CONFIDENTIAL
NOT FOR PUBLIC DISSEMINATION

THE DEVELOPMENT AND OPTIMIZATION OF LIQUID PISTON STIRLING ENGINES

Daniel A. Walsh

University Undergraduate Fellow

1995-1996

Texas A&M University

Department of Chemical Engineering

APPROVED

Fellows Advisor Mark Holtzapple

Honors Director Suzanne Fennell

ABSTRACT

THE DEVELOPMENT AND OPTIMIZATION OF A LIQUID PISTON STIRLING ENGINE

Daniel A. Walsh (Dr. Mark Holtzapple), Chemical Engineering, Texas A&M University.

The liquid piston Stirling engine is an environmentally conscious technology developed to replace the electric heat pump and the gas furnace. Dr. Holtzapple's innovative modifications to existing Stirling engine technology has lead to promising results. Working engines based on this "bounce chamber" modification have been built; however, they have exhibited poor performance. Rather than investing more funds in equipment modifications, it was decided that the system should first be modeled. This would hopefully provide great insight into what needed to be done to improve engine performance.

A Visual Basic program was developed to model the engine. This graphically oriented program was designed specifically to give the user a clear understanding of how key engine parameters related to engine performance. A FORTRAN multidimensional global optimization program was also written because of the enormous computational requirements required for proper parameter optimization. Results to date show a nine fold increase in engine power. More trials are required before the engine parameters are fully optimized and trends are developed.

TABLE OF CONTENTS

ABSTRACT ii

TABLE OF FIGURES iv

INTRODUCTION 1

REPORT OVERVIEW 1
HISTORY OF STIRLING ENGINE TECHNOLOGY 1
APPLICATIONS 2
ADVANTAGES OF STIRLING ENGINES 3
Environmental 3
Alternative Energy Sources 3
Safe Working Fluids 4
Engine Design & Maintenance 4
Economic 4
PROJECT HISTORY 5

THEORY 7

STIRLING CYCLES & ENGINES 7
STIRLING ENGINE COMPONENTS 10
Bounce Chamber 11
Regenerator 12
Snubber 13

DEVELOPMENT OF A LIQUID PISTON STIRLING ENGINE MODEL 13

DEVELOPMENT OF OPTIMIZATION CODE 17

SIMANN 18

RESULTS AND DISCUSSION 19

CONCLUSIONS 22

LITERATURE CITED 23

APPENDIX A: COMPUTER MODEL DEVELOPMENT 28

SAMPLE CALCULATIONS 28

APPENDIX B: VISUAL BASIC MODEL 35

APPENDIX C: FORTRAN OPTIMIZATION CODE 57

TABLE OF FIGURES

FIGURE 1: SIMPLE STIRLING WATER PUMP 2

FIGURE 2: COMPOUND STIRLING HEAT PUMP 3

FIGURE 3: STIRLING ENGINE QUADRAPLEX 6

FIGURE 4: STIRLING HEAT PUMP TEMPERATURE-ENTROPY DIAGRAM 8

FIGURE 5: STIRLING HEAT PUMP PRESSURE-VOLUME DIAGRAM 9

FIGURE 6: BASIC STIRLING FLUIDYNE 10

FIGURE 7: STIRLING ENGINE DIAGRAM 11

FIGURE 8: INTERFACE SCREEN 14

FIGURE 9: SCHEMATIC SCREEN 15

FIGURE 10: ENGINE PARAMETERS SCREEN 15

FIGURE 11: CHANGE VIEWS SCREEN 16

FIGURE 12: P-V CURVE WITH LIQUID PISTONS SCREEN 16

FIGURE 13: OPTIMAL LIQUID LEVEL COMPARISON 20

FIGURE 14: OPTIMAL P-V CURVE COMPARISON 21

FIGURE 15: OPTIMAL POWER OUTPUT COMPARISON 21

INTRODUCTION

REPORT OVERVIEW

The two primary purposes of this report are as follows:

- to present how computer programs were developed to model and optimize a liquid-piston Stirling engine
- to present how to use these programs

Although a summary of the results generated by these programs will be presented, a rigorous analysis of the data and its implications is beyond the scope of this report.

