL) {
sprintf(msgbuf, "Unable to open output file %s", fulloutputfile);
mexErrMsgTxt(msgbuf);
```

```
} else {
while(fgets(linebuf, sizeof(linebuf), fh)) {
char *maybenumber = linebuf;
char * comma = mystrchr(linebuf, ',');
while(comma != NULL) {
outputs[numoutputs] = atof(maybenumber);
DPRN(mexPrintf("Output number %d is %f\n", numoutputs, outputs[numoutputs]);)
numoutputs++;
if(numoutputs >= MAXOUTPUTS) {
sprintf(msgbuf, "Too many variables in output file %s.  Allowed a max of %d variables",
fulloutputfile, MAXOUTPUTS);
mexErrMsgTxt(msgbuf);
}
maybenumber = comma + 1;
comma = mystrchr(maybenumber, ',');
}
DPRN(mexPrintf("line from output file------>%s", linebuf));
DPRN(mexPrintf("line contains %d variables\n", numoutputs));
}
fclose(fh);
}
return numoutputs;
}
//-------------------------------------------------------------------------------------
// create a string from the jth input paramater
char *getStringFromrhs(int j, const mxArray*prhs[])
{
char errmsg[100];
int buflen;
char *input_buf;
int status;
/* Input must be a string.  */
if ( mxIsChar(prhs[j]) != 1) {
sprintf(errmsg, "Input %d must be a string.", j);
mexErrMsgTxt(errmsg);
}
/* Input must be a row vector.  */
if (mxGetM(prhs[j])!=1) {
sprintf(errmsg, "Input %d must be a row vector.", j);
mexErrMsgTxt(errmsg);
}
/* Get the length of the input string.  */
buflen = (mxGetM(prhs[j]) * mxGetN(prhs[j])) + 1;
/* Allocate memory for input and output strings.  */
input_buf=mxCalloc(buflen, sizeof(char));
/* Copy the string data from prhs[0] into a C string
* input_ buf.
* If the string array contains several rows, they are copied,
* one column at a time, into one long string array.
*/
status = mxGetString(prhs[j], input_buf, buflen);
if(status != 0) {
mexWarnMsgTxt("Not enough space.  String is truncated.");
}
return input_buf;
}
//-------------------------------------------------------------------------------------
// main entry point for dll, this is where matlab enters and leaves the dll
void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const mxArray*prhs[] )
{
char *deckpath, *trnpath, *deckfile, *outputfile;
int j;
char *input_buf;
/* Check for proper number of arguments */
if (nrhs < NUMSTRINGPATHS) {
mexErrMsgTxt("Need at least 4 input arguments .");
} else if (nlhs > 1) {
mexErrMsgTxt("Too many output arguments.");
}
// first NUMSTRINGPATHS args must be strings
for(j = 0; j < NUMSTRINGPATHS; j++) {
input_buf = getStringFromrhs(j, prhs);
switch (j) {
case 0:  deckpath = input_buf; DPRN(mexPrintf("deck path=%s\n",deckpath)); break;
case 1:  trnpath = input_buf; DPRN(mexPrintf("trn path=%s\n",trnpath)); break;
case 2:  deckfile = input_buf; DPRN(mexPrintf("deck file=%s\n",deckfile)); break;
case 3:  outputfile = input_buf; DPRN(mexPrintf("outputfile=%s\n",outputfile)); break;
}
}
{
// now subsequent arguments must be pairs of STRING, double, eg.  'AIRFLOW', 150000, 'GASTEMP', 300
int numextrapairs = (nrhs - NUMSTRINGPATHS)/2;
char **constnames = mxCalloc(numextrapairs, sizeof(char *));
```

```
double * constvalues = (double *)mxCalloc(numextrapairs, sizeof(double));
char * fulltempdeckfile;
int jconstants = 0;
for(j = NUMSTRINGPATHS; j < nrhs; j += 2) {
constnames[jconstants] = getStringFromrhs(j, prhs);
if (!mxIsDouble(prhs[j+1]) || mxIsComplex(prhs[j+1]) ||
(mxGetM(prhs[j+1]) * mxGetN(prhs[j+1])) != 1) {
char errmsg[100];
sprintf(errmsg, "argument number %d must be a double scalar", j+1);
mexErrMsgTxt(errmsg);
}
constvalues[jconstants] = mxGetScalar(prhs[j+1]);;
jconstants++;
}
fulltempdeckfile = insertconst(deckpath, deckfile, numextrapairs, constnames, constvalues);
if(fulltempdeckfile == NULL)
{
char errmsg[100];
sprintf(errmsg, "Tempdeck file not created");
mexErrMsgTxt(errmsg);
}
else
{
double outputs[MAXOUTPUTS]; // we
double *z;
int numoutputs = runtrn(deckpath, trnpath, fulltempdeckfile, outputfile, outputs);
// int numoutputs = 5;
// now pack the outputs into the lhs
/* Set the output pointer to the output matrix.  */
plhs[0] = mxCreateDoubleMatrix(1, numoutputs, mxREAL);
