Development and Integration of an Advanced Stirling Convertor Linear Alternator Model for a Tool Simulating Convertor Performance and Creating Phasor Diagrams

Jonathan F. Metscher [1] and Edward J. Lewandowski[2]
NASA Glenn Research Center, Cleveland, OH 44135

A simple model of the Advanced Stirling Convertor's (ASC) linear alternator and an AC bus controller has been developed and combined with a previously developed thermodynamic model of the convertor for a more complete simulation and analysis of the system performance. The model was developed using Sage, a 1-D thermodynamic modeling program that now includes electro-magnetic components. The convertor, consisting of a free-piston Stirling engine combined with a linear alternator, has sufficiently sinusoidal steady-state behavior to allow for phasor analysis of the forces and voltages acting in the system. A MATLAB graphical user interface (GUI) has been developed to interface with the Sage software for simplified use of the ASC model, calculation of forces, and automated creation of phasor diagrams. The GUI allows the user to vary convertor parameters while fixing different input or output parameters and observe the effect on the phasor diagrams or system performance. The new ASC model and GUI help create a better understanding of the relationship between the electrical component voltages and mechanical forces. This allows better insight into the overall convertor dynamics and performance.

Nomenclature

C_f	=	alternator motor constant (N·s/m)
F	=	Force (N)
FF	=	sinusoidally varying forcing function (N)
I	=	current (A)
L_{alt}	=	alternator inductance (H)
N	=	number of turns
Q_{in_net}	=	net heat input (W)
t	=	time (s)
R_{alt}	=	alternator resistance (Ω)
$R1, R2$	=	resistances (Ω)
$Sage\_Q_{in}$	=	net heat input as calculated by Sage (W)
V_{emf}	=	electromotive force (EMF) voltage (V)
Wnet	=	Power (W)
x	=	position (m)
ΔV	=	voltage (V)
Φ	=	magnetic flux (Wb)

I. Introduction

THE Advanced Stirling Radioisotope Generator (ASRG) is a radioisotope power system being developed for future NASA deep space missions, where solar power is not feasible. Current radioisotope thermoelectric generators (RTGs) provide reliable electric power for long duration space missions; however they have low conversion efficiency (around 5-7%). The heat source used for the RTGs is the General Purpose Heat Source

[1] Coop Student, Thermal Energy Conversion Branch, 21000 Brookpark Rd.

[2] Deputy Lead, ASRG Power System, Thermal Energy Conversion Branch, 21000 Brookpark Rd., AIAA Senior Member.

(GPHS) which generates heat by the radioactive decay of Plutonium-238, a limited resource that is just beginning to be reproduced in the U.S. again. The Multi-Mission RTG (MMRTG) powering the Mars Science Laboratory Rover Curiosity uses eight GPHS modules. The ASRG is a higher-efficiency system, requiring two GPHS modules. The ASRG achieves this by using highly efficient Stirling engines[1].

The Advanced Stirling Convertor (ASC), developed by Sunpower, Inc., is a free-piston Stirling engine coupled with a linear alternator (Fig. 1). The convertor consists of a helium filled, hermetically sealed, pressure vessel containing a displacer, piston, and the linear alternator. Heat is input to the system from a GPHS to heat the working fluid (helium). The piston is initially put into motion by the alternator as an AC voltage is applied. The working fluid is shuttled between the compression space and expansion space through heat exchangers and regenerator for increased efficiency. The oscillating pressure of the working fluid imparts a force on the displacer. A spring located in the bounce space provides a restoring force for the displacer whereas the bounce space pressure acts as a gas spring providing a restoring force for the piston. Magnets are attached to the piston allowing piston motion to be converted to electric power by the linear alternator. The linear alternator provides a damping force on the piston as well as a spring-like restoring force. To minimize vibrations in the ASRG, the ASCs are mounted opposite of each other and their piston strokes are controlled electrically by adjusting the AC bus voltage and phase of alternator (Fig. 2). The ASRG requires only two GPHS modules (one per ASC), a factor of 4 reduction in Plutonium-238 usage relative to an MMRTG, while providing comparable electric power output[2].

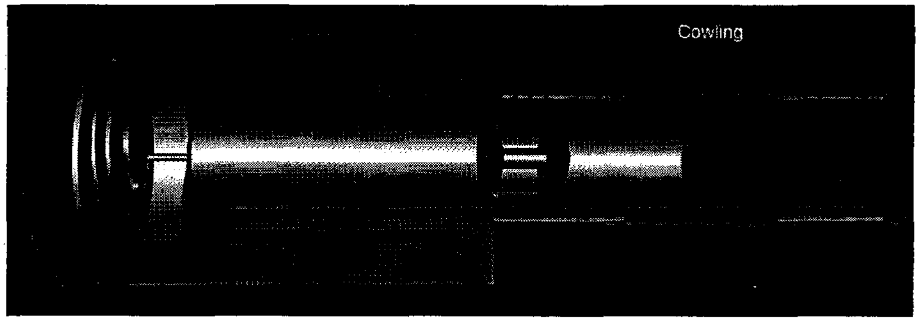


Figure 1. Cutaway of the Advanced Stirling Convertor (ASC).

