NASA/CR—2005-213637

Second Law Analysis of Sage and CFD-ACE Models of MIT Gas Spring and "Two-Space" Test Rigs

Asuquo B. Ebiana, Rupesh Savadekar, and Aparna Vallury
Cleveland State University, Cleveland, Ohio

September 2005

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at 301–621–0134

- Telephone the NASA Access Help Desk at 301–621–0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076

NASA/CR—2005-213637

Second Law Analysis of Sage and CFD-ACE Models of MIT Gas Spring and "Two-Space" Test Rigs

Asuquo B. Ebiana, Rupesh Savadekar, and Aparna Vallury
Cleveland State University, Cleveland, Ohio

Prepared under Grant NAG3–2819

National Aeronautics and
Space Administration

Glenn Research Center

September 2005

Acknowledgments

We are grateful for the sponsorship of this research by NASA Glenn Research Center under Grant NAG3–2819. Our grant monitor was Dr. Roy Tew, Jr. We wish to acknowledge the valuable guidance we received from Dr. Tew and the generous disposition of his time throughout the duration of this study.

This report is a formal draft or working paper, intended to solicit comments and ideas from a technical peer group.

This report contains preliminary findings, subject to revision as analysis proceeds.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100

Available electronically at http://gltrs.grc.nasa.gov

Contents

Second Law Analysis of Sage and CFD-ACE Models of MIT Gas Spring and "Two-Space" Test Rigs

Asuquo B. Ebiana, Rupesh Savadekar, and Aparna Vallury
Cleveland State University
Cleveland, Ohio 44115

1. Project Summary

The 2nd Law analysis is used to characterize the thermodynamic losses in two Massachusetts Institute of Technology (MIT) test rigs – piston-cylinder and piston-cylinder-heat exchanger. This second law analysis effort is the final phase of the numerical study of the heat transfer effects under conditions of oscillating pressure and oscillating helium gas flow in the MIT test rigs using Sage and CFD-ACE+ commercial numerical codes. The two-year research effort directed towards the ultimate goal of improving numerical codes for the prediction of heat transfer and thermodynamic losses in Stirling engines was sponsored by NASA Glenn Research Center under grant NAG3–2819.

The results of the first year effort which included temperature, pressure and surface heat transfer variations, pressure-volume diagrams, energy conservation and hysteresis losses are reported in [6]. In this report we present the results of the second year effort on thermodynamic loss analysis inside two solution domains of interest – the gas spring ("single-space") in the piston-cylinder test rig and the gas spring + heat exchanger ("two-space") in the piston-cylinder-heat exchange test rig. This report also documents the thermodynamic loss post-processors (user subroutines) developed for both solution domains.

Second law analysis has proven to be a very powerful tool in the optimization of complex thermodynamic systems and analysis focusing on entropy generation and its minimization has been playing a dominant role in recent times [2,3]. Entropy generation destroys available work of a system and thus provides a useful gauge to the significance of the various irreversibilities within engineering systems. Thus, in order to improve the thermodynamic performance of an engineering system it is necessary to identify and minimize the features guilty of the entropy generation (or available energy loss) within the system.

We have with some success quantified the impact of entropy generation due to conductive heat transfer, fluid friction and mass transfer on the efficiency of the MIT models. Under the specified conditions (201.7 RPM, 1.008 MPa, $T_{wall} = 294$), the heat exchanger heat transfer has the most severe impact on the "two-space" model efficiency followed by outer and inner cylinder heat transfer. Mass transfer and mid-cylinder heat transfer contribute slightly to the model's inefficiency. The relatively high losses in the heat exchanger support the extensive effort in the Stirling community to enhance the heat transfer characteristics of Stirling machine heat exchangers (e.g., the heater, regenerator and cooler). In compact-heat-exchanger passages for example, improvements can be made in the constructive details of the channels (channel shape and aspect ratio, curvature of the return channels, etc.), the actual temperature differences between the cold and hot fluids, the surface finishing, and the type of materials used.

Entropy-generation calculations are an effective design tool. Efficiency improvements in reciprocating machine components can only be achieved through a clearer, more quantitative understanding of the thermal and fluid flow phenomena involved. This work provides a point of reference for incorporation of loss post-processors into Stirling engine numerical codes. The incorporation of a loss post-processor in Stirling engine numerical codes, it is believed, will facilitate the optimization of Stirling engine performance.

2. Introduction

This report presents the results of the 2nd Law analysis post processing of the thermodynamic losses in two Massachusetts Institute of Technology (MIT) test rigs – piston-cylinder and piston-cylinder-heat exchanger. These are the final results of a two-year research effort sponsored by NASA Glenn Research Center under grant NAG3–2819.

The two solution domains of interest in the MIT test rigs are the gas spring ("single-space") in the piston-cylinder test rig and the gas spring + heat exchanger ("two-space") in the piston-cylinder-heat exchange test rig. 1-D and 2-D computer models of each of the two solution domains and the numerical simulation of the gas flow and heat transfer effects in these domains are obtained using two commercial numerical codes – Sage, a 1-D, multi-variable thermodynamic modeling package and CFD-ACE+ a general multi-purpose solver for 2-D and 3-D problems. CFD-ACE+ results are compared with Sage results and where applicable with results from the literature.

The results of the first year effort which included temperature, pressure and surface heat transfer variations, pressure-volume diagrams, energy conservation and hysteresis losses are reported in [6]. In this report we present the results of the second year effort on thermodynamic loss analysis inside the two solution domains of interest and document the thermodynamic loss post-processors (user subroutines) developed for both solution domains.

Thermodynamic loss analyses of simple systems such as the MIT test rigs are often useful to understand some important features of complex pattern forming processes in more complex systems like the Stirling engine. Stirling engines, like other heat engines, convert heat to useful work. In this conversion process, there are various system irreversibilities (finite heat transfer, fluid friction, fluid mixing, leakages, etc.), due to the non-ideal nature of the device, which prevent one from obtaining the desired high efficiency of the ideal Stirling cycle engine. A successful application of the 2nd Law analysis to characterize the various thermodynamic losses inside the two MIT test rigs and the development of a thermodynamic loss post-processor will facilitate the subsequent incorporation of a loss post-processor in Stirling engine numerical codes and the ultimate goal of Stirling engine performance optimization.

The 2nd Law of thermodynamics has proven to be a very powerful tool in the optimization of complex engineering systems. To understand the irreversibilities in these systems, second law analysis [2,3] focusing on entropy generation and its minimization has been playing a dominant role in recent times. Bejan [1] presented a simplified analytical expression for entropy generation rate in a circular duct with constant heat flux at the wall. Sahin [9,10] introduced the second law analysis of viscous fluid in a circular duct at isothermal and constant heat flux boundary conditions. For non-circular duct, Narusawa [8] gives a theoretical and numerical analysis of second law for flow and heat transfer inside a rectangular duct. For other geometry, the second law analysis as well as entropy generation profiles are available in the references by Drost and Zaworski [5] and Bejan [1].

Unlike the 1st Law of thermodynamics which deals with the quantity of energy, the 2nd Law is concerned with the quality of energy. More specifically, it is concerned with the degradation of energy during a process. The 2nd Law postulates the existence of total entropy S, a mathematically defined thermodynamic function which, unlike total energy E, is a non-conserved property of state and can be created via a generation or production term, $\dot{S}_{gen}$, which provides a useful gauge to the significance of the various irreversibilities within engineering systems. Entropy generation destroys available work of a system. Thus, in order to improve the thermodynamic performance of an engineering system it is necessary to identify and minimize the features guilty of the entropy generation (or available energy loss) within the system.

3. Entropy Generation, $\dot{S}_{gen}$

The entropy balance for a single stream (one-inlet, one-exit) system may be written as [12]:

$$\dot{S}_{in} - \dot{S}_{out} + \dot{S}_{gen} = \Delta\dot{S} \tag{1}$$

where the subscripts 'in/out' represent the transfer of entropy into and out of the system with mass flow or heat and 'gen' refers to the generation or production of entropy. The term on the right-hand side represents the change in entropy. The 2nd Law stipulates that entropy generation must be non-negative in all thermo-physical processes.

Introducing the mass flow and heat transfer terms in Eq.(1) we get:

$$\left(\frac{\dot{Q}}{T} + \dot{m}s\right)_{in} - \left(\frac{\dot{Q}}{T} + \dot{m}s\right)_{out} + \dot{S}_{gen} = \Delta\dot{S} \tag{2}$$

The general entropy balance relation (eq. (1)) can be expressed for control volumes as [12]:

$$\sum \frac{\dot{Q}_k}{T_k} + \sum (\dot{m}s)_{in} - \sum (\dot{m}s)_{out} + \dot{S}_{gen} = \Delta\dot{S} \tag{3}$$

where $\dot{Q}_k$ is the heat transfer in or out through the system boundary at location k and at surface temperature T_k. Over a full periodic cycle, $\Delta\dot{S} = 0$ and equation (3) can be solved for $\dot{S}_{gen}$ to yield:

$$\dot{S}_{gen} = -\sum \frac{\dot{Q}_k}{T_k} + \sum (\dot{m}s)_{out} - \sum (\dot{m}s)_{in} \tag{4}$$

In integral form, equation (4) can be written as [7]:

$$\dot{S}_{gen} = -\oint\int_{dt\,ds} \frac{n \cdot q}{T} + \left(\oint\int_{dt\,ds} \dot{m}s\right)_{cs} \tag{5}$$

where $n \cdot q$ and T are the heat transfer rate normal to the surface and the surface absolute temperature, and $\dot{m}$ and s are the mass flow rate and mass-specific entropy. Note that the integral terms are the net entropy transfer with heat and mass out of the system into the surrounding universe. Entropy generation defined in this way refers to the increase in entropy in the universe as a result of internal irreversibilities. Thus, entropy generation may be measured in one of two ways: by entropy flow across the external boundaries of a system (external entropy generation) or by entropy generated by internal processes (internal entropy generation). Equation (5) accounts for the former. Internal entropy generation can be accounted for by tallying up the individual entropy generations in all internal processes. In principle, the two methods of accounting for the entropy generation should give the same answer. Discrepancies which usually arise are often attributable to numerical errors (finite-difference truncation errors, round-off errors, interpolation errors, etc.).

For the MIT test rigs considered in this study, possible internal entropy generation mechanisms include conductive heat flow (axial and film) via temperature gradients and viscous dissipation via velocity gradients. That is:

$$\dot{S}_{gen,\,int.} = \dot{S}_{gen,\,cond} + \dot{S}_{gen,\,visc} \tag{6}$$

where,

$$\text{conductive entropy generation [1]: } \dot{S}_{gen,\,cond} = \frac{k}{T_0{}^2}\left\{\left(\frac{\partial T}{\partial x}\right)^2 + \left(\frac{\partial T}{\partial y}\right)^2\right\} \tag{7}$$

$$\text{viscous entropy generation [1]: } \dot{S}_{gen,\,visc} = \frac{2\mu}{T_0}\left\{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + \frac{1}{2}\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)^2\right\} \tag{8}$$

In the above expressions (eqs. 7-8), $\dot{S}_{gen}$ is the volumetric rate of entropy generation, u, v, and w denote the velocity components, T the temperature, $\dot{m}$ the mass flow rate, and T_0, μ, c_p and k denote respectively, the absolute reference temperature, dynamic viscosity, specific heat at constant pressure and thermal conductivity of the fluid.

Loss analysis using entropy-generation rates due to heat and fluid flow is a relatively new technique for assessing component performance. It offers a deep insight into the flow phenomena, allows a more exact calculation of losses than is possible with traditional means involving the application of correlation losses and provides an effective tool for improving performance. Entropy generation maps can be produced, and designers can use them by scanning them to detect critical areas (locations in which entropy generation is higher than its integral average value over the entire flow field). By considering the local values of the thermal, viscous and mixing entropy generation rates, designers can generate a thermodynamically better design by simply trying to avoid these critical areas or re-computing them after a design modification has been introduced, to assess local and global effects of the design change.

3.1. Availability Energy Loss

All internal entropy generations can also be characterized in terms of availability energy loss defined as [12]:

$$\text{Availability Loss} = W_r - W_i \tag{9}$$

where W_r and W_i ($< W_r$) are the net mechanical work available from reversible and irreversible heat engines respectively; both obtained from the same heat input $\dot{Q}_{in}$ and operating between a high temperature reservoir at temperature T_H and a low temperature reservoir at temperature T_L.

The efficiency of a heat engine is defined as [12]

$$\eta_{th} = \frac{\dot{W}_{net,\,out}}{\dot{Q}_{in}} = 1 - \frac{\dot{Q}_{out}}{\dot{Q}_{in}} \tag{10}$$

From this definition, the net heat outflow for the irreversible heat engine $\dot{Q}_{o,i}$ should be greater than $\dot{Q}_{o,r}$, the net heat outflow for the reversible heat engine.

For the reversible heat engine [12]:

$$\eta_{th,rev} = \frac{\dot{W}_r}{\dot{Q}_{in}} = 1 - \frac{T_L}{T_H} \tag{11}$$

where the Kelvin thermodynamic temperature scale $\left(\dot{Q}_{in}/\dot{Q}_{out}\right)_{rev} = T_H/T_L$ has been used [12]. Thus:

$$\dot{W}_r = \dot{Q}_{in}\left(1 - \frac{T_L}{T_H}\right) \tag{12}$$

and

$$\dot{W}_i = \dot{Q}_{in}\left(1 - \frac{\dot{Q}_{o,i}}{\dot{Q}_{in}}\right) \tag{13}$$

Substitution in (9) gives, after simplification:

$$\text{Availability Loss} = -T_L\left(\frac{\dot{Q}_{in}}{T_H} - \frac{\dot{Q}_{o,i}}{T_L}\right) \tag{14}$$

For a closed system, $\dot{m} = 0$ and referring to equation (4):

$$\text{Availability Loss} = T_L\dot{S}_{gen} \tag{15}$$

The availability-loss concept allows us to think about entropy generation in terms of the more concrete notion of lost mechanical work. A loss in availability equates to a decrease of PV power in an engine. For example, in turbo-machines that generate shaft power (turbines) or absorb power (pumps, compressors), the rate of power lost owing to irreversibilities is proportional to a loss in availability and thus to the rate of entropy generation.

