Available online at www.sciencedirect.com

ScienceDirect

Energy Procedia 49 (2014) 2482 – 2489

Energy Procedia

SolarPACES 2013

Advances in CSP simulation technology in the System Advisor Model

A. Dobos[a], T. Neises[a], M. Wagner[a]

[a]*National Renewable Energy Laboratory, 15031 Denver West Parkway, Golden, CO, 80401*

Abstract

The System Advisor Model (SAM) is modeling software for renewable energy systems developed by the National Renewable Energy Laboratory (NREL).SAM combines annual time series power production models with financial models to estimate the levelized cost of energy (LCOE) and other metrics for renewable energy projects. To date, SAM has utilized the general purpose commercial TRNSYS transient systems modeling software package for CSP simulations and originally PV and wind. To achieve: (1) significantly faster model performance, (2) easy parallelization of concurrent simulations to take advantage of modern multi-core processor desktop computers, (3) to allow straightforward modification of CSP component models in the SAM environment, and (4) ability to include CSP technologies in the SAM Software Development Kit (SDK), NREL has undertaken to reformulate the CSP models into a new transient simulation framework written in C++, by NREL. This framework is tailored specifically for use in SAM and not for general purpose modeling like TRNSYS. Preliminary results show excellent matching with the accepted TRNSYS-based models, as well as an order of magnitude reduction in simulation time for certain models. These runtime reductions enable larger scale plant configuration analysis, as well as grid-integration studies that require many thousands of simulations.

Selection and peer review by the scientific conference committee of SolarPACES 2013 under responsibility of PSE AG.
Final manuscript published as received without editorial corrections.
Keywords: systems modeling, concentrating solar power simulation,

1. Introduction

The System Advisor Model (SAM) is free software developed by the National Renewable Energy Laboratory that predicts hourly energy production for renewable energy systems. Technologies represented in SAM include photovoltaics (flat-plate and concentrating), concentrating solar power (parabolic troughs, towers, linear Fresnel, dish-Stirling), solar water heating, wind, geothermal, and biomass. Hourly performance models for PV, wind, geothermal, and biomass plants are relatively straightforward computationally, as a series of submodels are executed in sequence to calculate outputs given weather data inputs and system parameters. For concentrating solar thermal models, however, the solution techniques are not so simple. These systems are represented by interconnected individual components such as solar collectors, receivers, heat exchangers, piping, storage systems, and power

Selection and peer review by the scientific conference committee of SolarPACES 2013 under responsibility of PSE AG.
Final manuscript published as received without editorial corrections.
doi:10.1016/j.egypro.2014.03.263

cycles. Each component cannot be sequentially modeled because pressures, mass flow rates and temperatures at the interfaces between components must match, and energy and mass must be conserved among the piping loops and feedback systems that exist within the system design. Consequently, iterative numerical solutions are applied such that at each time step of the simulation, the system representation has "converged" to physically sensible values at each point.

To date, SAM has utilized the general-purpose commercial TRNSYS tool for modeling concentrating solar thermal power plant systems [1]. TRNSYS is a well-established software package that has been in development since the mid 1970s, and the simulation core and engineering component code is written entirely in FORTRAN and focused at engineers doing desktop system analysis. The software consists of a small kernel that iteratively calls individual system component models many times until the overall system has converged, as well as an extensive library of components such as building models, HVAC systems, geothermal heat pumps, pipes, tanks with heaters, solar collectors, PV systems, and related equipment. In previous version of SAM dating back to 2007, custom CSP component models have been implemented in FORTRAN in adherence to the TRNSYS conventions. SAM includes only the TRNSYS kernel and component models necessary for CSP modeling at this time. These models have proven to be reliable and capable predictors of system performance and have been validated and utilized in the literature.

The motivation to reconsider the use of TRNSYS within SAM has been driven by factors driving the need for simulating very large scenarios that may require thousands of simulations with different input parameters. To achieve these results in a permissible amount of time, it is highly advantageous to effectively utilize modern desktop computer processors that may have up to eight individual cores, as well as provide a software framework that can be deployed on distributing computing systems or dispatched over the internet. There is also a need to support the SAM Software Development Kit (SDK) which does not currently allow users to access CSP technologies through the SAM engine. This paper focuses on the recent advancements in the technology used to model the complex solar thermal systems to achieve the aforementioned goals of performance, portability, and parallelism. Preliminary results from the new CSP simulation core software show excellent agreement with the accepted outputs of the TRNSYS versions, as well as significant improvements in simulation speed.