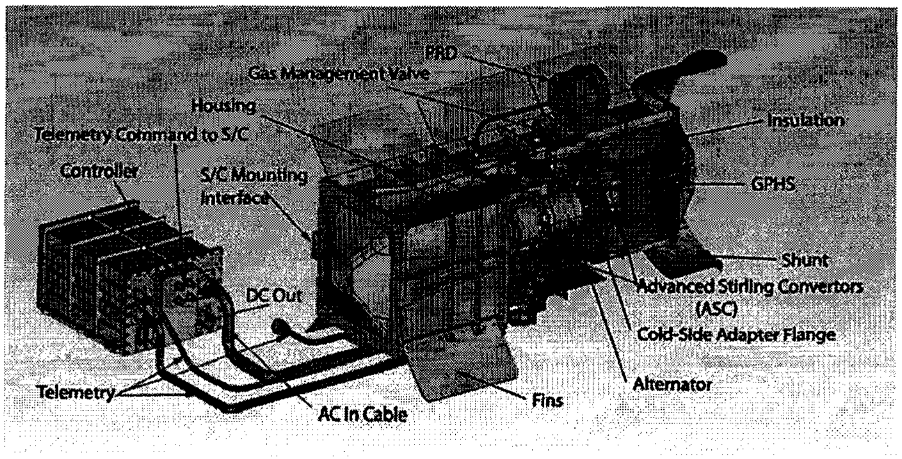


Figure 2. ASRG Diagram

A. Phasor Analysis

To accomplish control of the ASC along with maximizing efficiency requires a better understanding of the interaction of components and relationship of forces acting on those components. Insight into convertor dynamics can be gained by plotting the forces acting on components. A time dependent plot of the forces acting on the piston is shown in Fig. 3.

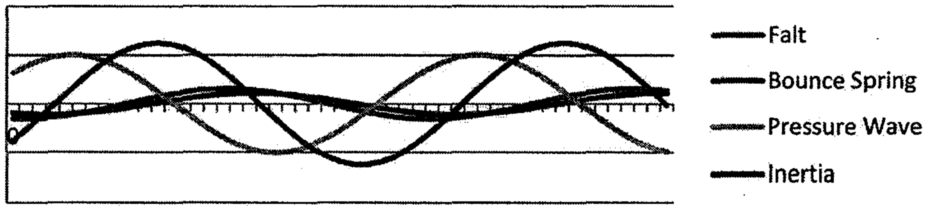


Figure 3. Time dependent plot of piston forces.

The components of a Stirling engine oscillate at the same frequency but vary in phase[3]. The forces are also sufficiently sinusoidal to be represented as phasors. This is a more useful method for plotting the time varying forces as it more clearly shows how forces change in response to a change in system parameters[4]. It is also a more intuitive method as all the force phasors added together should be equivalent to the inertia phasor, verifying $F = ma$. Figure 4 is a typical phasor diagram of forces acting on the piston.

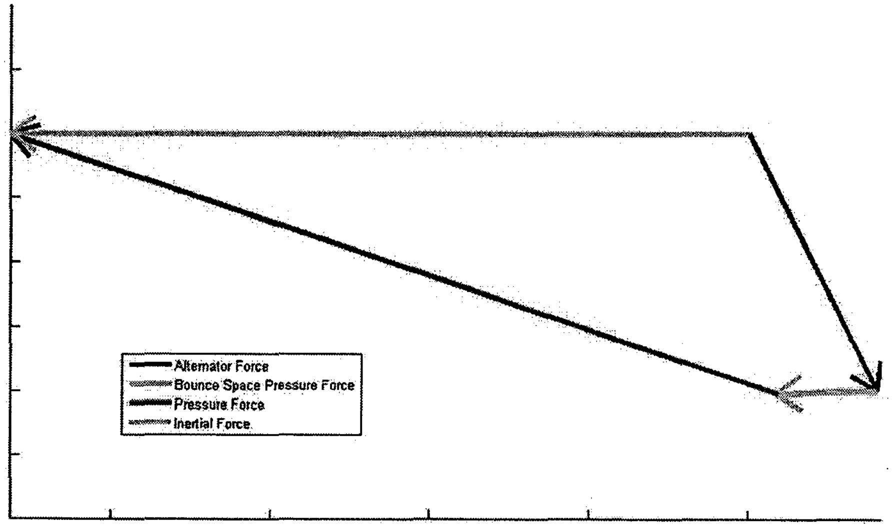


Figure 4. Phasor diagram of piston forces.

B. Sage Software

The Sage software package, developed by Gedeon Associates, is a 1-D thermodynamic modeling software for Stirling machines. It contains a library of generic model components which can be interconnected through the graphical user interface to create a model. The user can set both model geometry as well as initial model parameters. Sage can then use an iterative solver to find a converging solution to the system that balances energy flows and temperatures at interfaces, provided that the model is physically sound and input values are reasonable starting values. Sage also supports user-defined variables, and typically independent Sage model parameters can also be recast as dependent variables and defined through algebraic expressions. Design optimization is also supported by defining constraints and setting optimized variables, making Sage a powerful software tool for the development and analysis of Stirling engines[5].