4. Post Processing of Results

Equations (5), (7) and (8) clearly show that local entropy production depends functionally on the local values of velocity, temperature, mass flow and mass-specific entropy. Thus entropy generation can be considered a derived quantity that can be computed by post-processing experimental or numerical flow fields.

4.1. External Entropy Generation, $\dot{S}_{gen,\,ext.}$

The "single-space" and "two-space" models each taken as a whole constitutes a closed, isothermal, non-adiabatic, reciprocating system; reducing equation (5), the external entropy generation equation, to

$$\dot{S}_{gen,\,ext.} = -\oint_{dt}\int_{ds}\frac{\mathbf{n}\cdot\mathbf{q}}{T} \tag{16}$$

Whereas the "single-space" model is limited to the use of equation (16) because of its simple closed system construction, the "two-space" model can also be analyzed by considering its separate components

– the heat exchanger space and the cylinder space – as open systems, in which case equation (5) is applicable. For external entropy generation, results of surface heat transfer and temperature are post-processed for the "single-space" model and results of surface heat transfer, temperature, mass flow rate and mass specific entropy are post-processed for the "two-space" model.

4.2. Internal Entropy Generation, $\dot{S}_{gen,\,int.}$

For internal entropy generation, results of temperature and velocity gradients are post-processed for both MIT models using equations (6) to (8).

The temperature and velocity gradient functions can either be obtained directly from subroutine modules in CFD-ACE+ or calculated using finite difference approximations for the non uniform grid spacing in the "single-space" and "two- space" domains. For example, a finite difference approximation with second order truncation error for $(\partial T/\partial y)_{i,j}$ can be formulated as [11]:

$$\left(\frac{\partial T}{\partial y}\right)_{i,j} = \frac{T_{i,\,j+1} + T_{i,\,j}(\alpha^2 - 1) - \alpha^2 T_{i,\,j-1}}{\alpha(\alpha+1)\Delta y_-};\ \alpha = \left(\frac{\Delta y_+}{\Delta y_-}\right);\quad \Delta y_+ = y_{i,\,j+1} - y_{i,\,j};\quad \Delta y_- = y_{i,\,j} - y_{i,\,j-1} \tag{17}$$

Similar formulations hold for the other gradient functions. Application of this and similar finite difference formulas on the computational domain is left to the creative imagination of the analyst. For the simpler gas spring model where the cell arrangement in the domain can easily be identified and mapped out, we have chosen to apply the finite difference formulas on a virtual grid that is superimposed on the real grid such that the grid points of the virtual grid lie in the center of the cells of the real grid as shown in the figure 1 below. The solid black grid is an arbitrary real grid chosen to illustrate the technique. The dashed grid is the superimposed virtual grid. To calculate a gradient function at i,j, say $(\partial T/\partial y)_{i,j}$, we need y-coordinate and temperature values at (i,j-1) and (i,j+1), which are the cell centers of the cells before and after the cell with center at (i,j). The temperature value at each cell center is obtained from CFD-ACE+ by calling the cell index. Temperature and y-coordinate values are substituted into equation (17) to obtain $(\partial T/\partial y)_{i,j}$.

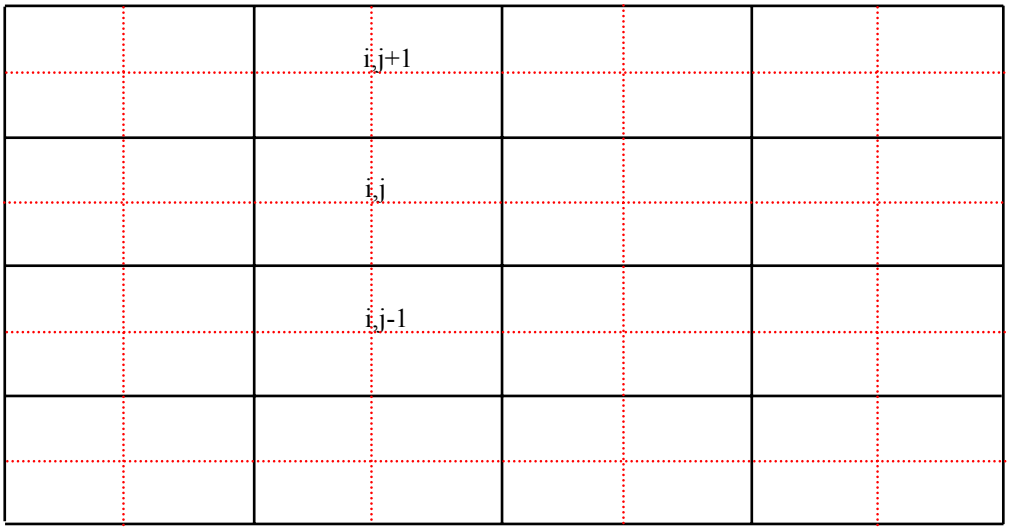


Figure 1.—Virtual Grid Superimposed on Real Grid for Gradient Function Calculation

Similar procedure is used for gradient functions in the x-direction. In the case of a cell adjacent to the domain boundary, one of the parameter values will be taken at the "face" center with index of (i,j-1) or (i,j+1) in the y-direction or (i-1,j) or (i+1,j) in the x-direction. For the more complicated "two-space" domain where the cell arrangements are not easily mapped out, a more sophisticated and systematic technique of identifying three adjacent cells in the domain for application of Eq. (19) as noted above is used.

Calculating gradient functions at cell centers via these techniques is convenient since temperature and velocity values can be obtained readily at cell centers from CFD-ACE+. Also, since CFD-ACE+'s "get gradient function" evaluates the gradient functions at cell centers, it was necessary to apply a technique that will also evaluate gradient functions at cell centers for proper comparison of results.

The use of both methods for calculating the gradient functions yielded internal energy generation results that differ only by about an average of 0.3 percent over 4 cycles of piston motion in the gas spring model and by an average of 5.7 percent over 6 cycles of piston motion in the gas spring + heat exchanger model. The results are shown in table 1. A different approach for calculating the gradient functions provides flexibility to the computational approach.

TABLE 1.—COMPARISON OF INTERNAL ENTROPY GENERATION RESULTS USING TWO DIFFERENT METHODS OF COMPUTING THE GRADIENT FUNCTIONS

Cycle Number	Internal Entropy (W/K)		
	Using CFD-ACE+ Get Gradients Subroutine	Using Finite Difference Approximations	% Difference
Gas Spring Model			
1	0.001533592	0.001538447	0.316077506
2	0.001432515	0.001436870	0.303550066
3	0.001432510	0.001436874	0.304177489
4	0.001432516	0.001436874	0.303758642
	Heat Exchanger Sub-domain of the Gas Spring + Heat Exchanger Model		
1	0.167	0.177	5.907110
2	0.160	0.169	5.673658
3	0.160	0.169	5.657427
4	0.160	0.169	5.656452
5	0.160	0.169	5.656395
6	0.160	0.169	5.656391

4.3. Using CFD-ACE+

External entropy generation can be calculated from equation (16) for closed systems and equation (5) for open systems by post-processing CFD-ACE+ generated data such as integrated surface heat transfer rates. Integrated surface heat transfer rates generated each time step can be stored in output data files called assembly files. An assembly file is enabled by first creating an input file – "filename.fmt" – which specifies the relevant system bounding surfaces in notepad and then choosing the "BC Integral Output" option in CFD-ACE-GUI. Whereas post-processing of assembly file data is only possible for closed systems, user subroutines interfaced with the CFD-ACE+ solver can be used to post-process any CFD-ACE+ generated closed or open system data. The use of user subroutines is especially necessary for open systems.

To calculate the external entropy generation for the gas spring (a closed system), assembly file data can be exported to the excel spreadsheet for calculation of the cyclic time integral value (see equation (16)) and division by the constant surface temperature completes the calculation. The use of user subroutines requires an in-depth knowledge of Fortran concepts. A brief explanation of how user subroutines are used to post-process CFD-ACE+ generated data is provided below.

Internal entropy generation for this study is calculated from equations (6) to (8) using user subroutines to post-process CFD-ACE+ generated data for a closed or open system, for specified absolute reference temperature T_0, dynamic viscosity μ and fluid thermal conductivity k.

4.4. User Subroutines

User subroutines are written in FORTRAN 90 to post-process CFD-ACE+ generated heat transfer, temperature, velocity, mass flow and mass specific entropy data. The interior domain of interest and bounding surfaces are first discretized into cells (interior) and short line segments called "faces" (boundary) as shown in figure 2.

Heat transfer, temperature, velocity, mass flow and mass specific entropy values can be calculated at cell or "face" centers except at the interface between the heat exchanger and cylinder ("two-space" model) where specific entropy and temperature values can only be obtained at cell centers near the interface. The summation of surface and time values, implied by the integrals in equations (5) and (16), is facilitated with the use of do loops in subroutines. In order to minimize program complexity, the subroutines are written to sum up the surface integral values at each time step only and thereafter data is exported to excel spreadsheet for calculation of the cyclic time integrals.

The primary computational domain may also be simplified by division into sub-domains. For example, the cylinder domain in the "two-space" model is divided into outer cylinder, mid cylinder and inner cylinder as illustrated in figure 2. With the heat exchanger considered a sub-domain, the "two-space" model is thus essentially divided into four sub-domains for this study.

The user subroutines (post processors) used to calculate external and internal entropy generations in the "single-space" and "two-space" models are included in the appendix for reference.

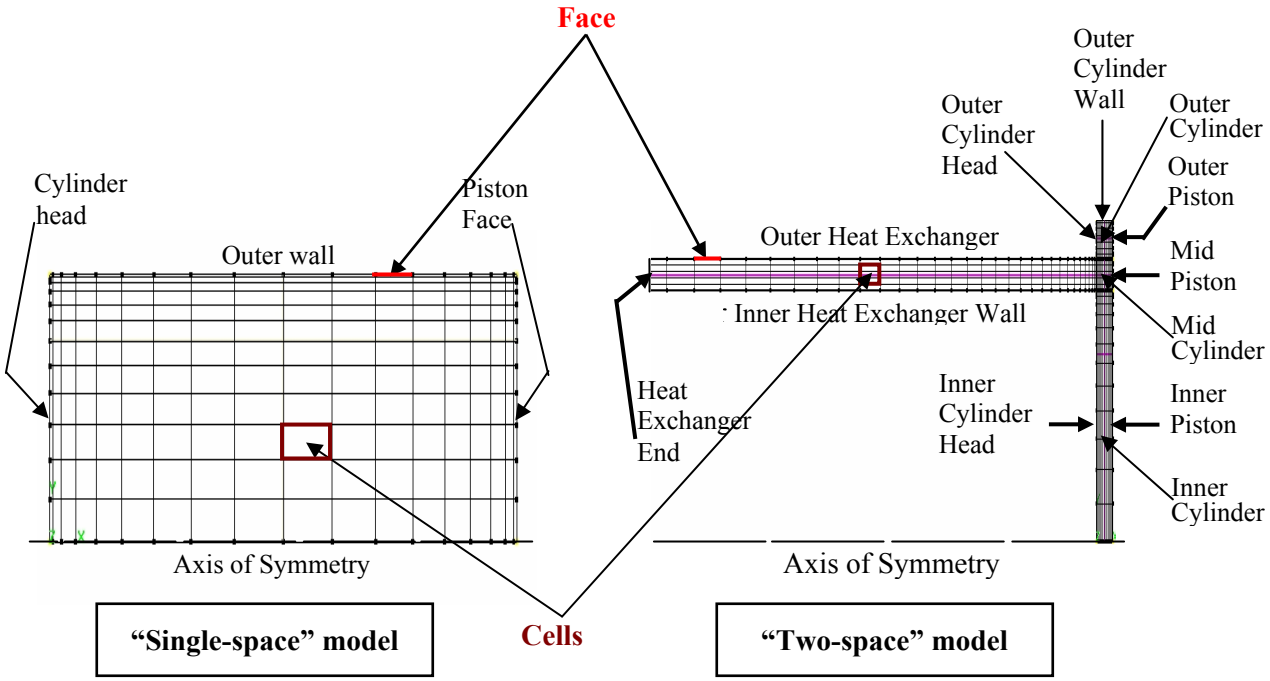


Figure 2.—Discretized computational domain showing "faces" and "cells"

4.5. Using Sage

Sage entropy generation results are calculated from equation (17) using availability loss results obtained from Sage output file viz.:

$$\dot{S}_{gen} = \frac{\text{Availability Loss}}{T_L} = \frac{\text{AEfric} + \text{AEQw} + \text{AEQx} \pm |\text{AEDiscr}|}{T_0} \tag{18}$$

The parameters AEfric, AEQw, and AEQx are available energy losses due to flow friction, surface heat flow and axial heat flow respectively. AEDiscr is the discrepancy between the total available energy loss due to internal entropy generation (AEfric + AEQw + AEQx) and that due to external entropy generation. The "+" is used when it is assumed that AEexternal is greater than AEinternal and the "–" sign is used when the reverse is the case. These assumptions are arbitrary since the relative magnitudes of AEexternal and AEinternal cannot be determined apriori. A discrepancy of zero implies the total available energy loss due to internal entropy generation is equal to that due to external entropy generation.