HISTORY OF STIRLING ENGINE TECHNOLOGY

As early as 1698, Thomas Savery developed a commercial business out of heat engines used to pump water (West, 1983). Years later in 1853, Sir William Thomas, Lord Kelvin, formally derived and documented the fact that engine-driven heat pumps could be designed. In the first half of the nineteenth century, Robert Stirling conceived and designed hot-air engines. Engines based on the Stirling cycle named after him have been developed; however, heat pump and refrigeration applications have not been given much attention. The commercial success of Stirling-cycle heat pumps has been limited to cryogenic applications, where they have technological advantages over other practical technologies (Wurm, 1990).

In the 1930's, technological advancements and market conditions were right for the development of small engine-driven air conditioning and heat pump equipment. A heat pump is driven by an engine. Attempts were made to integrate these two components into one unit. An integrated unit based on a Stirling cycle has potential efficiency advantages over other practical technologies. Unfortunately, coupling the heat pump with an engine resulted in high overall equipment costs. The cost savings from lower operating costs was not able to sufficiently compensate for the higher equipment costs. Therefore, less expensive vapor-compression heat pumps driven by electric motors dominate today's market (Wurm, 1990).

APPLICATIONS

Two feasible applications for liquid piston Stirling engines include pumping fluids and replacing the electric heat pump or gas furnace. A simple Stirling engine designed for pumping water is shown in the following figure. The oscillating water columns in the U tube provide the pressure driving force for the water pump.

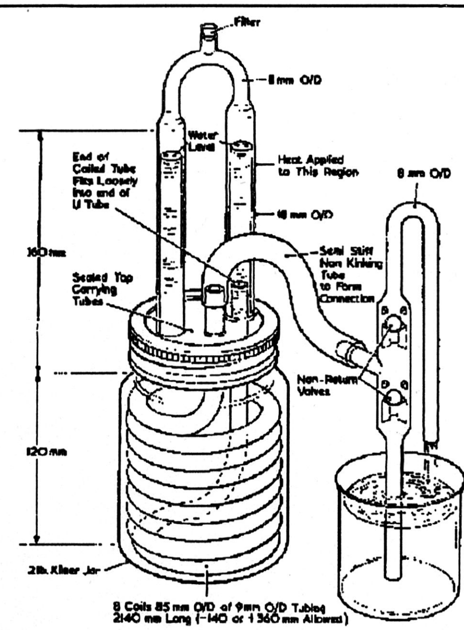


Figure 1: Simple Stirling Water Pump

(West, 1983)

Stirling engines also have the potential to replace the electric heat pump and gas furnace. To date, no commercial heat pumps and few prototypes have been build serve this end. One potential engine design for this type of application is shown below.

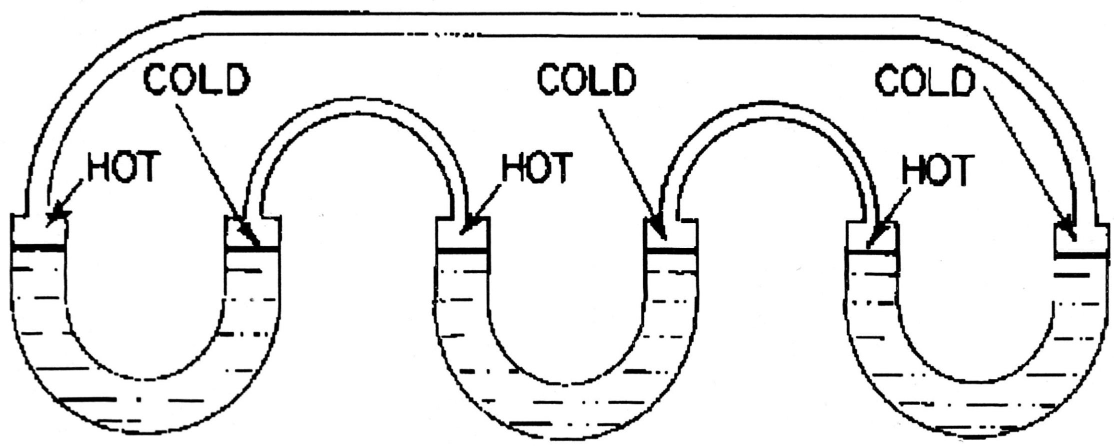


Figure 2: Compound Stirling Heat Pump

(West, 1983)

ADVANTAGES OF STIRLING ENGINES

ENVIRONMENTAL

ALTERNATIVE ENERGY SOURCES

In the United States, space conditioning accounts for approximately one third of all the fossil fuel consumed for purposes other than transportation (Wurm, 1990). In recent years, widespread concern that the use fossil fuels has been too profligate has grown.