2. Implementation

This section details the implementation of a new solver kernel for calculating the performance of concentrating solar thermal systems. The kernel is henceforth referred to as "TCS", a loose acronym denoting the *transient component simulation* purpose of the tool.

2.1. Kernel structure

TCS is a transient physical system simulation tool at whose core is an iterative successive-substitution solving engine. Each unique physical system component is known as a *type*, and instances of types are *units* (this follows the TRNSYS convention). A system consists of a set of n units ordered $0..n$-1. It is allowed to have multiple units of the same type in a system. The order in which units are defined in the system is the same order that the iterative solver calls each type at each time step. A type is essentially a compiled subroutine that calculates the values of output variables from input variables.

Each type defines a specific set of input and output variables. Each variable is given at compile-time a data type, index, label, units, description, optional metadata, and optional default value. Variables can be numbers, one dimensional arrays, two dimensional matrices, or strings. There is no defined limit on the number of variables, or size of arrays and matrices. Information about a type's input and output variables can be dynamically queried through the TCS application programming interface (API).

A simulation progresses with a constant time step from a specified start time to a specified end time. The internal time unit in TCS is the second. At each time step, TCS calls each unit in the given calling sequence. After each unit is called, TCS checks to see if any of the outputs are connected to inputs of other units. If so, TCS propagates the output value to the input value, marking the unit associated with the input for iteration if the previous input value

was outside the specified tolerance for the particular connection. TCS repeats the unit calling sequence, calling only marked units until all output and input values have converged to tolerances and there are no marked units left in the calling sequence. At this point, the simulation is said to have converged at that time step. TCS increments the time and repeats this sequence at the next time step until the end time is reached. The successive substitution strategy employed is outlined in Figure 1.

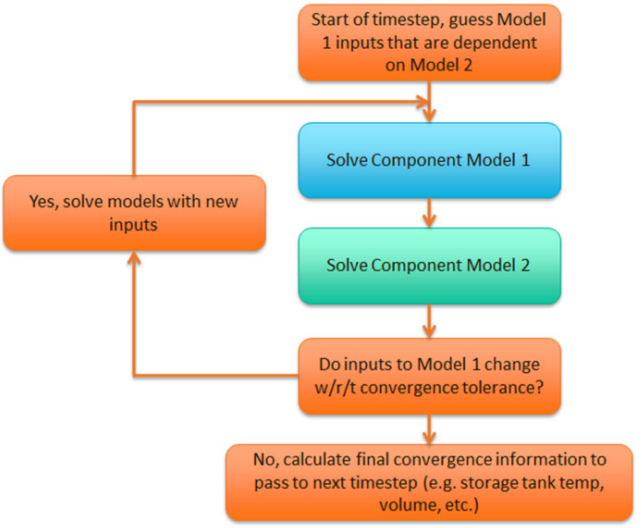

Figure 1. Simulation kernel solution strategy

Key features of the simulator kernel include:

- *Fully dynamic type interface API*: Types can be written in C or C++, and dynamically loaded by the TCS kernel. Because the type API uses the standard C *__cdecl* calling convention, types can be written using any standard C or C++ compiler on any operating system.
- *Multithreading*: The TCS kernel is fully reentrant and thread-safe, allowing it to work well with host software that can dispatch concurrent simulations, provided all the types used in a simulation are also written in this way. The SAM software is being updated to automatically utilize all available processor cores to dispatch multiple parallel simulations. This is not possible with the TRNSYS framework.
- *Data types*: Input and output variables can be numbers, arrays, matrices, or strings. This affords significant flexibility when defining a type subroutine and moving data between units.
- *Tolerance configurability*: TRNSYS requires the specification of a single tolerance value for all variables. TCS enables each connection between an output and an input to be given a unique tolerance value to reach for convergence. Additionally, the tolerance may be specified as a percentage or an absolute value.