/* Create a C pointer to a copy of the output matrix.  */
z = mxGetPr(plhs[0]);
// copy the output across
for(j = 0; j< numoutputs; j++) {
z[j] = outputs[j];
DPRN(mexPrintf("backin Mexfunction Output number %d is %f\n", j, outputs[j]);)
}
}
}
return;
}
/*=================================================================
*
* RUNTRNSYS.C program to run trnsys , to insert certain constants into a deck
* file, and to retrieve results of run from output file
*
* The calling syntax is:
*
* [outputvector] = runtrnsys(deck filepath, trnsyspath, deckfilename,
* outputfilename, NamedPairsOfInputs)
*
sample call is
outresults = runtrnsys('C:\Trnsys16\MyProjects\TestProject','C:\Trnsys16\Exe\TRNExe.exe','junkTestDeck.dck','STEC_CYC.out','AIRFLOW',123,'GASTEM
NOTE: move trnsys to a path with no spaces, because of windows command shell limits of handling multiple quotes in a command line
The named pairs of inputs come in the order, Name1, numeric value1, Name2, Numeric Value 2..  etc
There is no limit on how many inputs you can have
There is a limit of 100 outputs.  the outputs are all the numeric values from the first line of the output file
---
*
* This is a MEX-file for MATLAB.
* Copyright 2009 Paul Harper Light Year CC
*
*=================================================================*/
/* $Revision:  1.10 $ */
#include <math.h>
#include "mex.h"
/* Input Arguments */
#define NUMSTRINGPATHS 4 // first 4 args are string paths
#define MAXOUTPUTS 100 // max number that we can handle from trnsys output file
// define this to get debug printing
// #define DEBUGP
#ifdef DEBUGP
#define DPRN(x) x
#else
#define DPRN(x)
#endif
#if !defined(MAX)
#define MAX(A, B) ((A) > (B) ? (A) : (B))
#endif
#if !defined(MIN)
#define MIN(A, B) ((A) < (B) ? (A) : (B))
#endif
//-------------------------------------------------------------------------------------
```

```
// since lcc's strchr seems to be broken (return type is int)
char *mystrchr(const char *str, int ch)
{
char *ret = NULL;
char *pch = str;
while(*pch) {
if(*pch == ch) {
ret= pch;
break;
}
pch++;
}
return ret;
}
//------------------------------------------------------------------------------------
// appends the trailing zero
char *mystrncpy(char *strout, const char *str, int numchr)
{
strncpy(strout, str, numchr);
strout[numchr] = '\0';
return strout;
}
//------------------------------------------------------------------------------------
// append a path separator char if path does not have one
char *appendsep(char *path)
{
if(path[strlen(path)-1] != '\\') {
strcat(path, "\\");
}
return path;
}
//------------------------------------------------------------------------------------
// replace the constants in the deck file with these ones
char * insertconst(char *deckpath, char *deckfile, int numconst, char ** constnames,
double *constvalues)
{
char linebuf[1024];
char fulldeckfile[256];
static char fulltmpdeckfile[256];
FILE *fh, *fhtmp;
int constfound = 0;
char *equals = NULL;
char *retval = NULL;
// open temp deck file
strcpy(fulltmpdeckfile, deckpath);
appendsep(fulltmpdeckfile);
strcat(fulltmpdeckfile, deckfile);
strcat(fulltmpdeckfile, "zap");
fhtmp = fopen(fulltmpdeckfile,"w");
if(fhtmp == NULL) {
sprintf(linebuf, "Unable to open temp deck file for writing %s", fulltmpdeckfile);
mexErrMsgTxt(linebuf);
return retval;
}
// now open the deck file, read it line by line, and copy to an output file, replacing the constants as we go
// we replace the second occurrance of the CONSTANTS statement
strcpy(fulldeckfile, deckpath);
appendsep(fulldeckfile);
strcat(fulldeckfile, deckfile);
fh = fopen(fulldeckfile,"r");
if(fh == NULL) {
sprintf(linebuf, "Unable to open deck file %s", fulldeckfile);
mexErrMsgTxt(linebuf);
return retval;
} else {
while(fgets(linebuf, sizeof(linebuf), fh)) {
// mexPrintf("line from deck file------>%s", linebuf);
if(strncmp(linebuf, "CONSTANTS ", 10) == 0) {
constfound++;
}
if(constfound == 2) {
// we might be dealing with constants
if(strncmp(linebuf, "*", 1) == 0) {