5. Results and Discussion

The results of the post-processing analysis are shown and discussed below for the indicated models and operating conditions.

5.1. Gas Spring

Figure 3 illustrates the CFD-ACE+ results of the external and internal entropy generations as functions of the number of cycles of piston motion in the gas spring model. A negative external entropy generation which appears to violate the 2nd Law is noted for the first cycle of piston motion. This amounts to an effective source term in the Navier-Stokes equations because of initial transients generated by the moving piston in a flow field that is initially not properly defined. The internal entropy generation appears insensitive to the number of cycles of piston motion one cycle earlier than does the external entropy generation and from the third cycle onward the two methods of accounting for the entropy generation appear to yield essentially the same result as they should in principle. The ~1.3 percent discrepancy between the external and internal entropy generation results is insignificant and probably due to numerical errors. As expected, the external entropy generation profile is the mirror image of the graph of $\oint \delta Q / T$ in the energy conservation plot (fig. 6(a), [6]).

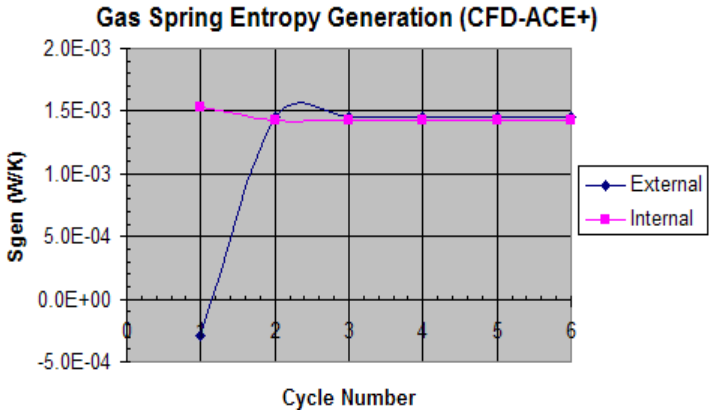


Figure 3.—Entropy generation vs. Cycle number (10 RPM, 192.4 kPa, T_{wall} = 294 K, Grid size = 32 x 20, #tspc = 480).

Table 2 shows Sage and CFD-ACE+ entropy and availability energy loss results for the gas spring. All CFD-ACE+ results, with the exception of the external entropy generation, are less than corresponding Sage results. Sage results show that the primary contribution to the total availability energy loss is the energy loss to surface heat flow (AEQw). With no contributions from flow friction (AEfric) and axial heat flow (AEQx) and negligible numerical errors as reflected by a discrepancy value of 1.031E-06 (W), Sage appears to have completely satisfied the principle that the two methods of accounting for the entropy generation should give the same answer. CFD-ACE+ calculates about 1.3 percent difference between internal and external entropy generation values and therefore does not satisfy the accounting principle as well as Sage.

The percentage differences in the internal and external availability energy loss results between Sage and CFD-ACE+ are on the average about a factor of 3.4 greater than the corresponding entropy generation results. The external results show better agreement between Sage and CFD-ACE+ than corresponding internal results by an average factor of about 1.7. This is not surprising, since internal entropy calculations using 1-D Sage code are expected to incur more errors where the flow features inside the gas spring are more two-dimensional than one-dimensional such as during the expansion stroke at crank angle $\theta = 10°$ and 180° and compression stroke at $\theta = 350°$ and 360° when the flow fields show large recirculation zones [6]. CFD-ACE+ is more suited to handle multi-dimensional flow situations.

TABLE 2.—GAS SPRING ENTROPY GENERATION AND AVAILABILITY ENERGY LOSS (SAGE VS. CFD-ACE+)

Gas Spring			
Sage (T_{wall} = 300 K)		**CFD-ACE+: V2004 (Alpha version)** (10 RPM, 192.4 kPa., T_{wall} = 294 K,) (Grid size = 32x20, #tspc = 480, Opt. Cycle = 4)	
Internal	**External**	**Internal**	**External**
Entropy Generation (W/K)			
$\dot{S}_{gen,int.} = \frac{AEfric + AEQw + AEQx}{T_{wall}}$	$\dot{S}_{gen,ext.} = \frac{AEfric + AEQw + AEQx - \lvert AEDiscr \rvert}{T_{wall}}$	$\dot{S}_{gen,int.} = \dot{S}_{gen,cond} + \dot{S}_{gen,visc}$	$\dot{S}_{gen,ext} = -\oint_{dt}\int_{ds}\frac{n \cdot q}{T}$
0.001444667	0.001444667	0.001432516	0.001451513
Availability Energy Loss (W)			
$AE_{int} = AEfric + AEQw + AEQx$	$AE_{ext} = AEfric + AEQw + AEQx - \lvert AEDiscr \rvert$	$AE_{int} = T_{wall}\dot{S}_{gen,int}$	$AE_{ext} = T_{wall}\dot{S}_{gen,ext}$
0.4334	0.4334	0.421159704	0.426744822
AEfric 0.000E+00	----------	----------	----------
AEQw 4.334E-01	----------	----------	----------
AEQx 0.000E+00	----------	----------	----------
AEDiscr 1.031E-06	----------	----------	----------
|% Difference| between Internal and External Entropy Generation (or Availability Energy Loss)			
0.0		1.3	
	|% Difference| between Sage and V2004 (Alpha version)		
	Internal	**External**	
Entropy Generation	0.8	0.5	
Availability Energy Loss	2.9	1.6	

5.2. Gas Spring + Heat Exchanger

As noted in section 4.4, to simplify the entropy generation analysis, the "two-space" domain was divided into four sub-domains – the heat exchanger, outer cylinder, mid cylinder and inner cylinder – as illustrated in figure 2 above. Figures 4(a) and (b) illustrate the CFD-ACE+ results of the external entropy generation from each of the "two-space" sub-domains. Figure 4(a) shows the functional dependence of entropy generation on the number of cycles of piston motion and figure 4(b) shows a histogram of the entropy generation values characterizing the contributions of each sub-domain to the external entropy generation. Figures 5 (a) and (b) illustrate corresponding CFD-ACE+ results for the internal entropy generation.

As with the gas spring domain, figure 4(a) shows the external entropy generation to be independent of the number of cycles of piston motion beyond the third cycle. Also, the total entropy generation plot is a mirror image of the graph of $\oint \delta Q / T$ in the energy conservation plot (fig. 6(b), [6]). The negative values for the entropy generation in the cylinder sub-domains should be interpreted as the cylinder acting as "entropy sink". That is, these sub-domains appear to extract entropy from the surrounding universe (due to heat entering the cylinder in these sub-domains).

Figure 5(a) shows the internal entropy generation to be independent of the number of cycles of piston motion one cycle earlier than does the external entropy generation, as was the case in the gas spring model. It is noted that the heat exchanger's contribution to entropy generation exceeds the total external entropy generation by ~81 percent (see fig. 4(b)) and is less than the total internal entropy generation by ~14 percent (see fig. 5(b)). A clear inference from these plots is that the major sub-domain contribution to the entropy generation is from the heat exchanger, more so in the case of external entropy generation (fig. 4(b)). (Note: In fig. 4(b) component "external" entropy generations are external to the particular component. Only the total external entropy generation is solely due to "external to the system" entropy generation)

External Entropy Generation from "Two-Space" Sub-domains (CFD-ACE+)

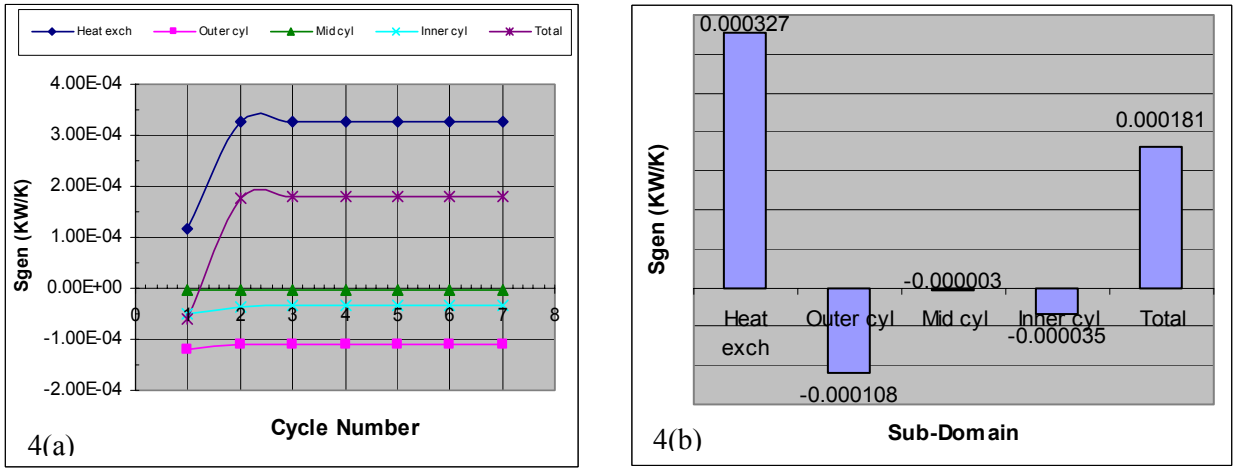


Figure 4.—"Two-Space" External Entropy Generation vs. Cycle Number and Sub-Domain (201.7 RPM, 1.008 MPa, T_{wall} = 294 K; Grid size = 147 x 51, #tspc = 480).

Internal Entropy Generation in "Two-Space" Sub-domains (CFD-ACE+)

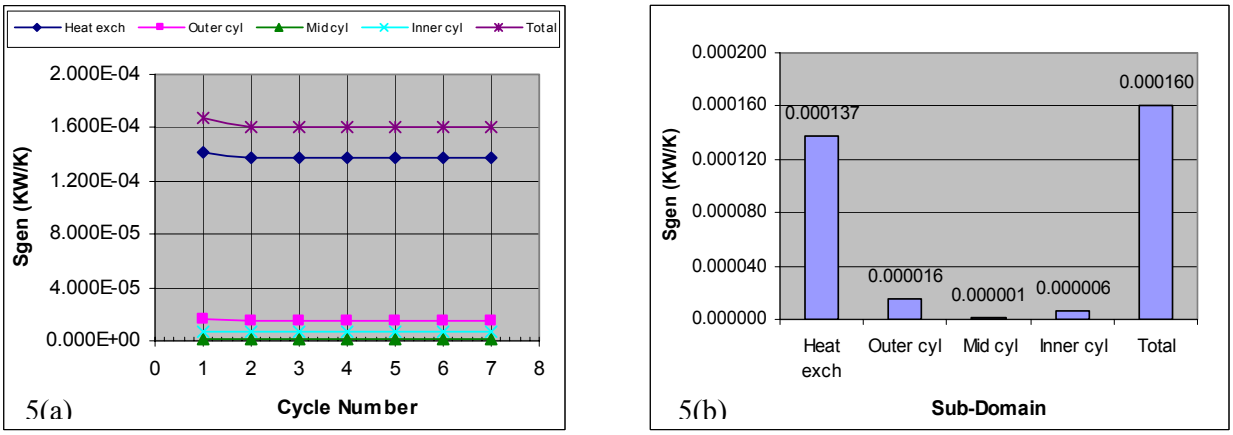


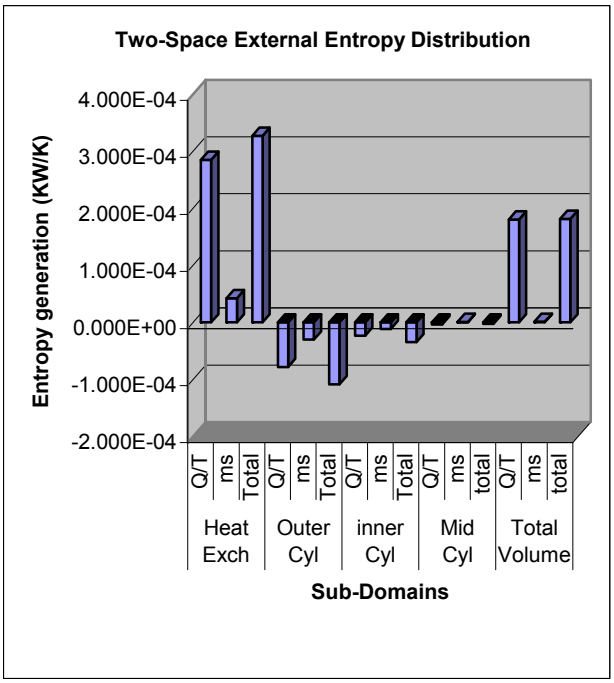
Figure 5.—"Two-Space" Internal Entropy Generation vs. Cycle Number and Sub-Domain (201.7 RPM, 1.008 MPa, T_{wall} = 294 K ; Grid size = 147 x 51, #tspc = 480).