These concerns are based on a variety of problems commonly associated with fossil fuels, such as global warming and pollution. Only heat is required to power a Stirling engine. This heat could come from a variety of more environmentally friendly sources, such as solar energy.

SAFE WORKING FLUIDS

Another important advantage of Stirling engine technology is that the working fluids could be composed of safe materials. The liquid pistons in the engine can be made out of water, and helium gas be used as an inter gas above the pistons.

ENGINE DESIGN & MAINTENANCE

Perhaps the liquid-piston Stirling engine's greatest advantage is its simplicity. It does not require accurately dimensioned cylinders and it permits great flexibility in mechanical design with relatively simple construction. If the liquid piston is water, then the engine and pump can even be integrated into a single system (West, 1983). Liquid piston engines avoid the use of a sliding mechanical seal and have no moving parts. The engine is also self-starting. When the temperature is raised beyond some threshold level, the liquid begins to oscillate of its own accord (West, 1983).

ECONOMIC

Another problem with conventional space conditioning is the dependence on fossil fuels and electricity for energy. Supply-side forces have played havoc with fossil fuel prices throughout the 1970's and 1980's. Electricity is also an expensive energy supply for conventional space conditioning equipment. For these reasons, energy should be used as efficiently as possible.

In principle, heat pumps offer the prospect of efficiencies that cannot be matched by any other means. In fact, the Stirling cycle is equivalent to a Carnot cycle in thermodynamic efficiency. However, the perceived advantages of the new designs have not yet been impressive enough to overcome the large capital investments required for mass production.

PROJECT HISTORY

Dr. Holtzapple first began working with Stirling engines in the mid 1980's while in the U.S. Army. There, he worked on developing a portable cooling unit to be used by infantry. Over the years, he has developed new theories and has built a few functional Stirling engines.

Recently, experiments with a four U-tube design of the Stirling heat pump were performed by Martini. His design exhibited poor performance because the heat pump and the engine were 180° out of phase. This meant the paths of compression and expansion traced the same curve on a P-V diagram. Thus, little or no work was produced by the engine. At low operating pressures, he determined that a slight chilling effect was obtainable; however, because the pressure was low, a negligible amount of work was produced (Nivarthi, 1993).

Holtzapple built an engine according to Martini's design, but it would not run. Later, he modified Martini's design by adding bounce chambers to the engine (see Figure 3).

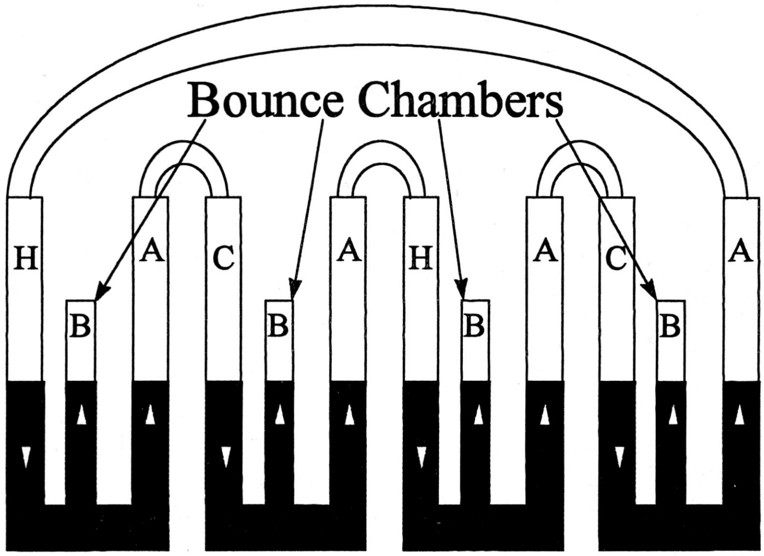


Figure 3: Stirling Engine Quadraplex (PROPRIETARY INFO)

With the addition of bounce chambers, the engine ran, but did not exhibit cooling. Experiments showed that the expected improvements in engine performance were not obtained because the heat pump and engine were still out of phase. Before more capital was spent on equipment modifications, a FORTRAN computer model was written to optimize engine parameters. Although the computer model of the heat pump with bounce chambers was written, optimal tuning parameters were not found. The difficulty of finding the values lies in the immense number of combinations that can be considered. As a result, the search for optimal tuning parameters is comparable to finding the proverbial needle in the haystack.