// a comment ..end of constants ..  just copy rest of input file across
constfound = 99;
} else {
equals = mystrchr(linebuf, '=');
// mexPrintf("linebuf is %d equals is %d\n", linebuf, equals);
if(equals != NULL) {
char constname[100];
int constnamelen = equals - linebuf;
int j;
mystrncpy(constname, linebuf,MIN(constnamelen, sizeof(constname)));
// check if this constant matches one in our list
for(j = 0; j < numconst; j++) {
// mexPrintf("comparing %s to incoming:%s\n", constname, constnames[j]);
if(strcmp(constname, constnames[j]) == 0) {
// we have a match
// create a new constant line
sprintf(linebuf, "%s=%d\n", constname, (int)(constvalues[j]));
DPRN(mexPrintf("creating new line in const file:%s", linebuf));
}
}
}
}
}
// copy the linebuf(maybe altered) across to the temp file
fputs(linebuf, fhtmp);
}
fclose(fh);
fclose(fhtmp);
retval = fulltmpdeckfile;
// replace the original deck file with the temp file,
// no ..  now preserve the original deck, and run with the new tmp deck
// unlink(fulldeckfile);
// rename(fulltmpdeckfile, fulldeckfile);
}
return retval;
}
//-------------------------------------------------------------------------------------
// execute a systems command to change to the deck directory, then to execute trnsys with the given deck
// Note that because of windows limitations trnsys should be on a path with no spaces, i.e.  'program files' is a no-no
int runtrn(char *deckpath, char *trnpath, char *fulltempdeckfile, char *outputfile, double *outputs)
{
char syscmd[512];
char linebuf[1024];
char fulloutputfile[256];
char msgbuf[256];
FILE *fh;
int numoutputs = 0;
// char fulldeckfile[256];
// strcpy(fulldeckfile, deckpath);
// appendsep(fulldeckfile);
// strcat(fulldeckfile, deckfile);
// change directory to trnsys and run the program
//sprintf(syscmd, "cmd /C \" cd %s && %s %s /h \" ", deckpath, trnpath, deckfile);
// try simpler exec, without starting up a cmd shell
sprintf(syscmd, "%s %s /h ", trnpath, fulltempdeckfile);
DPRN(mexPrintf("------> System command=%s\n", syscmd));
system(syscmd);
// exec(syscmd);
// now open the output file and read it line by line
// assume we have just one line in the output file, and it just consists of a number of floating point variables
// i.e we use type 25 (printer) and dontprinter headers, and dont append to the log file
strcpy(fulloutputfile, deckpath);
appendsep(fulloutputfile);
strcat(fulloutputfile, outputfile);
fh = fopen(fulloutputfile,"r");
if(fh == NULL) {
sprintf(msgbuf, "Unable to open output file %s", fulloutputfile);
mexErrMsgTxt(msgbuf);
} else {
while(fgets(linebuf, sizeof(linebuf), fh)) {
char *maybenumber = linebuf;
char * comma = mystrchr(linebuf, ',');
while(comma != NULL) {
outputs[numoutputs] = atof(maybenumber);
DPRN(mexPrintf("Output number %d is %f\n", numoutputs, outputs[numoutputs]);)
numoutputs++;
if(numoutputs >= MAXOUTPUTS) {
sprintf(msgbuf, "Too many variables in output file %s.  Allowed a max of %d variables",
fulloutputfile, MAXOUTPUTS);
mexErrMsgTxt(msgbuf);
}
maybenumber = comma + 1;
comma = mystrchr(maybenumber, ',');
}
DPRN(mexPrintf("line from output file------>%s", linebuf));
DPRN(mexPrintf("line contains %d variables\n", numoutputs));
}
fclose(fh);
}
return numoutputs;
}
//-------------------------------------------------------------------------------------
// create a string from the jth input paramater
```

```
char *getStringFromrhs(int j, const mxArray*prhs[])
{
char errmsg[100];
int buflen;
char *input_buf;
int status;
/* Input must be a string.  */
if ( mxIsChar(prhs[j]) != 1) {
sprintf(errmsg, "Input %d must be a string.", j);
mexErrMsgTxt(errmsg);
}
/* Input must be a row vector.  */
if (mxGetM(prhs[j])!=1) {
sprintf(errmsg, "Input %d must be a row vector.", j);
mexErrMsgTxt(errmsg);
}
/* Get the length of the input string.  */
buflen = (mxGetM(prhs[j]) * mxGetN(prhs[j])) + 1;
/* Allocate memory for input and output strings.  */
input_buf=mxCalloc(buflen, sizeof(char));
/* Copy the string data from prhs[0] into a C string
* input_ buf.