To build a model, Sage model components are selected and placed into the edit space. Many components allow for sub-components to be placed within the main (or parent) components. Some component interfaces are automatically generated while others may be added individually. These interfaces are the physical inputs and outputs of components such as force, pressure, heat flow, mass flow, and so on. The interfaces are connected from one component to another as can be seen in Fig. 5 below. The Sage model does not give a physical sense of the geometry of the system, however it does give a sense of the physical interactions between components.

It should also be noted that there are two general types of Sage components: time-ring and phasor. Phasor components assume purely sinusoidal motion without harmonics and input/output is given by a magnitude and an angle. "Time-ring" components do not assume purely sinusoidal motion and are solved along a time-grid. Input/output for time-ring components are given as a Fourier series. Time-ring and phasor components cannot be connected unless a motion filter is applied. The motion filter forces the time-ring components solution grid to have sinusoidal characteristics. This can cause some loss of physical properties if the system has significant non-linear properties. The ASC is modeled with phasor components as the steady-state motion is very nearly sinusoidal (Fig. 5).

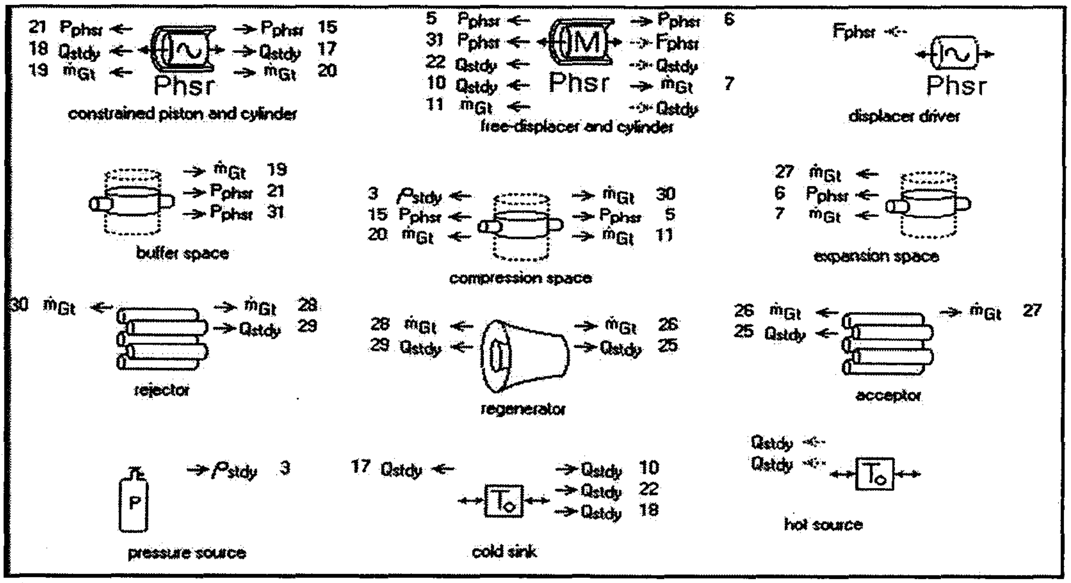


Figure 5. Sage Stirling engine model.

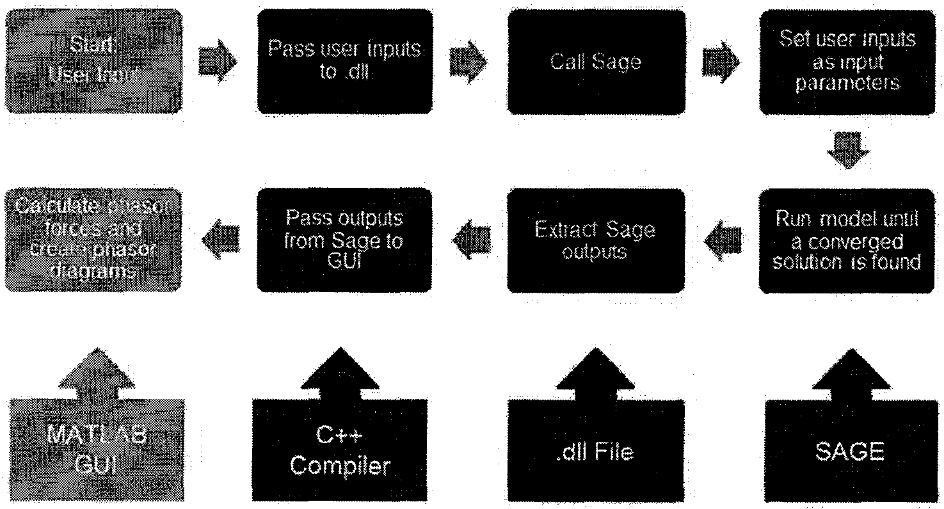

Start:
User Input
Pass user inputs to .dll
Call Sage
Set user inputs as input parameters
Run model until a converged solution is found
Extract Sage outputs
Pass outputs from Sage to GUI
Calculate phasor forces and create phasor diagrams
MATLAB GUI
C++ Compiler
.dll File
SAGE

Figure 6. Diagram of interaction between MATLAB and Sage.