These software framework advancements enable high performance simulations that can leverage intra-process parallelism on modern computer architectures.

2.2. Representing systems

Systems are described in TCS by programmatically defining a set of components and their interconnections. TCS includes a utility program for development and debugging models that includes a scripting interface for configuring models, as well as visual and tabular data browsers to view time series outputs calculated by the models. Simulation control parameters (start, step, end times) are also controlled in the development utility, which also provides an online help system that provides variable information for all the inputs and outputs of each component type.

3. Verification

The SAM default system configurations were configured in TCS for the empirical trough model, physical trough model, generic solar thermal system model, molten salt power tower, direct steam power tower, and dish-Stirling models. The six systems were simulated in both the TRNSYS and TCS versions, and scatter plots of the hourly energy yield predictions from both models are shown in Figures 2-7. All three show very good match in the calculated outputs. The physical trough and molten salt power tower models shows a greater spread, which is due primarily to numerical difference between the model implementations when switching operational modes in the controller. The deviations between the two models most frequently occur at startup and shutdown times, when slight changes in the converged outputs of the previous time step may cause the plant controller to enter a different mode of operation in a subsequent time step. Additional investigations are underway to document the exact causes for hour-by-hour differences in models, despite the greater flexibility in TCS to specify convergence tolerances on a variable-by-variable basis.

The total annual energy yield deviation between the two models is characterized by calculating the root mean square error normalized to nameplate capacity of the power plant (Eqn. 1). The RMSE is calculated for the base-case simulations for each model. Table 1 shows that RMSE values are very small, even for the physical trough model which shows slightly more deviation between the two models.

$$RMSE = \frac{\frac{1}{3760}\sqrt{\sum \overline{(_{TCS} - {}_{TRNSYS})^2}}}{100\,MWE} \tag{1}$$

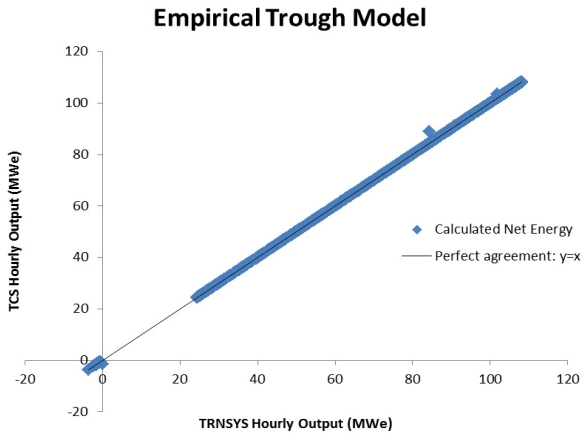

Figure 2. Scatter plot of hourly net energy yield for empirical trough model

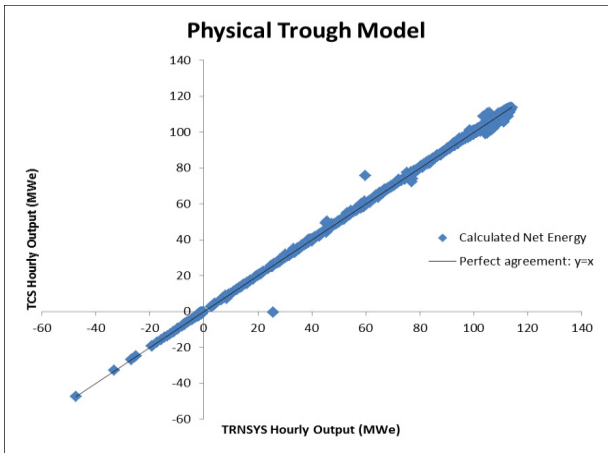

Figure 3. Scatter plot of hourly net energy yield for physical trough model

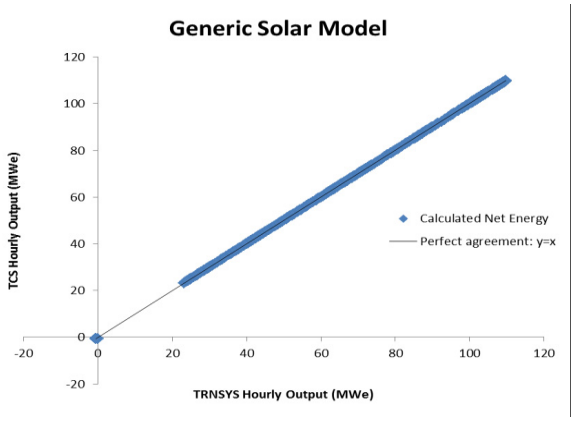

Figure 4. Scatter plot of hourly net energy yield for generic solar thermal system model