* If the string array contains several rows, they are copied,
* one column at a time, into one long string array.
*/
status = mxGetString(prhs[j], input_buf, buflen);
if(status != 0) {
mexWarnMsgTxt("Not enough space.  String is truncated.");
}
return input_buf;
}
//---------------------------------------------------------------------------------------
// main entry point for dll, this is where matlab enters and leaves the dll
void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const mxArray*prhs[] )
{
char *deckpath, *trnpath, *deckfile, *outputfile;
int j;
char *input_buf;
/* Check for proper number of arguments */
if (nrhs < NUMSTRINGPATHS) {
mexErrMsgTxt("Need at least 4 input arguments .");
} else if (nlhs > 1) {
mexErrMsgTxt("Too many output arguments.");
}
// first NUMSTRINGPATHS args must be strings
for(j = 0; j < NUMSTRINGPATHS; j++) {
input_buf = getStringFromrhs(j, prhs);
switch (j) {
case 0:  deckpath = input_buf; DPRN(mexPrintf("deck path=%s\n",deckpath)); break;
case 1:  trnpath = input_buf; DPRN(mexPrintf("trn path=%s\n",trnpath)); break;
case 2:  deckfile = input_buf; DPRN(mexPrintf("deck file=%s\n",deckfile)); break;
case 3:  outputfile = input_buf; DPRN(mexPrintf("outputfile=%s\n",outputfile)); break;
}
}
{
// now subsequent arguments must be pairs of STRING, double, eg.  'AIRFLOW', 150000, 'GASTEMP', 300
int numextrapairs = (nrhs - NUMSTRINGPATHS)/2;
char **constnames = mxCalloc(numextrapairs, sizeof(char *));
double * constvalues = (double *)mxCalloc(numextrapairs, sizeof(double));
char * fulltempdeckfile;
int jconstants = 0;
for(j = NUMSTRINGPATHS; j < nrhs; j += 2) {
constnames[jconstants] = getStringFromrhs(j, prhs);
if (!mxIsDouble(prhs[j+1]) || mxIsComplex(prhs[j+1]) ||
(mxGetM(prhs[j+1]) * mxGetN(prhs[j+1])) != 1) {
char errmsg[100];
sprintf(errmsg, "argument number %d must be a double scalar", j+1);
mexErrMsgTxt(errmsg);
}
constvalues[jconstants] = mxGetScalar(prhs[j+1]);;
jconstants++;
}
fulltempdeckfile = insertconst(deckpath, deckfile, numextrapairs, constnames, constvalues);
if(fulltempdeckfile == NULL)
{
char errmsg[100];
sprintf(errmsg, "Tempdeck file not created");
mexErrMsgTxt(errmsg);
}
else
{
double outputs[MAXOUTPUTS]; // we
double *z;
int numoutputs = runtrn(deckpath, trnpath, fulltempdeckfile, outputfile, outputs);
// int numoutputs = 5;
// now pack the outputs into the lhs
/* Set the output pointer to the output matrix.  */
plhs[0] = mxCreateDoubleMatrix(1, numoutputs, mxREAL);
/* Create a C pointer to a copy of the output matrix.  */
z = mxGetPr(plhs[0]);
// copy the output across
for(j = 0; j< numoutputs; j++) {
z[j] = outputs[j];
DPRN(mexPrintf("backin Mexfunction Output number %d is %f\n", j, outputs[j]);)
}
}
}
return;
}
```

Acknowledgments

Thanks to Tom Fluri and Theo Von Backstrom for many helpful discussions.

[Kehlhofer 1999] Kehlhofer, R., Nielson, H., Warner, J., Bachmann, R. : “Combined-cycle gas & steam turbine power plants”, Chapter 4, PennWell Books, 1999.

[Garcia 2006] Garcia, P.:”Tools for design and performance analysis of future concentrated solar power plants, Application to the PEGASE project ”, SOLLAB Doctoral-Colloquium October17th, 2006