C. MATLAB Graphical User Interface

Sage is also available as a dynamic link library (dll) file which allows access to Sage models from another program running in the Windows operating system using a C++ compiler. MATLAB is such a program that can use a C++ compiler to run the Sage dll. Model parameters can be accessed and changed from the dll file, and outputs can be calculated and exported. When the simulation is executed, MATLAB passes the input parameters to the corresponding model input in the dll file. The dll file calls the Sage solver to update the input parameters and run the model to find a solution. The outputs are then passed back to the dll file and are available to the MATLAB GUI. Figure 6 shows the interaction between MATLAB and Sage.

Previously, a MATLAB GUI (Fig. 7) was developed to interface with the Sage dll to vary input parameters and plot calculated outputs in phasor diagram form[6]. The user selects the Sage model from a library of models and defines the input parameters from the MATLAB GUI. The outputs from the Sage model are used to determine the forces acting on the components and plot phasor diagrams. Plots can be held from run to run and overlaid to show how forces vary as input parameters are changed. The displacer and piston phasor diagrams are drawn with the piston phase along the x-axis.

The Sage model of the ASC and corresponding MATLAB phasor output shown in Fig. 7 does not include the linear alternator dynamics. This model also requires piston amplitude to be set, whereas a more realistic model would allow piston amplitude to change as other operating parameters were varied. Another limitation concerns the heat input parameter which is input as the hot-end temperature. To model an ASC powered by a GPHS, the user should be able to be set the heat input in watts from the heat source and have the tool calculate the temperature at the hot end.

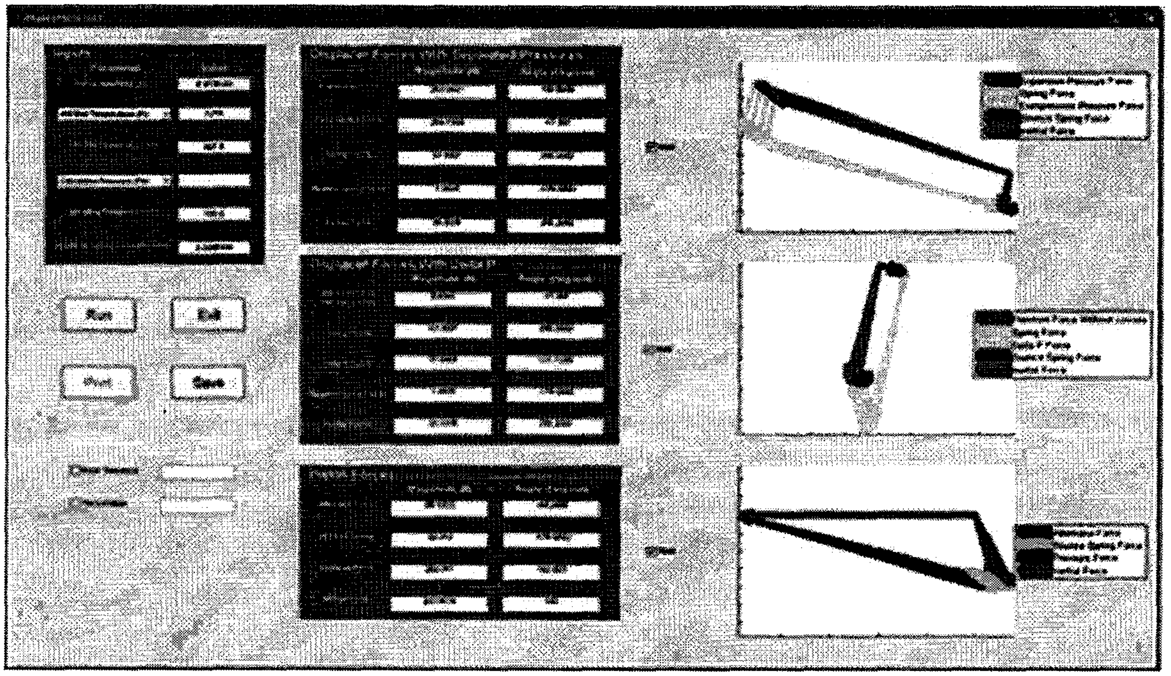


Figure 7. MATLAB GUI running Sage dll and creating phasor diagrams.

II. Linear Alternator Model Development

Alternators could not be modeled with older versions of Sage software. The newest version of the Sage software (ver. 9.1) now includes electromagnetic model components. The electromagnetic model library includes basic electronic circuit components such as voltage and current sources, resistors, capacitors, and inductors. Figure 8 shows a basic RLC circuit model in Sage. It also includes components such as wire coil, permanent magnets, ferromagnetic material, magnetic flux sources, and magnetic field sources. These components can be used to develop simple circuit models or combined with mechanical components to create more complex parts such as alternators or linear motors. The components are connected in the same manner as the mechanical components. The components have only single input/output current interfaces, but may be connected to connection blocks or voltage references which can have multiple (user-defined) interface connections. It should be noted that all electromagnetic components are time-ring rather than phasor components.