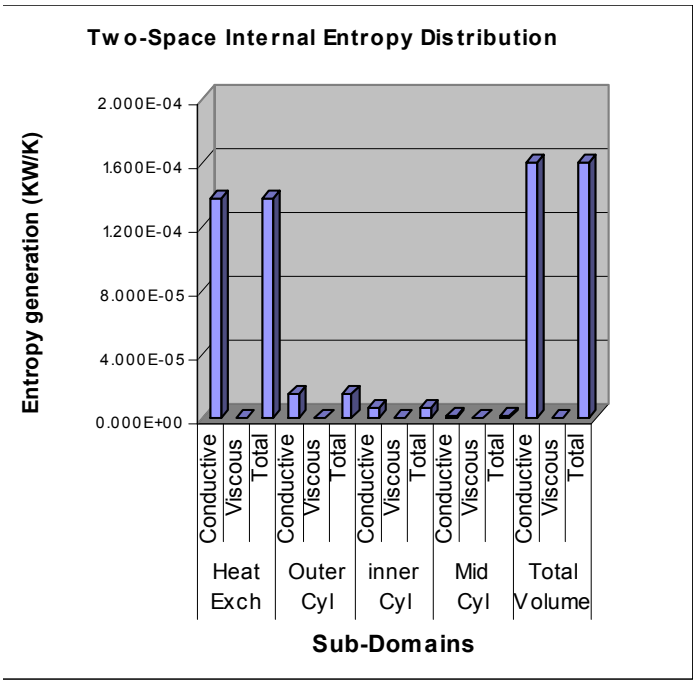
Table 3 illustrates entropy generation and availability loss results obtained for the "two-space" domain using Sage and CFD-ACE+. All CFD-ACE+ results, with the exception of the external entropy generation and availability energy loss results for the cylinder, are greater than corresponding Sage results. Calculation of the percentage differences between internal and external results of entropy and availability energy loss using CFD-ACE+ are based on the assumption that external results are more accurate since the calculation of the integral heat transfer rates are more dependable. With Sage, it does not seem to matter what the basis for calculation of the percentage differences is. Calculation of the percentage differences between Sage and CFD-ACE+ are based on the assumption that CFD-ACE+ results are more accurate since CFD-ACE+ is more suited to handle multi-dimensional flow situations.

Unlike in the gas spring model, there is considerable discrepancy between the internal and external entropy generation results. Sage calculates ~7 percent difference and CFD-ACE+ ~12 percent difference. Thus the two methods of accounting for the entropy generation do not satisfy the accounting principle of equality between internal and external entropy, although Sage does a better job than CFD-ACE+. As in the gas spring model, Sage results show that the primary contribution to the total availability energy loss is the energy loss to surface heat flow (AEQw) with the heat exchanger contributing more than the cylinder to this loss by a factor of about 3.2. The cylinder on the other hand contributes more than the heat exchanger to the energy loss due to the axial heat flow (AEQx) by a factor of about 2.4. There is no contribution from the cylinder to the energy loss to flow friction (AEfric). The discrepancy between internal and external availability energy loss calculated by CFD-ACE+ (0.006188 W) is about 3.6 times that calculated by Sage (0.0017411 W). The percentage differences between Sage and CFD-ACE+ for entropy generation and availability energy loss results are about the same for both the internal and external cases with the exception of the internal results for the cylinder where the percentage difference for entropy generation is a factor of about 2.7 more than that for the availability energy loss.

From table 3, the difference between Sage and CFD-ACE total internal and external entropy generations for the two-space test rig are 41.8 and 51.8 percent, respectively. Whereas, from table 2, the difference between Sage and CFD-ACE internal and external entropy generations for the gas spring 0.8 and 0.5 percent respectively. Thus there is much better agreement between Sage and CFD-ACE in the entropy generation calculations for the gas spring, than for the two-space rig. This may possibly be due to the inability of the Sage 1-D code to accurately account for the changes in flow area between the heat exchanger and the cylinder (Sage can only approximately account for the effect of flow separations, etc. where there are changes in flow area).

Figure 6 shows the distribution of entropy generation inside the two-space model. The impact of the entropy generation due to conductive heat transfer, fluid friction and mass transfer on the efficiency of the MIT models is quantified. Under the specified condition (201.7 RPM, 1.008 MPa, $T_{wall} = 294$), the heat exchanger heat transfer has the most severe impact on the "two- space" model efficiency followed by outer and inner cylinder heat transfer. Mass transfer and mid-cylinder heat transfer contribute slightly to the model's efficiency. The relatively high losses in the heat exchanger support the extensive effort in the Stirling community to enhance the heat transfer characteristic of the regenerator. In compact-heat-exchanger passages for example, improvements can be made in the constructive details of the channels (channel shape and aspect ratio, curvature of the return channels, etc.), the actual temperature differences between the cold and hot fluids, the surface finishing, and the type of materials used.





Figures 6.—"Two-Space" External and Internal Entropy Distribution in the Two-Space Model (201.7 RPM, 1.008 MPa, T_{wall} = 294 K ; Grid size = 147 x 51, #tspc = 480, at Opt. Cycle = 6).

TABLE 3.—GAS SPRING + HEAT EXCHANGER ENTROPY GENERATION AND AVAILABILITY ENERGY LOSS (SAGE VS. CFD-ACE)

Gas Spring + Heat Exchanger								
Sub-Domain	**Sage** (T_{wall} = 300 K)				**CFD-ACE+: V2004 (Alpha version)** (201.7 RPM, 1.008 MPa, T_{wall} = 294 K,) (Grid size = 147x46, #tspc = 480, Opt. Cycle = 6)			
	Internal		**External**		**Internal**		**External**	
	Entropy Generation (KW/K)							
	$\dot{S}_{gen,\,int.} = \frac{AEfric + AEQw + AEQx}{T_{wall}}$		$\dot{S}_{gen,ext.} = \frac{AEfric + AEQw + AEQx - \lvert AEDiscr \rvert}{T_{wall}}$		$\dot{S}_{gen,\,int.} = \dot{S}_{gen,\,cond} + \dot{S}_{gen,\,visc}$		$\dot{S}_{gen,\,ext} = -\oint_{dt}\int_{ds} \frac{n \cdot q}{T}$	
Heat Exchanger	0.00007069		0.00006535		0.000137		0.000327	
Cylinder	0.00002245		0.00002198		0.000023		-0.000146	
Total	0.00009314		0.00008733		0.000160		0.000181	
Outer Cylinder	----------		----------		0.0000155		-0.000108	
Mid Cylinder	----------		----------		0.0000014		-0.000003	
Inner Cylinder	----------		----------		0.0000062		-0.000035	
Availability Energy Loss (KW)								
AE Components	Heat Exchanger	Cylinder	Heat Exchanger	Cylinder	Heat Exchanger	Cylinder	Heat Exchanger	Cylinder
AEfric	0.00025860	0.00000000	----------	----------	----------	----------	----------	----------
AEQw	0.02089000	0.00659400	----------	----------	----------	----------	----------	----------
AEQx	0.00005789	0.00014010	----------	----------	----------	----------	----------	----------
Sub-Domain	$AE_{int.} = AEfric + AEQw + AEQx$		$AE_{ext.} = AEfric + AEQw + AEQx - \lvert AEDiscr \rvert$		$AE_{int} = T_{wall}\dot{S}_{gen,\,int}$		$AE_{ext} = T_{wall}\dot{S}_{gen,\,ext}$	
Heat Exchanger	0.02120649		0.01960549		0.040295		0.096142	
Cylinder	0.00673410		0.00659400		0.006793		-0.042866	
Total	0.02794059		0.02619949		0.047088		0.053276	
|AEDiscr|	0.0017411				0.006188			
Outer Cylinder	----------		----------		0.004565		-0.031859	
Mid Cylinder	----------		----------		0.000412		-0.000809	
Inner Cylinder	----------		----------		0.001816		-0.010198	
Sub-Domain	**|% Difference| between Internal and External Entropy Generation (or Availability Energy Loss)**							
Heat Exchanger	8.0				58.0			
Cylinder	2.0				116.0			
Total	7.0				12.0			
Outer Cylinder	----------				114.0			
Mid Cylinder	----------				149.0			
Inner Cylinder	----------				118.0			
	|% Difference| between Sage and V2004 (Alpha version)							
	Internal				**External**			
	Entropy Generation		**AE Loss**		**Entropy Generation**		**AE Loss**	
Heat Exchanger	48.4		47.4		80.0		79.6	
Cylinder	2.4		0.9		115.1		115.4	
Total	41.8		40.7		51.8		50.8	

6. Concluding Remarks

The comparisons of the results of the 1-D Sage and 2-D CFD-ACE+ 2nd Law analysis of the MIT "single-space" and "two-space" models may be summarized as follows:

(1) By the third cycle of piston motion, external and internal entropy generations in both models are insensitive to the number of cycles of piston motion.
(2) As expected, the external entropy generation profile is the mirror image of the heat transfer profile as a function of cycle number in both models.
(3) Sage results in both models show that the primary contribution to the total availability energy loss is the energy loss to surface heat flow (AEQw).
(4) CFD-ACE+ does not satisfy the entropy generation accounting principle as well as Sage.
(5) Unlike in the gas spring model, there is considerable discrepancy between the internal and external entropy generation results in the gas spring + heat exchanger model.
(6) Sage and CFD-ACE+ predictions of external entropy generation and availability loss and are in much better agreement for the gas spring model than for the cylinder domain of the gas spring + heat exchanger model.
(7) There is much better agreement between Sage and CFD-ACE in the entropy generation calculations for the gas spring, than for the two-space rig. This may possibly be due to the inability of the Sage 1-D code to accurately account for the changes in flow area between the heat exchanger and the cylinder.
(8) Under the specified conditions (201.7 RPM, 1.008 MPa, $T_{wall} = 294$), the heat exchanger heat transfer has the most severe impact on the "two-space" model efficiency.
(9) Entropy-generation calculations are an effective design tool. Efficiency improvements in reciprocating machine components can only be achieved through a clearer, more quantitative understanding of the thermal and fluid flow phenomena involved.

With a modest extra effort, entropy-generation calculations can be performed to obtain significant information that will better the design engineer's understanding of the physical phenomena involved. This work provides a point of reference for incorporation of loss post-processor into Stirling engine numerical codes.