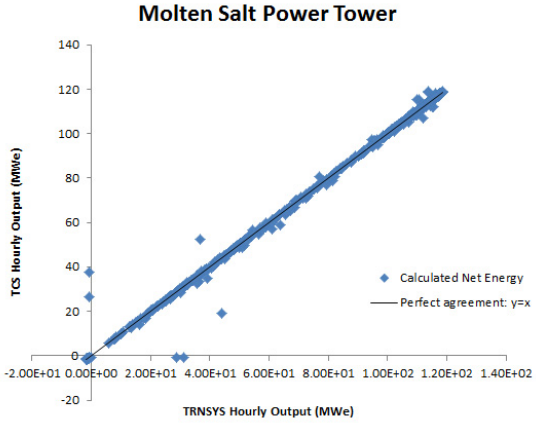

Figure 5. Scatter plot of hourly net energy yield for molten salt power tower model

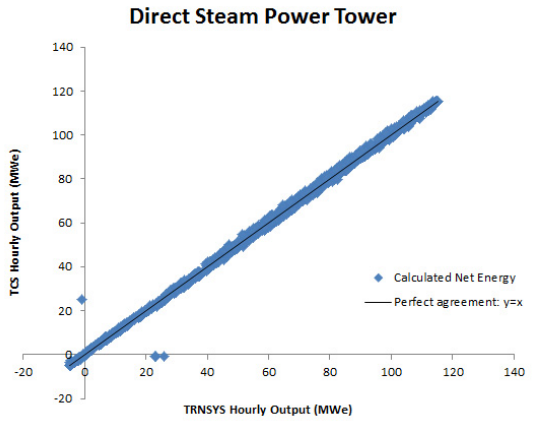

Figure 6. Scatter plot of hourly net energy yield for direct steam power tower model

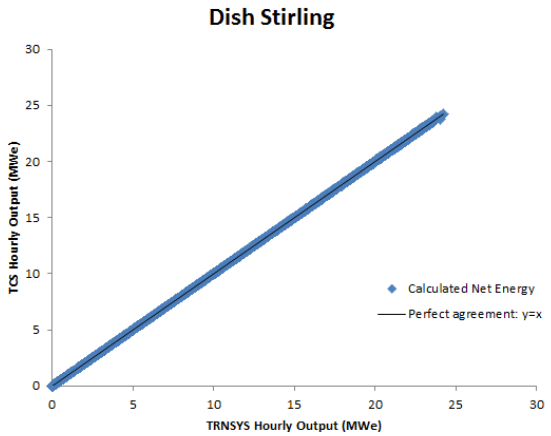

Figure 7. Scatter plot of hourly net energy yield for dish Stirling model

Table 1. CSP system model root mean square deviation between TRNSYS and TCS

System	RMSD (%)
Parabolic trough, empirical model	0.06
Parabolic trough, physical model	0.69
Generic solar thermal system model	0.02
Molten salt power tower	0.78
Direct steam power tower	0.69
Dish Stirling	0.05

4. Simulation Run Time

Simulation speed was tested on a dual-core computer with a 2.5 GHz processor, 8 GB RAM, Windows 7 64-bit, and a solid state hard drive. While TCS is designed to be parallelizable to run multiple hourly simulations simultaneously, for the purposes of this comparison, only a single processor core was utilized. The 32-bit binaries of both TCS and TRNSYS were used.