Another component of the library is labeled a "transducer." This part largely ignores the physics of the interaction of mechanical force and electric current in an alternator and assumes the relationship shown in Eq. 1,

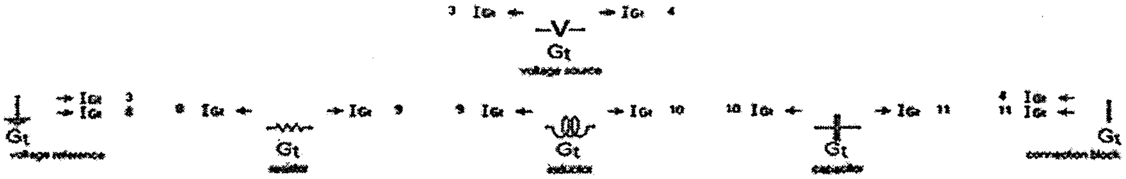


Figure 8. Example RLC circuit model in Sage.

where C_f is the measured motor constant. It also assumes ideal power transfer from the piston to the alternator by Eq. 2. In reality there is loss associated with the alternator. Although not as rigorous as developing an alternator model from basic components, this method does provide a simplified approach of developing an alternator model to test with the existing thermodynamic model of the ASC. This is useful for testing the integration of the two models as well as learning some of the idiosyncrasies of the Sage modeling system.

$$F = C_f * I \tag{1}$$

$$F\,dx/dt = \Delta VI \tag{2}$$

A. Linear Alternator

The linear alternator operates on the principle of Faraday's law. Magnets are attached to a magnet can on the piston of the ASC such that they pass through the alternator coil as the piston reciprocates. The changing magnetic field through the alternator coil induces a voltage V_{emf}. The voltage generated is proportional to rate of change of magnetic flux φ through the coil (therefore proportional to the velocity of the piston) and proportional to the number of turns N in the coil (Eq. 3).

$$Vemf = -N\frac{d\varphi}{dt} \tag{3}$$

The circuit diagram in Fig. 9 is a representation of the linear alternator elements and the AC bus controller. The coil of wire in the alternator has an inductance and resistance represented by L_{alt} and R_{alt} in the diagram. The resistors *R1* and *R2* represent the wire resistance of the circuit and the lead resistance, and are typically small values. The inductance of the alternator creates a voltage phase shift relative to the current which reduces the power factor. A tuning capacitor is used to correct the phase shift caused by the alternator inductance. The AC Power Supply (or AC bus voltage) is used to control the piston amplitude by applying a voltage back to the alternator. The magnitude of the AC bus voltage alters the alternator force on the piston thereby altering the piston amplitude and phase. It is important to model and understand this relationship to be able to achieve control and synchronization of both pistons in an ASRG.

The circuit can be solved simply by Kirchhoff's voltage law to find the necessary AC bus voltage, however it is more useful to model this in Sage to see how circuit elements affect ASC performance. Figure 10 shows the linear alternator circuit in Sage using the transducer model component. A force input to the transducer (from a piston) is translated to a current. The amplitude and frequency of the piston determines the voltage generated.

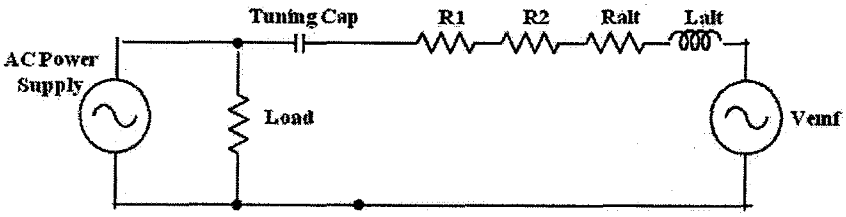


Figure 9. Circuit diagram of the linear alternator and AC bus controller.

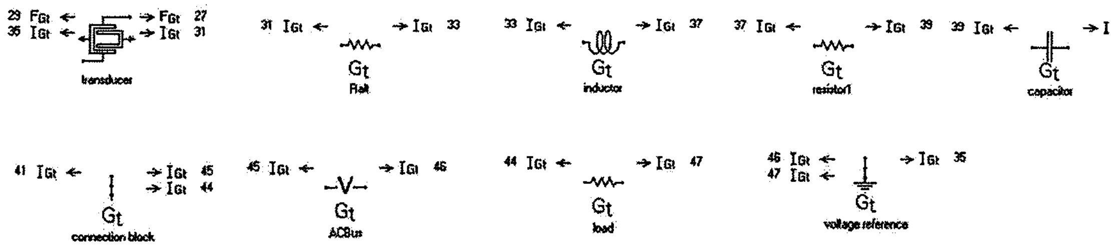


Figure 10. Linear alternator circuit model in Sage.