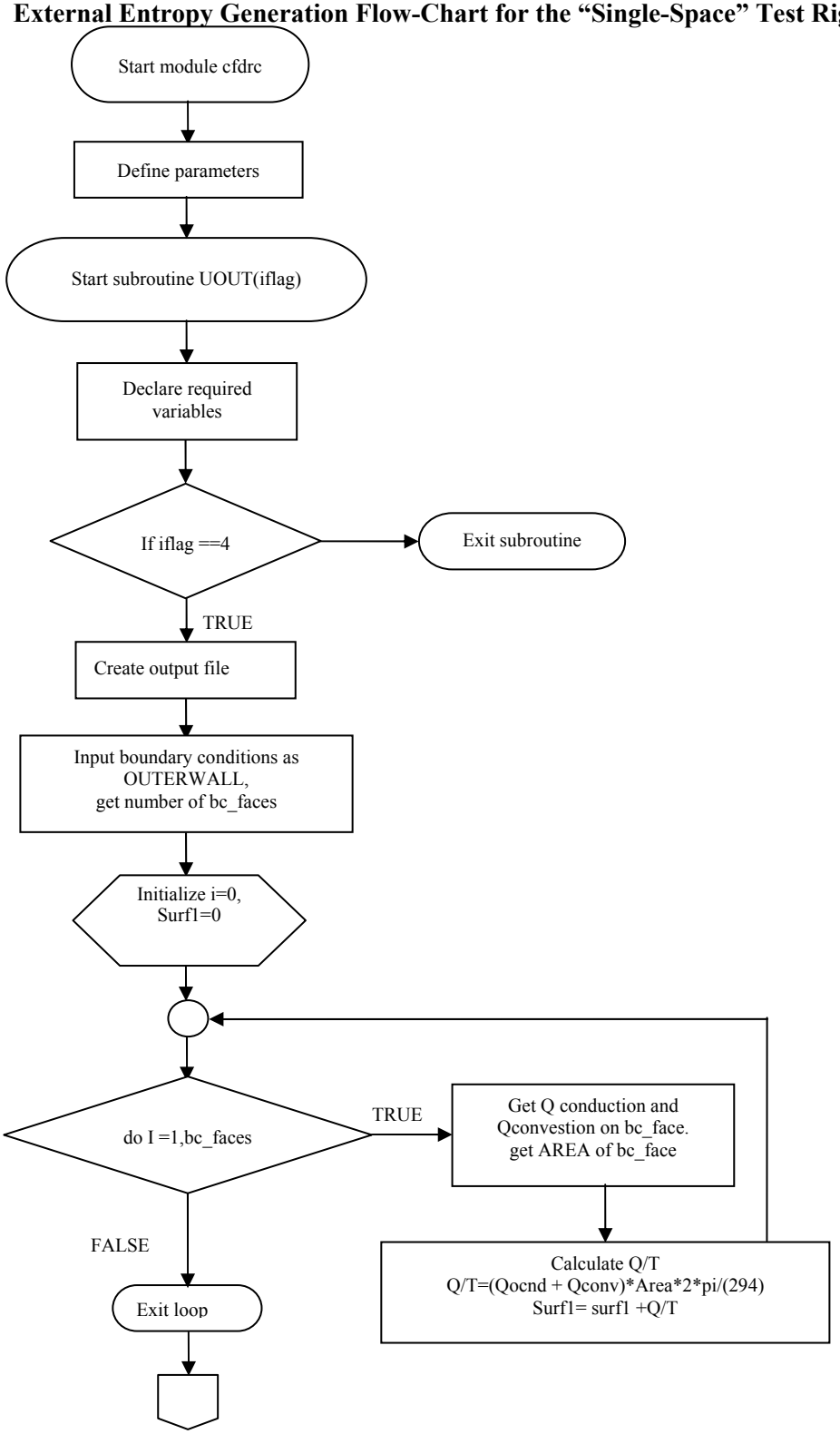
7. References

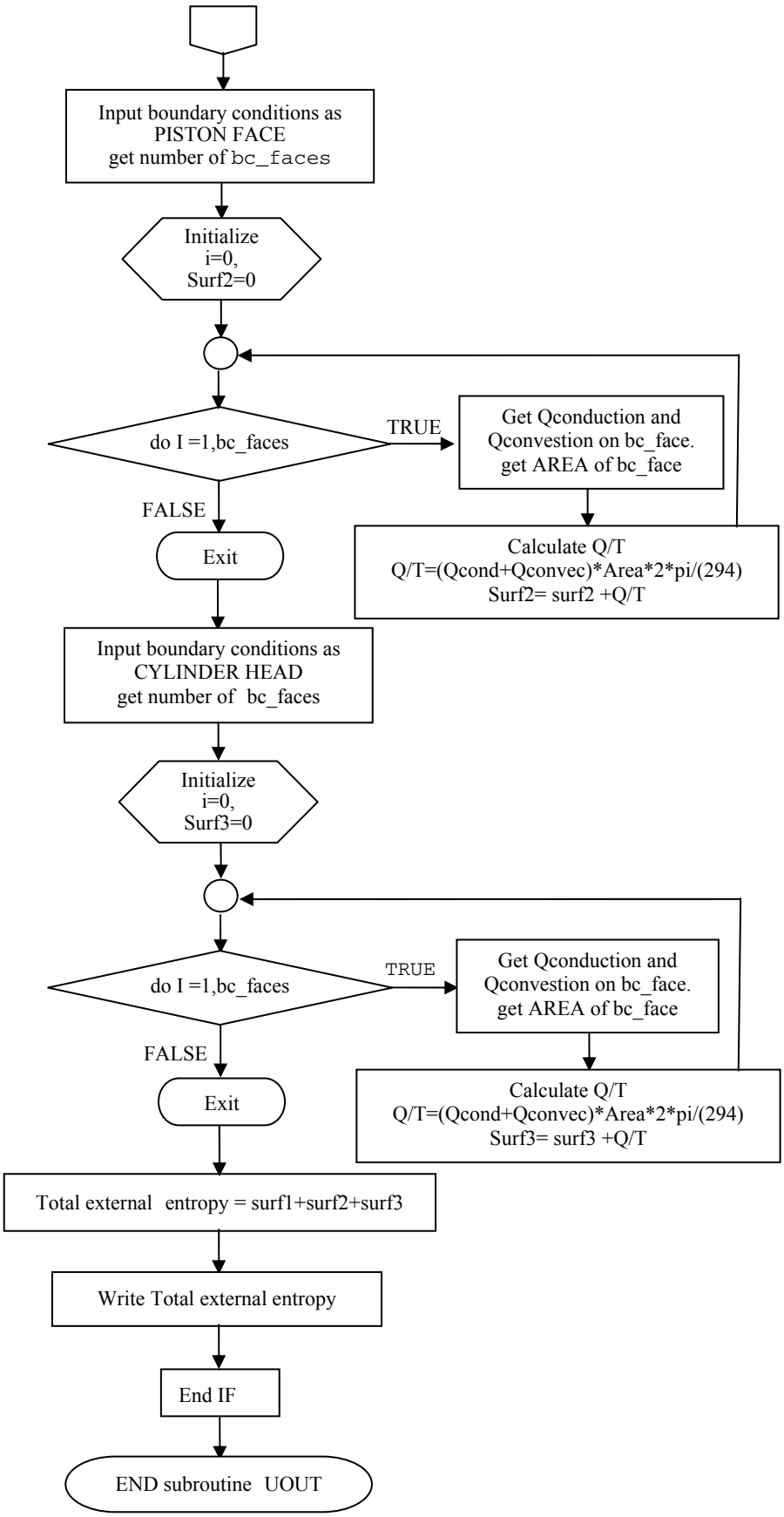
[1] Bejan, A. "A study of entropy generation in fundamental convective heat transfer," J. Heat Transfer 101, 718–725, 1979.
[2] Bejan, A., "Second-law analysis in heat transfer and thermal design," Adv. Heat Transfer 15, 1–58, 1982.
[3] Bejan, A., Entropy Generation Minimization, CRC Press, Boca Raton, NY, 1996
[4] Bejan A., Advanced Engineering Thermodynamics, 2nd. ed., Wiley, New York, 1997.
[5] Drost, M.K., Zaworski, J.R., "A review of second law analysis techniques applicable to basic thermal science research," ASME AES 4, 7-12, 1988.
[6] Ebiana, A.B., Savadekar, R.T, Vallury, A., "2nd Law Analysis of Sage and CFD-ACE+ Models of MIT Gas Spring" and "Two-Space" Test Rigs," First Year Report, NASA Grant NAG3–2811, 2003. Also presented at the 2nd. International Energy Conversion Engineering Conference. August 16–19, 2004, Providence, RI.
[7] Gedeon, D., Sage User's Guide, 3rd.ed., (Gedeon Associates, 16922 South Canaan Road, Athens, OH 45701), 1999.
[8] Narusawa, U., "The second-law analysis of mixed convection in rectangular ducts," Heat Mass Transfer 37, 197–203, 2001.

[9] Sahin, A.Z., "Second law analysis of laminar viscous flow through a duct subjected to constant wall temperature," J. Heat Transfer 120, 76–83, 1998.
[10] Sahin, A.Z., "Effect of variable viscosity on the entropy generation and pumping power in a laminar fluid flow through a duct subjected to constant heat flux," Heat Mass Transfer 35, 499–506, 1999.
[11] Tannehill, J.C., Anderson, D.A. and Pletcher, R.H., Computational Fluid Mechanics and Heat Transfer, Taylor & Francis, 2nd. ed., Hemisphere Publishing Corporation, 1984.
[12] Yunus A. Cengel and Michael A. Boles, Thermodynamics, an Engineering Approach, 4th. ed., The McGraw-Hill Companies, Inc., 2002.

Appendix A—External Entropy Generation

External Entropy Generation Flow-Chart for the "Single-Space" Test Rig.





External Entropy Generation Subroutine for the "Single-Space" Test Rig.

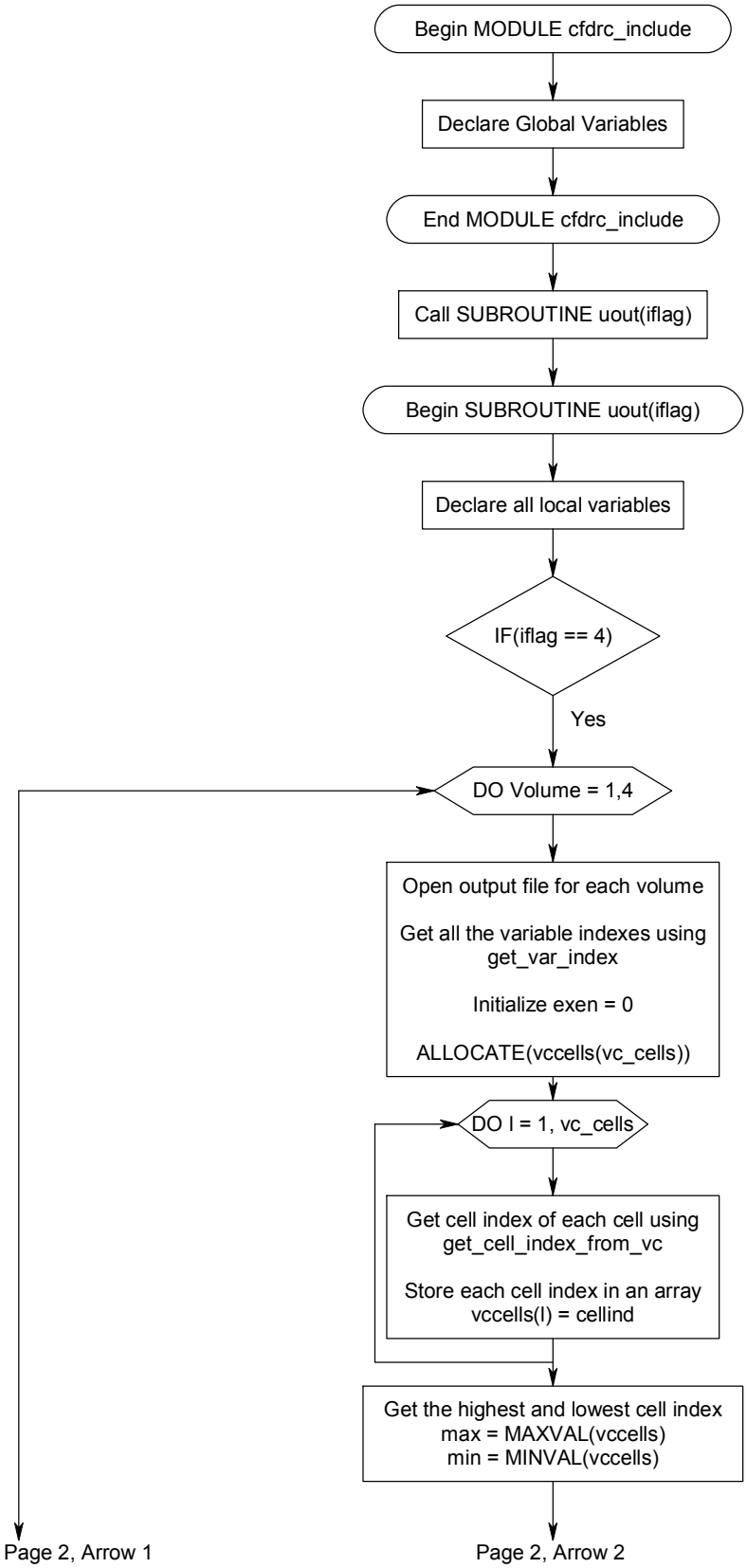
```
MODULE cfdrc_user
IMPLICIT NONE
INTEGER, PARAMETER :: int_p = SELECTED_INT_KIND(8)
INTEGER, PARAMETER :: string_length = 80
INTEGER, PARAMETER :: real_p = SELECTED_REAL_KIND(8)
INTEGER, PARAMETER :: XDIR = 1, YDIR = 2, ZDIR = 3
REAL(real_p) , PARAMETER :: zero = 0.0d0, one = 1.0d0,two = 2.0d0, &
three = 3.d0, four = 4.0d0, pi = 3.1415926535898d0
END MODULE cfdrc_user
SUBROUTINE uout(iflag)
!DEC$ ATTRIBUTES DLLEXPORT :: uout
USE cfdrc_user, ONLY : int_p, real_p, string_length
INCLUDE 'cfdrc_include'
IMPLICIT NONE
! Variable Declarations
INTEGER(int_p), INTENT(IN) :: iflag
INTEGER(int_p) :: bf_index(32)
REAL(real_p) :: time,surf1,surf2,surf3, &
&QbyT(32),Q1(32),Q2(32),A(32),totalQbyT
CHARACTER(len=string_length) :: bc_name, var_name
INTEGER(int_p) :: i, ierror, bc_index, bc_type, bc_faces, &
& var_index, time_step
LOGICAL :: error
! Begin program (for each time step)
IF(iflag == 4) THEN
OPEN(UNIT=10, FILE='extentropy.OUT', STATUS='REPLACE', IOSTAT=ierror)
! Time step
CALL get_time(time, time_step, error)
! OUTER WALL
bc_name='outer wall'
CALL get_bc_index(bc_name, bc_index, error)
CALL get_bc_info(bc_index, bc_type, bc_faces, error)
surf1=0
CALL get_bc_global_faces(bc_index, bc_faces, bf_index, error)
! Calculating heat transfer due to conduction and convection on each boundary face
DO i = 1, bc_faces
var_name='Q_COND'
CALL get_var_index(var_name, var_index, error)
CALL get_value_one_bc_face(bc_index, bf_index(i), var_index, &
& Q1(i), error)
var_name='Q_CONV'
```

```
CALL get_var_index(var_name, var_index, error)
CALL get_value_one_bc_face(bc_index, bf_index(i), var_index, &
& Q2(i), error)
var_name='Area'
CALL get_var_index(var_name, var_index, error)
CALL get_value_one_bc_face(bc_index, bf_index(i), var_index, &
& A(i), error)
QbyT(i)=(Q1(i)+Q2(i))*A(i)/294
surf1=surf1+QbyT(i)
ENDDO
! PISTON FACE
bc_name='piston face'
CALL get_bc_index('piston face', bc_index, error)
CALL get_bc_info(bc_index, bc_type, bc_faces, error)
surf2=0
CALL get_bc_global_faces(bc_index, bc_faces, bf_index, error)
! Calculating heat transfer due to conduction and convection on each boundary face
DO i = 1, bc_faces
var_name='Q_COND'
CALL get_var_index(var_name, var_index, error)
CALL get_value_one_bc_face(bc_index, bf_index(i), var_index, &
& Q1(i), error)
var_name='Q_CONV'
CALL get_var_index(var_name, var_index, error)
CALL get_value_one_bc_face(bc_index, bf_index(i), var_index, &
& Q2(i), error)
var_name='Area'
CALL get_var_index(var_name, var_index, error)
CALL get_value_one_bc_face(bc_index, bf_index(i), var_index, &
& A(i), error)
QbyT(i)=(Q1(i)+Q2(i))*A(i)/294
surf2=surf2+QbyT(i)
ENDDO
! CYLINDER HEAD
bc_name='cylinder head'
CALL get_bc_index('cylinder head', bc_index, error)
CALL get_bc_info(bc_index, bc_type, bc_faces, error)
surf3=0
CALL get_bc_global_faces(bc_index, bc_faces, bf_index, error)
```