Table 2. CSP system model simulation times and annual prediction differences, baseline inputs

System	TRNSYS (s)	TCS (s)	Difference, TCS-TRNSYS
Parabolic trough, empirical model	2.1	1.3	-38 %
Parabolic trough, physical model	12.4	9.3	-24 %
Generic solar thermal system model	1.9	0.7	-63 %
Molten salt power tower model	3.0	1.8	-40 %
Direct steam power tower model	23.6	13.8	+71 %
Dish Stirling model	1.7	1.3	-24 %

Nearly all cases show significant speed improvements for TCS compared with the TRNSYS version. The performance improvements for the empirical and generic solar models are due to the fact that the TRNSYS version spends most of the time reading and writing files on disk, while in the TCS system the inputs are set programmatically and outputs at each time step are extracted directly. Thus, a significant amount of overhead unrelated to actual calculation has been removed. Additionally, the models are relatively simple and do not require significant iteration at each time step for convergence; the calculations are essentially "straight through" at each time step from inputs to outputs. However, the physical trough model requires iteration of the solver kernel at each time step, and thus the performance difference relative to the TRNSYS version is not as large, though still significant. The direct steam power tower was not fully converted and validated at the time of writing, and we expect its simulation time will be shorter than TRNSYS when it is finished, judging from the experience with all of the other models.

5. Conclusions

A high performance transient time series solver framework (TCS) for SAM has been implemented in C++ and compared to TRNSYS, the current simulation engine for CSP models in SAM. Equivalent model implementations for six CSP system models show very low root mean square deviations between the two simulation frameworks, and that the TCS version provides a significant reduction in computation time for the same model relative to TRNSYS. Future implementations in SAM will leverage the multi-threading parallelization potential of the TCS kernel, and will consequently greatly reduce simulation times for large parametric scenario analysis, as well as enable the use of CSP simulation engines on a variety of platforms including Linux, Mac OS X, web servers, and others. This new framework will enable distribution of the CSP models within the SAM Software Development Kit (SDK) which currently only includes PV, Wind and Geothermal technologies. In the future, TCS will be used by NREL to continue to augment SAM with new and emerging CSP technologies.

Acknowledgements

This work was supported by the U.S. Department of Energy under Contract No. DE-AC36-08-GO28308 with the National Renewable Energy Laboratory.

References

[1] TRNSYS: Transient System Simulation Tool User's Manual. Version 17. http://www.trnsys.com
[2] EES: Engineering Equation Solver. F-Chart Software. http://www.fchart.com/ees/
[3] Price, H. (2003). Parabolic Trough Solar Power Plant Simulation Model. Proceedings of the ISEC 2003: International Solar Energy Conference, 15-18 March 2003, Kohala Coast, Hawaii. New York: American Society of Mechanical Engineers. 665-673 pp.; NREL Report No. CP-550-34742.
[4] Wagner, M. J.; Gilman, P. (2011). "Technical Manual for the SAM Physical Trough Model." 124 pp.; NREL Report No. TP-5500-51825.
[5] Neises, T.; Wagner, M. "Simulation of Direct Steam Power Tower Concentrated Solar Plant." ASME SE 2012 Conference.
[6] Wagner, M. (M.S. 2008). "Simulation and Predictive Performance Modeling of Utility-Scale Central Receiver System Power Plants." University of Wisconsin-Madison.
[7] Kistler, B. (1986). "A User's Manual for DELSOL3: A Computer Code for Calculating the Optical Performance and Optimal System Design for Solar Thermal Central Receiver Plants." Sandia Report No. SAND86-8018.
[8] Feierabend, L. (M.S., 2009). "Thermal Model Development and Simulation of Cavity-Type Solar Central Receiver Systems." University of Wisconsin-Madison.
[9] Wagner, M.; Zhu, G. (2012). "A Direct-steam Linear Fresnel Performance Model for NREL's System Advisor Model." NREL Conference Paper CP-5500-55044.
[10] Wagner, M. (2012). "Results and Comparison from the SAM Linear Fresnel Technology Performance Model: Preprint. NREL Conference Paper CP-5500-54758."
[11] Wagner, M. J.; Zhu, G. (2011). "Generic CSP Performance Model for NREL's System Advisor Model: Preprint." 10 pp.; NREL Report No. CP-5500-52473.
[12] System Advisor Model (SAM). National Renewable Energy Laboratory, 2013. https://sam.nrel.gov/