B. Model Integration

To form a complete model, the alternator needs to be combined with the ASC model, as shown in Fig. 11. The alternator model (Fig. 10) is inserted into the ASC Sage model (Fig. 5) and a force interface is added to the free-piston component in order to connect to the force interface of the transducer. Unfortunately the transducer component does not consider any losses due to eddy-currents or hysteresis. To simulate some loss a damper was added and connected to the piston. While not entirely accurate, it is enough for development purposes and can be calibrated based on test data at a later point.

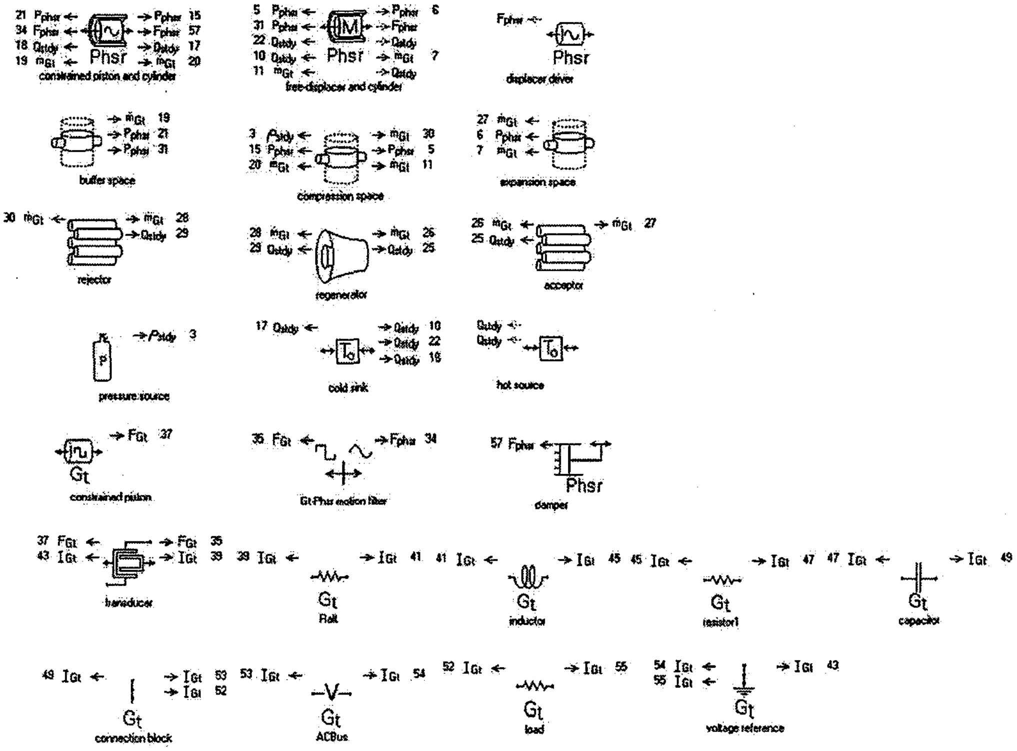
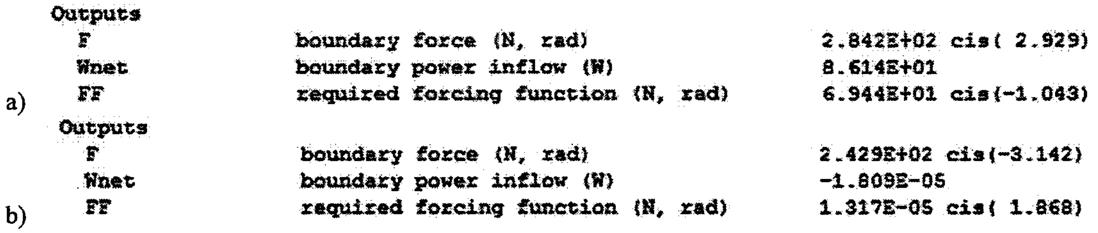


Figure 11. ASC constrained piston and linear alternator model.

The ASC model assumes a constrained piston (piston with set amplitude). This is not realistic but it is useful in determining system characteristics with a set piston amplitude. The piston amplitude is set as an input and system forces are calculated. Any required force necessary to keep the piston in motion at the given amplitude is calculated and output as a required forcing function (Fig. 12a). With the addition of the linear alternator model this can also be used as a sanity check on the model. The AC bus voltage is used to control piston amplitude; therefore the required forcing function should be driven to zero with a reasonable voltage input as shown in Fig. 12b.

To create a more realistic model the constrained piston component was replaced with a free-piston component. This allows the piston amplitude to vary based on system input parameters, including temperature variations of the system and AC bus voltage changes. Piston amplitude is now an output of the Sage solver rather than an input.



a)

```
Outputs
  F          boundary force (N, rad)                    2.842E+02 cis( 2.929)
  Wnet       boundary power inflow (W)                  8.614E+01
  FF         required forcing function (N, rad)         6.944E+01 cis(-1.043)
```

b)

```
Outputs
  F          boundary force (N, rad)                    2.429E+02 cis(-3.142)
  Wnet       boundary power inflow (W)                  -1.809E-05
  FF         required forcing function (N, rad)         1.317E-05 cis( 1.868)
```

Figure 12. a) Model output with a required forcing function (69.44 N). b) Model output with the required forcing function driven to near zero (1.809E-05 N).