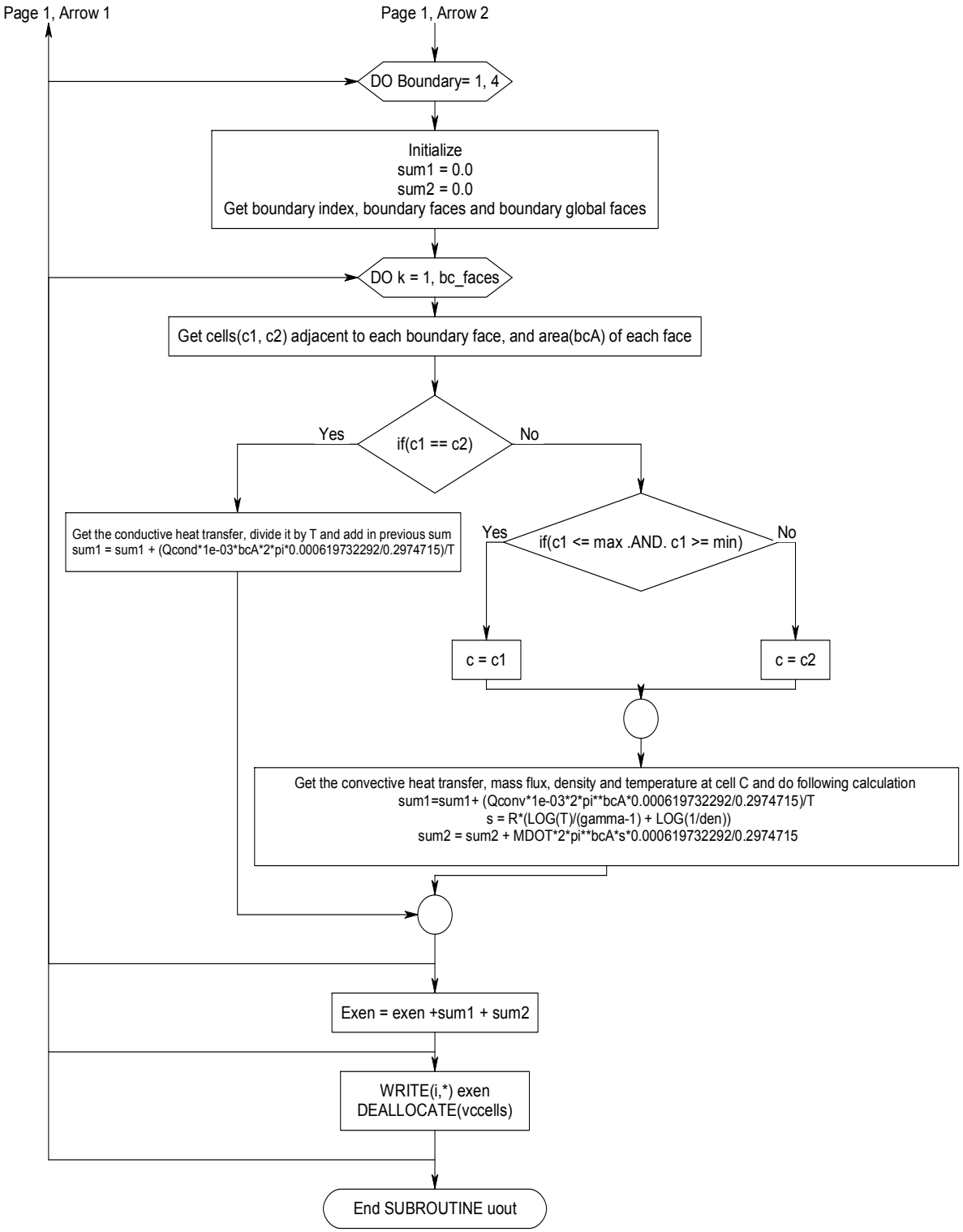
```
! Calculating heat transfer due to conduction and convection on each boundary face
```

```
DO i = 1, bc_faces
var_name='Q_COND'
CALL get_var_index(var_name, var_index, error)
CALL get_value_one_bc_face(bc_index, bf_index(i), var_index, &
& Q1(i), error)
var_name='Q_CONV'
CALL get_var_index(var_name, var_index, error)
CALL get_value_one_bc_face(bc_index, bf_index(i), var_index, &
& Q2(i), error)
var_name='Area'
CALL get_var_index(var_name, var_index, error)
CALL get_value_one_bc_face(bc_index, bf_index(i), var_index, &
& A(i), error)
QbyT(i)=(Q1(i)+Q2(i))*A(i)/294
surf3=surf3+QbyT(i)
ENDDO
! Adding up total heat transfer on all boundary faces (including the –ve sign and 2* pi() )
totalQbyT= -(surf1+surf2+surf3)*2*3.141593
!WRITE(10,*)'totalQbyT='
WRITE(10,*)totalQbyT
ENDIF

! We obtain the external entropy generation over the boundaries for one time step
! After the run is complete for specified number of time steps, have to add up all the values in excel
! sheet to get the final answer
RETURN
END SUBROUTINE uout
```

External Entropy Generation Flow-Chart for the "Two-Space" Test Rig.



External Entropy Generation Flow-Chart for the "Two-Space" Test Rig (Cont'd).



External Entropy Generation Subroutine for the "Two-Space" Test Rig.

```
MODULE cfdrc_user
! Declare Global variables
IMPLICIT NONE
INTEGER, PARAMETER :: int_p = SELECTED_INT_KIND(8)
INTEGER, PARAMETER :: string_length = 80
INTEGER, PARAMETER :: real_p = SELECTED_REAL_KIND(8)
INTEGER, PARAMETER :: XDIR = 1, YDIR = 2, ZDIR = 3
REAL(real_p) , PARAMETER :: zero = 0.0d0, one = 1.0d0,two = 2.0d0,&
three = 3.d0, four = 4.0d0, pi = 3.1415926535898d0
END MODULE cfdrc_user
! Declare subroutine name and the variable that is passed to the subroutine
SUBROUTINE uout(iflag)
! DEC$ ATTRIBUTES DLLEXPORT :: uout
! Declare local variable types
USE cfdrc_user, ONLY : int_p, real_p, string_length
! Subroutine defnition starts
INCLUDE 'cfdrc_include'
IMPLICIT NONE
INTEGER(int_p), INTENT(IN) :: iflag
INTEGER(int_p), DIMENSION(:):: bf_index(130), &
bcglobalfaces(130),bc_cells1(130), bc_cells2(130), bc_cells(130)
INTEGER(int_p), DIMENSION(:), ALLOCATABLE :: vccells
REAL(real_p), DIMENSION(:):: tden(130), tempT(130), xc(130), yc(130)
REAL(real_p) :: time, gamma=1.66664527, R=2.0769, Sum1,sum2,exen,&
Qcond,Qconv,T,len,len2,numpi, pi = 3.1415926535898, MT, totalM, bcA,&
MDOT, Tcell, den, s
CHARACTER(len=string_length), DIMENSION(4,4) :: BC
CHARACTER(len=string_length), DIMENSION(4) :: VC = (/"heat &
exch","upper cyl", "mid cyl", "inner cyl"/)
CHARACTER(len=string_length) :: varQ='Q', varT='T', varM='MASS_FLUX',&
Qconduction='Q_COND', Qconvection='Q_CONV', varDEN='RHO', &
vol_name ='upper cyl', ycc='YC', xcc='XC', AREA='AREA', varHC='HC',&
varK='COND'
INTEGER(int_p) :: i, j, k,l, m, bc_type, Qind, Tind, bc_faces,&
Mind, RHOind, vol_index, vc_cells, Xind, Yind, denind, Aind, uind,&
vind, cellind, n_cells, xfind, yfind, tmp1, tmp2, max, min, time_step_no,&
Kind, HCind, b, bc_index, vc_index(4), bc_cell_index, c1, c2, bcfaceindex,&
Qcondind, Qconvind,c
LOGICAL :: error
BC = RESHAPE((/"outer hexwall","outer cylhead","interfaceHE",&
"inner cylhead","inner hexwall","outer cylwall","interfaceUC",&
"interfaceIC","heat exchend","outer piston","interfaceIC",&
```

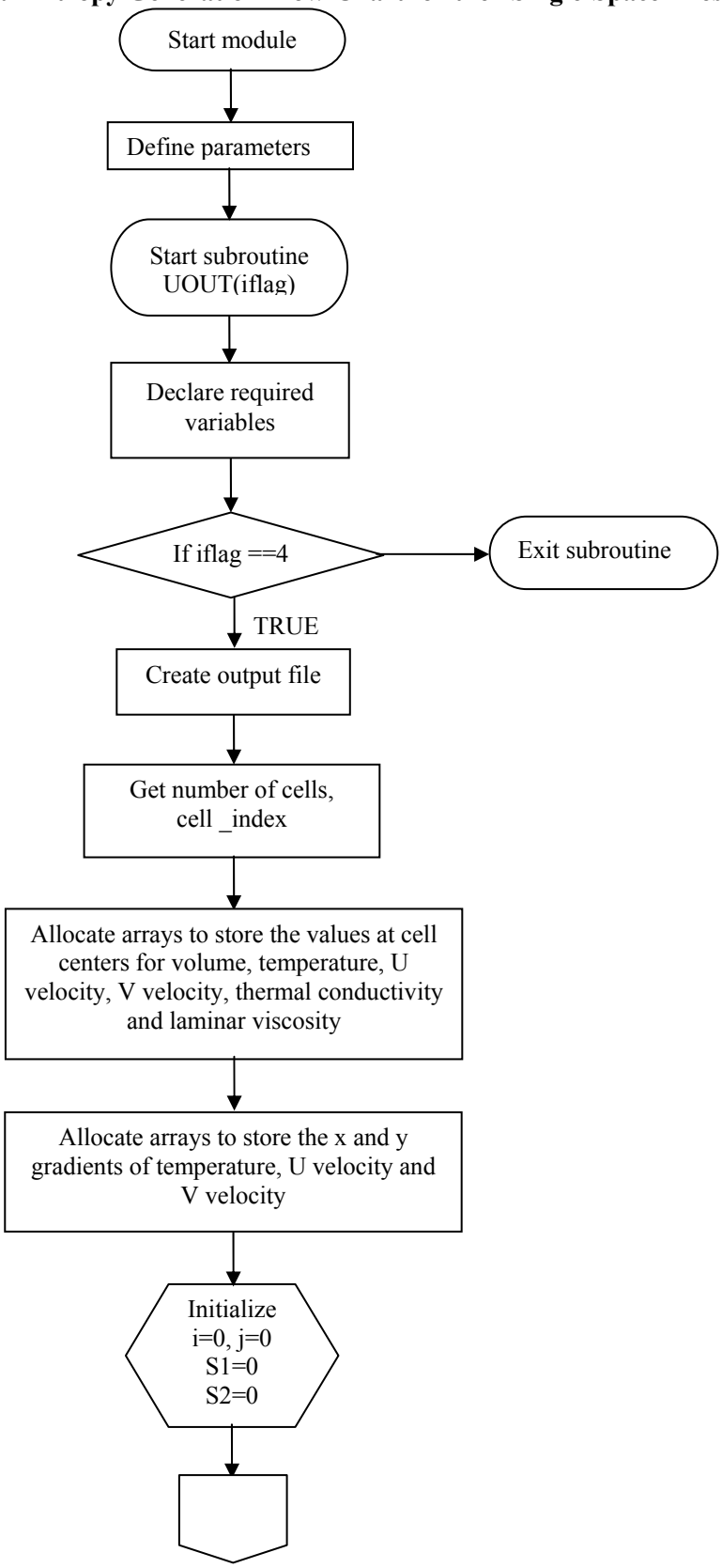
```
“inner piston","interfaceHE","interfaceUC","mid piston",&
"axis symmetry"/),(/4,4/))

! If iflag is equal to 4 to run the following calculation at the end of each time step.

IF(iflag == 4) THEN

! Form the do loop that repeats the calculation inside it for each volume

DO i = 1,4

OPEN(UNIT=i, FILE= VC(i), STATUS='REPLACE', IOSTAT=error)

!open statement opens new output file named as the corresponding volume name

exen = 0

! Value of total entropy for particular time step is initialized to zero and all the variable and volume
! indexes are taken according to particular variable names which are already identified by CFD-
! ACE-SOLVER. All these variable are named in character type !variables defined at the beginning
! of the subroutine.

CALL get_var_index(xcc, Xind, error)

CALL get_var_index(ycc, Yind, error)

CALL get_var_index(varQ, Qind, error)

CALL get_var_index(varT, Tind, error)

CALL get_var_index(varM, Mind, error)

CALL get_var_index(varDEN, RHOind, error)

CALL get_var_index(AREA, Aind, error)

CALL get_var_index(Qconduction, Qcondind, error)

CALL get_var_index(Qconvection, Qconvind, error)

CALL get_time(time, time_step_no, error)

CALL get_vc_index(VC(i), vc_index(i), error)

! The total number of cells is identified and the array is allocated that much memory to store all the
! cell indexes

CALL get_cells_vc(vc_index(i), vc_cells, error)

ALLOCATE(vccells(vc_cells))

! For each volume cell corresponding cell index is taken and stored as individual
! element in ‘vccells’ array

DO l = 1, vc_cells

CALL get_cell_index_from_vc(l, vc_index(i), cellind, error)

vccells(l) = cellind

ENDDO

! The highest and lowest cell index in each volume is identified and stored as max and min
! variables respectively

max = MAXVAL(vccells)

min = MINVAL(vccells)

! Do loop is formed to consider each boundary from the volume for which the volume do loop is being
! run

DO j = 1,4
```