C. Alternator Voltage Phasor Diagram

Using the new model of ASC and the Sage dll, the MATLAB GUI is able to run the model and collect the outputs to create a voltage phasor diagram of the linear alternator. Voltage amplitudes and phases are taken in by the GUI and vectors created in a head-to-tail method. The phase of the voltage through a resistor is the same as the phase of the current, hence the voltage R_{alt} is in phase with current. The phase of the current is considered to be along the x-axis in the voltage phasor diagram.

Figure 13 shows a typical voltage phasor diagram for the linear alternator. The voltage across the alternator resistance is in phase with the current, however the phasor is plotted in the opposite direction to indicate a voltage drop. The alternator inductance and tuning capacitance phasors are -90 and 90 degrees respectively as a result of this sign convention. The alternator voltage phasor is the voltage at the terminal leads and is the sum of the V_{emf}, alternator resistance, and alternator inductance phasors. This is useful to plot because the terminal lead voltage can be measured in testing while the V_{emf},, resistance, and inductance can only be calculated.

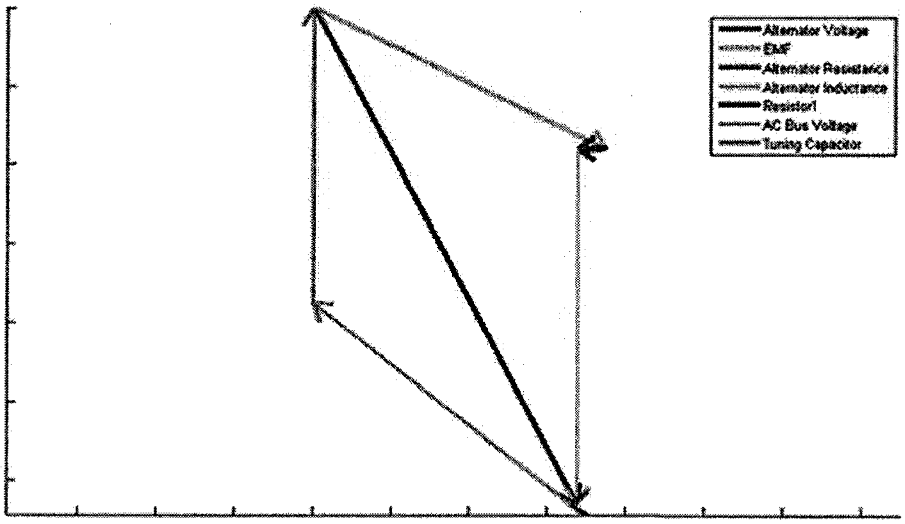


Figure 13. Voltage Phasor Diagram of the Linear Alternator

III. MATLAB GUI and Model Improvements

A new MATLAB GUI, shown in Fig.14, was developed to interface with the new Sage ASC model. The GUI expands the input panel to include linear alternator parameters. A main feature of the interface is the ability to set either AC bus voltage and solve for the resulting piston amplitude or vice versa. This capability is enabled by the newly integrated Sage model. Two phasor diagrams have been added to the original three (Piston Forces, Displacer Forces with Separated Pressures, and Displacer Forces with Delta Pressure). The phasor diagram of alternator voltages has been added along with corresponding output data. A phasor diagram depicting the phase of forces relative to the phase of the piston motion has also been added.