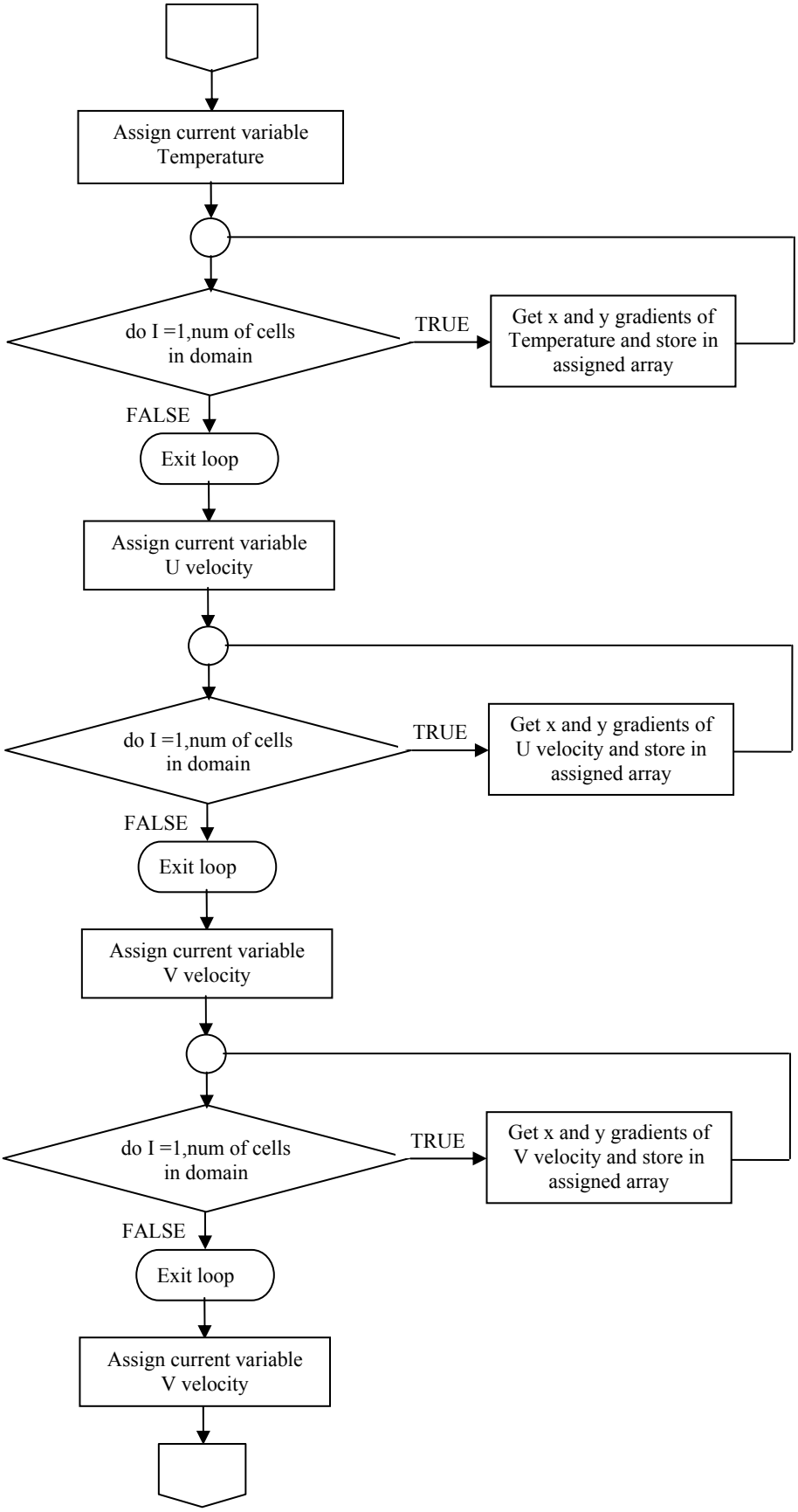
```
! sum1 and sum2 variables are initialized to zero. 'sum1' variable represents the first term in
! external entropy equation and 'sum2' represents the second term
sum1 = 0.0
sum2 = 0.0
! get the boundary index, and boundary face indices
CALL get_bc_index(BC(i,j), bc_index, error)
CALL get_bc_info(bc_index, bc_type, bc_faces, error)
CALL get_bc_global_faces(bc_index, bc_faces, bcglobalfaces, error)
! Form another do loop to get values at each boundary face
DO k = 1, bc_faces
! Inside the do loop, the volume cells adjacent to each face and also the face area for it are taken
CALL get_bc_cell_index(bcglobalfaces(k), bc_cell_index, error)
CALL get_face_cells(bcglobalfaces(k), c1, c2, error)
CALL get_bc_face_index_from_global(bcglobalfaces(k), bcfaceindex, error)
CALL get_value_one_bc_face(bc_index, bcfaceindex, Aind, bcA, error)
! The 'if' loop below determines whether the boundary type is 'wall' or 'interface and accordingly it
! takes corresponding terms in entropy equation to do the calculation. If the two cell indices adjacent
! to the face are same then it is a 'wall' type boundary and only first term in external entropy
! equation should be considered. If they are different then it is an interface and both the terms in
! external entropy equation account for total external !entropy.
if(c1 == c2) then
CALL get_value_one_bc_face(bc_index, bcfaceindex, Qcondind, Qcond, error)
CALL get_value_one_bc_face(bc_index, bcfaceindex, Tind, T, error)
sum1 = sum1 + (Qcond*1e-03*bcA*2*pi*0.000619732292/0.2974715)/T
else
if(c1 <= max .AND. c1 >= min) then
c = c1
else
c = c2
endif
CALL get_value_one_bc_face(bc_index, bcfaceindex, Qconvind, Qconv, error)
CALL get_value_one_bc_face(bc_index, bcfaceindex, Mind, MDOT, error)
CALL get_value_one_cell(RHOind, c, den, error)
CALL get_value_one_cell(Tind, c, T, error)
sum1=sum1+ (Qconv*1e-03*2*pi**bcA*0.000619732292/0.2974715)/T
s = R*(LOG(T)/(gamma-1) + LOG(1/den))
sum2 = sum2 + MDOT*2*pi**bcA*s*0.000619732292/0.2974715
endif
! Close the boundary face do loop
ENDDO
! sum1 and sum2 are added to the previous value of exen
```

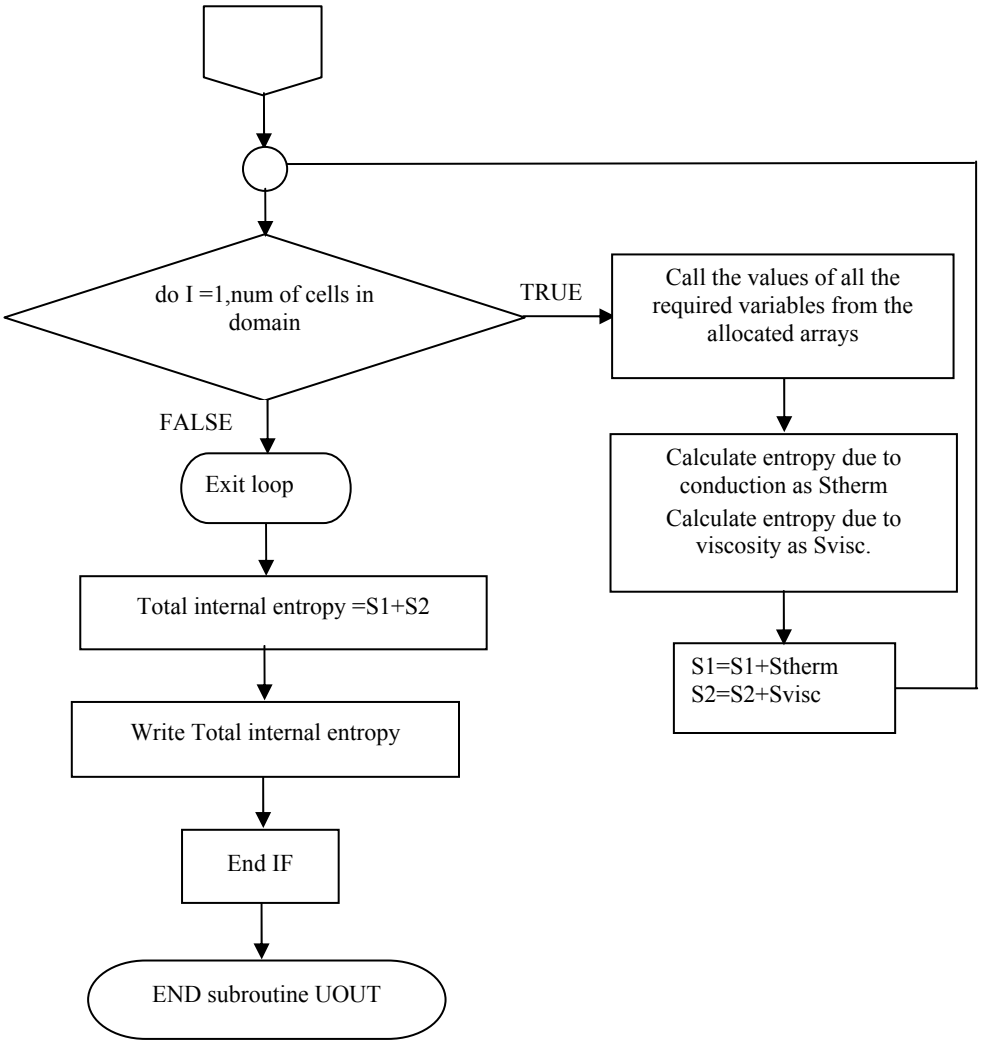
```
exen = exen+sum1+sum2
! Close the boundary do loop
ENDDO
! Finally write the current value of exen and deallocate the 'vccells' array to free the storage for next
! volume condition
WRITE(i,*) exen
DEALLOCATE(vccells)
! Terminate the volume do loop
ENDDO
ENDIF
RETURN
END SUBROUTINE uout
```

Appendix B—Internal Entropy Generation

Internal Entropy Generation Flow-Chart for the "Single-Space" Test Rig.




Assign current variable Temperature
do I =1,num of cells in domain
TRUE
Get x and y gradients of Temperature and store in assigned array
FALSE
Exit loop
Assign current variable U velocity
do I =1,num of cells in domain
TRUE
Get x and y gradients of U velocity and store in assigned array
FALSE
Exit loop
Assign current variable V velocity
do I =1,num of cells in domain
TRUE
Get x and y gradients of V velocity and store in assigned array
FALSE
Exit loop
Assign current variable V velocity


do I =1,num of cells in domain
TRUE
Call the values of all the required variables from the allocated arrays
FALSE
Exit loop
Calculate entropy due to conduction as Stherm
Calculate entropy due to viscosity as Svisc.
Total internal entropy =S1+S2
S1=S1+Stherm
S2=S2+Svisc
Write Total internal entropy
End IF
END subroutine UOUT

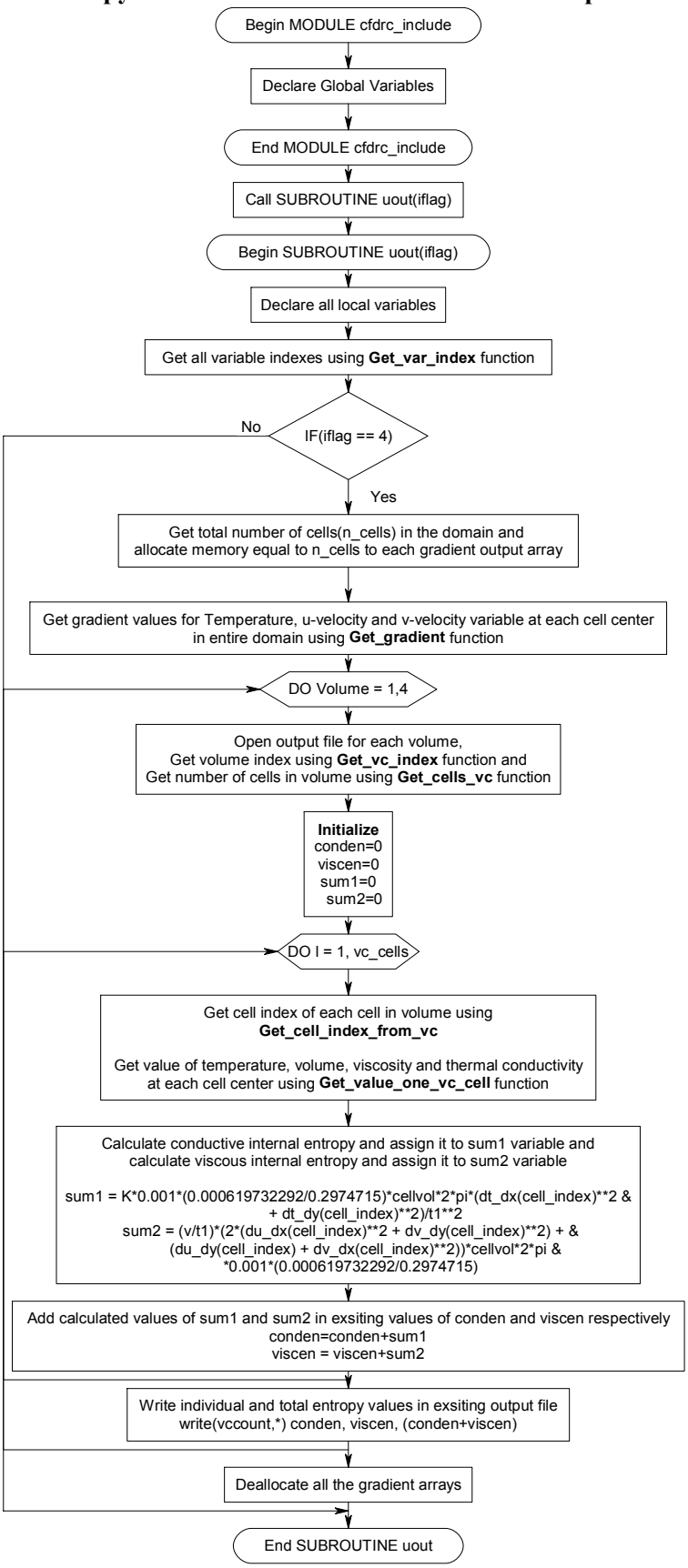
Internal Entropy Generation Subroutine for the "Single-Space" Test Rig.

```
MODULE cfdrc_user
IMPLICIT NONE
INTEGER, PARAMETER :: int_p = SELECTED_INT_KIND(8)
INTEGER, PARAMETER :: string_length = 80
INTEGER, PARAMETER :: real_p = SELECTED_REAL_KIND(8)
INTEGER, PARAMETER :: XDIR = 1, YDIR = 2, ZDIR = 3
REAL(real_p) , PARAMETER :: zero = 0.0d0, one = 1.0d0,two = 2.0d0,&
three = 3.d0, four = 4.0d0, pi = 3.1415926535898d0
SUBROUTINE uout(iflag)
!DEC$ ATTRIBUTES DLLEXPORT :: uout
USE cfdrc_user, ONLY : int_p, real_p, string_length
INCLUDE 'cfdrc_include'
IMPLICIT NONE
! Variable Declarations
INTEGER(int_p), INTENT(IN) :: iflag
REAL(real_p), DIMENSION(:), ALLOCATABLE :: vol,T,U,V,&
&K,mew
REAL(real_p) :: time
REAL(real_p) :: S1,S2,Sint,Stherm,Svisc
CHARACTER(len=string_length) :: var_name ,bc_name,&
& vol_name ='gas spring'
INTEGER(int_p) :: ierror, n_cells, vc_cells, var_index, i,j, &
& vol_index, YC_index, XC_index, cell_index, time_step &
&vc_cell_index,vc_index,bc_index
INTEGER(int_p) :: bc_type, nbc_faces,&
& global_face_index
REAL(real_p) :: bc_value
REAL(real_p), DIMENSION(589):: gxT,gxU,gxV,gyT,gyU,gyV
INTEGER(int_p),DIMENSION(589):: bc_cell_index
INTEGER(int_p) :: ncells
REAL(real_p), DIMENSION(589):: dphidx, dphidy, dphidz
LOGICAL :: error
IF(iflag == 4) THEN
OPEN (UNIT=8, FILE='output.OUT', STATUS='REPLACE', IOSTAT=ierror)
! Begin program for each time step
CALL get_vc_index(vol_name, vol_index, error)
CALL get_cells(n_cells, error)
CALL get_num_bc_faces(nbc_faces, error)
! Allocating arrays for volume, temp., U-velocity, V-velocity, thermal conductivity,
! laminar viscosity
var_name='VOLUME'
```

```
CALL get_var_index(var_name, var_index, error)
ALLOCATE(vol(n_cells))
var_name='T'
CALL get_var_index(var_name, var_index, error)
ALLOCATE(T(n_cells))
var_name='U'
CALL get_var_index(var_name, var_index, error)
ALLOCATE(U(n_cells))
var_name='V'
CALL get_var_index(var_name, var_index, error)
ALLOCATE(V(n_cells))
var_name='THERMAL_CONDUCTIVITY'
CALL get_var_index(var_name, var_index, error)
ALLOCATE(K(n_cells))
var_name='Laminar_Viscosity'
CALL get_var_index(var_name, var_index, error)
ALLOCATE(mew(n_cells))
! Time step
CALL get_time(time, time_step, error)
! Initializing counter variables
j=0
i=0
S1=0
S2=0
! Getting the gradients of required variables
CALL get_cells(ncells, error)
var_name='T'
CALL get_var_index(var_name, var_index, error)
CAll get_gradient(var_index, dphidx, dphidy, dphidz,ncells, error)
Do i=1,589
gxT(i)=dphidx(i)
gyT(i)=dphidy(i)
ENDDO
var_name='U'
CALL get_var_index(var_name, var_index, error)
CAll get_gradient(var_index, dphidx, dphidy, dphidz,ncells, error)
Do i=1,589
gxU(i)=dphidx(i)
gyU(i)=dphidy(i)
ENDDO
```

```
var_name='V'
CALL get_var_index(var_name, var_index, error)
CAll get_gradient(var_index, dphidx, dphidy, dphidz,ncells, error)
Do i=1,589
gxV(i)=dphidx(i)
gyV(i)=dphidy(i)
ENDDO
! Getting thermal conductivity, laminar viscosity, volume and temperature of all the cells
! and allocating to the assigned arrays
DO j=1,589
CALL get_cell_index_from_vc(j, vol_index, cell_index, error)
var_name='THERMAL_CONDUCTIVITY'
CALL get_var_index(var_name, var_index, error)
CAll get_value_one_cell(var_index, cell_index, K(j), error)
var_name='Laminar_Viscosity'
CALL get_var_index(var_name, var_index, error)
CAll get_value_one_cell(var_index, cell_index, mew(j), error)
var_name='VOLUME'
CALL get_var_index(var_name, var_index, error)
CAll get_value_one_cell(var_index, cell_index, vol(j), error)
var_name='T'
CALL get_var_index(var_name, var_index, error)
CAll get_value_one_cell(var_index, cell_index,T(j), error)
! Calculating the internal entropy due to conduction and viscosity using all the values
! obtained
Stherm=(K(j)/T(j)**2)*(gxT(j)**2+gyT(j)**2)&
&*vol(j)*2*3.1415926535898
Svisc=(2*mew(j)/T(j))*((gxU(j))**2+(gyV(j))**2+&
&0.5*(gyU(j)+gxV(j))**2)*vol(j)*2*3.1415926535898
! Adding up the entropies over the volume
S1=S1+Stherm
S2=S2+Svisc
ENDDO
Sint=S1+S2
WRITE(8,*)'Sinternal=',Sint
! We obtain the internal entropy generation over the volume for one time step.
! After the run is complete for specified number of time steps, have to add up all the values
! in excel sheet to get the final answer
ENDIF
RETURN
END SUBROUTINE uout
```

Internal Entropy Generation Flow-Chart for the "Two-Space" Test Rig.



Internal Entropy Generation Flow-Chart for the "Two-Space" Test Rig.

```
MODULE cfdrc_user
! Declare Global variables
IMPLICIT NONE
INTEGER, PARAMETER :: int_p = SELECTED_INT_KIND(8)
INTEGER, PARAMETER :: string_length = 80
INTEGER, PARAMETER :: real_p = SELECTED_REAL_KIND(8)
INTEGER, PARAMETER :: XDIR = 1, YDIR = 2, ZDIR = 3
REAL(real_p) , PARAMETER :: zero = 0.0d0, one = 1.0d0,two = 2.0d0,&
three = 3.0d0, four = 4.0d0, pi = 3.1415926535898d0
END MODULE cfdrc_user
! Declare subroutine name and the variable that is passed to the subroutine
SUBROUTINE uout(iflag)
!DEC$ ATTRIBUTES DLLEXPORT :: uout
! Declare local variable types
USE cfdrc_user, ONLY : int_p, real_p, string_length
! Subroutine definition starts
INCLUDE 'cfdrc_include'
IMPLICIT NONE
! Declare all local variables used in subroutine uout
INTEGER(int_p), INTENT(IN) :: iflag
INTEGER(int_p) :: vc_index(4), cell_index, ncell_faces, vc_cells, i, cell_faces(4), Xind, Yind, xfcind, yfcind, j, c1,
c2, x(4), y(4), m, n,l,c3,c4, T_index, bc_cell_index, numitems,n_cells,volind, smallestitem, & locationsmallest, h,
Aind, vccount,t, mnf, mxf,f, viscind, condind, uind, vind
REAL(real_p), DIMENSION(:), allocatable :: dt_dx, dt_dy, dt_dz, du_dx, du_dy, du_dz, dv_dx, dv_dy, & dv_dz
INTEGER, DIMENSION(1) :: MINLOC_array
REAL(real_p) :: xfcor(4), yfcor(4), xc1, yc1, xc2,cellvol, pi = 3.1415926535898, yc2, fcx, fcy, fa, t1, t2, & sum1,
conden, condeny, viscen, sum2, v, K
CHARACTER(len=string_length) :: vc_name = 'heat exch', ycc='YC', vol='VOLUME', xcc='XC', & xfc='XF',
yfc='YF', T_name='T', A = 'AREA', conductivity = 'COND', viscosity = 'Vis',uvel='U', vvel='V'
CHARACTER(len=string_length), DIMENSION(4) :: VC = (/"heat exch","upper cyl", "mid cyl", &
"inner cyl"/)
LOGICAL :: error
! Declare all the variable indices
CALL get_var_index(xcc, Xind, error)
CALL get_var_index(ycc, Yind, error)
CALL get_var_index(xfc, xfcind, error)
CALL get_var_index(yfc, yfcind, error)
CALL get_var_index('AREA', Aind, error)
CALL get_var_index(T_name, T_index, error)
CALL get_var_index(vol, volind, error)
```

```
CALL get_var_index(conductivity, condind, error)
CALL get_var_index(viscosity, viscind, error)
CALL get_var_index(uvel, uind, error)
CALL get_var_index(vvel, vind, error)
! Declare value of iflag variable equal to 4 to run the following calculation at the end of each
! time step.
IF(iflag == 4) THEN
! Get number of cell in whole domain and allocate that much memory to all the gradient
! variable
Call get_cells(n_cells, error)
allocate(dt_dx(n_cells))
allocate(dt_dy(n_cells))
allocate(dt_dz(n_cells))
allocate(du_dx(n_cells))
allocate(du_dy(n_cells))
allocate(du_dz(n_cells))
allocate(dv_dx(n_cells))
allocate(dv_dy(n_cells))
allocate(dv_dz(n_cells))
! 'get_gradient' function is used to get the temperature, u-velocity and v-velocity gradients at
! each cell center of the domain
CALL get_gradient(T_index, dt_dx, dt_dy, dt_dz, n_cells, error)
CALL get_gradient(uind, du_dx, du_dy, du_dz, n_cells, error)
CALL get_gradient(vind, dv_dx, dv_dy, dv_dz, n_cells, error)
! Do loop is formed to repeat the set of calculation for each volume
DO vccount = 1,4
! Open statement opens new output file for each volume
OPEN(UNIT=vccount, FILE= VC(vccount), STATUS='REPLACE', IOSTAT=error)
! Get the volume index and number of cells in each volume
Call get_vc_index(VC(vccount), vc_index(vccount), error)
Call get_cells_vc(vc_index(vccount), vc_cells, error)
! Initialize the conden viscen, sum1 and sum2 variables as zero. These variables are used to
! store calculated value temporarily for every time step
conden=0
viscen=0
sum1=0
sum2=0
! Do loop is formed to repeat the calculation inside it for every cell in the particular volume
! for which the volume do loop is being run
Do i = 1, vc_cells
! Get the cell index, temperature at cell center, volume of each cell, viscosity and
```

```
! conductivity at cell center
Call get_cell_index_from_vc(i, vc_index(vccount), cell_index, error)
CALL get_value_one_cell(T_index, cell_index, t1, error)
CALL get_value_one_cell(volind, cell_index, cellvol, error)
CALL get_value_one_cell(viscind, cell_index, v, error)
CALL get_value_one_cell(condind, cell_index, K, error)
! Conductive entropy generated is calculated at every cell index and temporarily assigned to
! sum1 variable
sum1 = K*0.001*(0.000619732292/0.2974715)*cellvol*2*pi*(dt_dx(cell_index)**2 + &
dt_dy(cell_index)**2)/t1**2
! Viscous entropy generated is also calculated at the cell center and temporarily assigned to
! sum2 variable
sum2 = (v/t1)*(2*(du_dx(cell_index)**2 + dv_dy(cell_index)**2) + (du_dy(cell_index) + &
dv_dx(cell_index)**2))*cellvol*2*pi *0.001*(0.000619732292/0.2974715)
! sum1 and sum2 variable values are added to previous values of conden and viscen
! variables respectively. This enables the summation of the value calculated value at one cell
! center to the value at next center
conden=conden+sum1
viscen = viscen+sum2
! Do loop for volume cells is terminated
Enddo
! Finally summation of conductive and viscous entropy generation for each volume is
! written in it's corresponding output file. 'conden+viscen' is the total internal entropy
! generated for particular volume condition at the end of each time step.
write(vccount,*) conden, viscen, (conden+viscen)
! Do loop for volume condition is terminated
Enddo
! Free the storage that is used to store all the gradient values. This allows to assign new
! memory locations as required for new volume condition
deallocate(dt_dx)
deallocate(dt_dy)
deallocate(dt_dz)
deallocate(du_dx)
deallocate(du_dy)
deallocate(du_dz)
deallocate(dv_dx)
deallocate(dv_dy)
deallocate(dv_dz)
ENDIF
RETURN
END SUBROUTINE uout
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2005	Final Contractor Report

4. TITLE AND SUBTITLE

Second Law Analysis of Sage and CFD-ACE Models of MIT Gas Spring and "Two-Space" Test Rigs

5. FUNDING NUMBERS

WBS–22–972–30–01
NAG3–2819

6. AUTHOR(S)

Asuquo B. Ebiana, Rupesh Savadekar, and Aparna Vallury

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Cleveland State University
1983 E. 24th Street
Cleveland, Ohio 44115–2440

8. PERFORMING ORGANIZATION REPORT NUMBER

E–15120

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR—2005–213637

11. SUPPLEMENTARY NOTES

Project Manager, Roy C. Tew, Thermal Energy Conversion Division, NASA Glenn Research Center, organization code RPT, 216–433–8471.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category: 20

Available electronically at http://gltrs.grc.nasa.gov

This publication is available from the NASA Center for AeroSpace Information, 301–621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

Thermodynamic second law analysis is applied to two-dimensional axisymmetric computer models of two reciprocating MIT test rigs—gas spring and modified "gas spring plus heat exchanger" test rigs. This work is the first step in application of this type of analysis to a multi-dimensional computer model of a complete Stirling engine.

14. SUBJECT TERMS

Stirling engines; Computerized simulation; Unsteady flow; Convective heat transfer; Second law analysis

15. NUMBER OF PAGES

47

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	