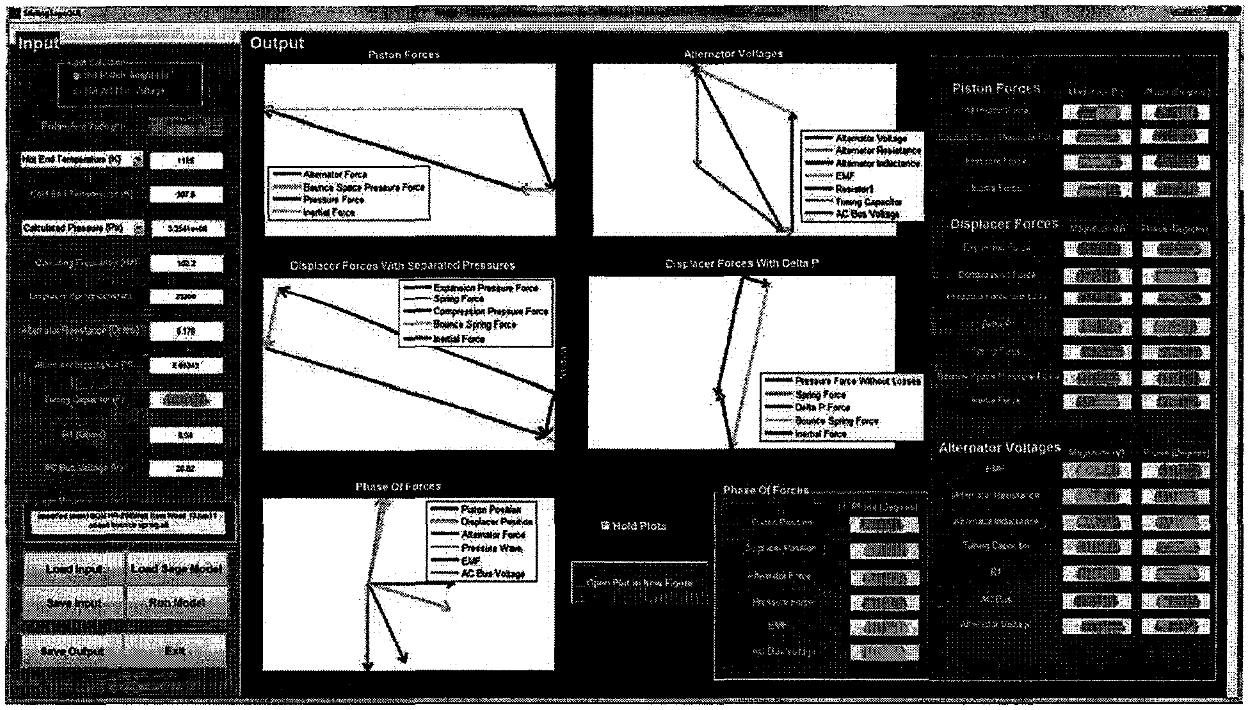


Figure 14. MATLAB GUI running Sage model and plotting phasor diagrams.

A. Improvements

Other improvements have been made to the GUI such as the ability to save input and output data. The output data includes the force data in the output window as well as the Sage output data listing for the model. Previous input data can then be loaded to rerun a simulation. Phasor diagrams can be opened into separate figure windows for easier viewing and manipulation.

The heat input parameter has been improved to allow the user to specify the heat input of the heat source in watts instead of specifying the hot-end temperature. This can be useful to simulate the heat input of the GPHS. Sage however requires hot-end temperature in Kelvin as an input parameter. The net heat input to the convertor is not equal to the gross heat output from the GPHS. Some heat is lost to the insulation, leaving a net heat input (Q_{in_net}). An iterative process is used to determine Q_{in_net} and the corresponding hot-end temperature. The hot-end temperature is estimated and input to Sage, which calculates as an output the heat input $Sage\_Q_{in}$. This is compared to the calculated Q_{in_net} which takes into account heat loss through insulation. The estimated hot-end temperature is adjusted based on the difference between $Sage\_Q_{in}$ and Q_{in_net} and the cycle is repeated.

B. Future Work

The new Sage model and MATLAB GUI offer increased capability over previous models, however there is still room for improvement. The linear alternator model does not accurately calculate alternator losses. The method currently used can change as input parameters are altered, but it is unknown if these changes accurately represent reality. The current model also is a very simple representation of the alternator. A better physical model could be developed in Sage utilizing the magnetic and coil components to build a more physical representation. This type of model in Sage may also be able to better model alternator losses. Another improvement is to combine two ASC models in Sage to develop an ASRG model, allowing a more complete simulation of the entire generator. Finally, the models need to be validated with test data to quantify their accuracy.

IV. Conclusion

The addition of the linear alternator and AC bus controller to the ASC model gives a more realistic representation of the system. It allows the piston amplitude to be determined based on the system input parameters while offering insight into how system performance is affected by changes to the AC bus voltage, which was not possible in the previous model. The reworked GUI complements the new model by simplifying the access to the Sage model parameters and providing quick insight into model performance through phasor diagrams. This tool can be easily adapted and expanded to accomodate future model improvements and GUI enhancements.

Acknowledgments

This work was funded by the NASA Office of Education through the Undergraduate Student Research Program (USRP). Any opinions, findings, conclusions, or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of NASA.

References

[1] Richardson, R. and Chan, J., "Advanced Stirling Radioisotope Generator," Proceedings of NASA Science Technology Conference (NSTC 2007)

[2] Wood, J.G., Wilson, K., Buffalino, A., et al., "Continued Development of the Advanced Stirling Convertor (ASC)," *Proceedings of the 5th International Energy Conversion Engineering Conference (IECEC 2007)* American Institute for Aeronautics and Astronautics, 2007

[3] Walker, G and Senft, J.R. *Free Piston Stirling Engines*, Springer-Verlag, 1985.

[4] Shaler,K and Lewandowski, E.J., "Stirling Convertor Dynamic Analysis Using Phasor Diagrams", *Proceedings of Nuclear and Emerging Technologies for Space (NETS 2011)*

[5] Gedeon, David, "Sage: Object-Oriented Software for Cryo-cooler Design," *Cryocoolers 8*, Edited by R.G. Ross, Jr., Plenum Press, New York, pp. 281-292, 1995.

[6] Saha, Dipanjan, and Lewandowski, E.J., "Development of a Phasor Diagram Creator to Visualize the Piston and Displacer Forces in an Advanced Stirling Convertor," *Proceedings of Nuclear and Emerging Technologies for Space (NETS 2013)*