THEORY

STIRLING CYCLES & ENGINES

Most engineers do not understand the design of equipment based on Stirling cycles and the related Ericsson and Vuilleumier thermodynamic cycles because these cycles have not been in the mainstream of practical applications. The basic principle of the Stirling engine is simple. When a gas is heated, it tends to expand or, if confined in a closed vessel, to rise in pressure (West, 1983). This pressure, along with the force of gravity, cause the liquid columns to oscillate in the U-tubes (see Figure 3).

The following figure shows the temperature-entropy diagram for the Stirling heat pump cycle. The cycle consists of the following sequence of steps:

1 to 2	Isothermal expansion at low source temperature T_{in}, with heat input Q_{in}
2 to 3	Isochoric (constant-volume) compression, with heat input Q_r from the regenerator
3 to 4	Isothermal compression at high sink temperature T_{out}, with heat output Q_{out}
4 to 1	Isochoric expansion, with heat output Q_r stored in the regenerator

The net result is a cooling effect in the cold liquid piston. Reversing this process results in a Stirling engine cycle.

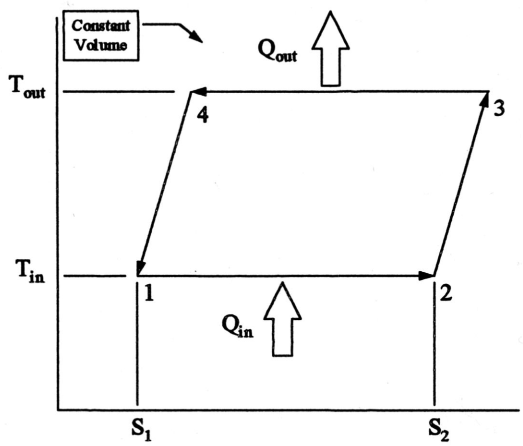


Figure 4: Stirling Heat Pump Temperature-Entropy Diagram

(Wurm, 1990)

The next figure shows a pressure-volume diagram for a Stirling engine cycle. This cycle consists of the following sequence of steps:

1 to 2	Isochoric (constant-volume) expansion, with heat input Q_r from the regenerator
2 to 3	Isothermal expansion at high sink temperature, with heat output Q_{out}
3 to 4	Isochoric compression, with heat output Q_r stored in the regenerator
4 to 1	Isothermal compression at low source temperature, with heat input Q_{in}

The amount of work produced by the engine is calculated by integrating the area inside this P-V curve.

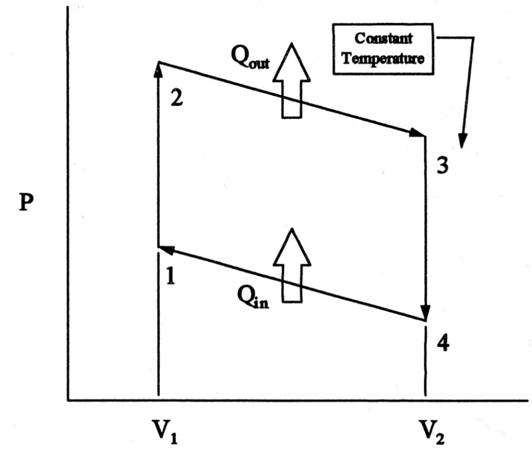


Figure 5: Stirling Heat Pump Pressure-Volume Diagram

A basic Stirling fluidyne unit is presented in the next figure. As the displacer liquid column oscillates in its U tube, the gas above the liquid surface is transferred back and forth between hot and cold spaces. The resulting pressure variations act upon the liquid in the output column, causing it to move (West, 1983).

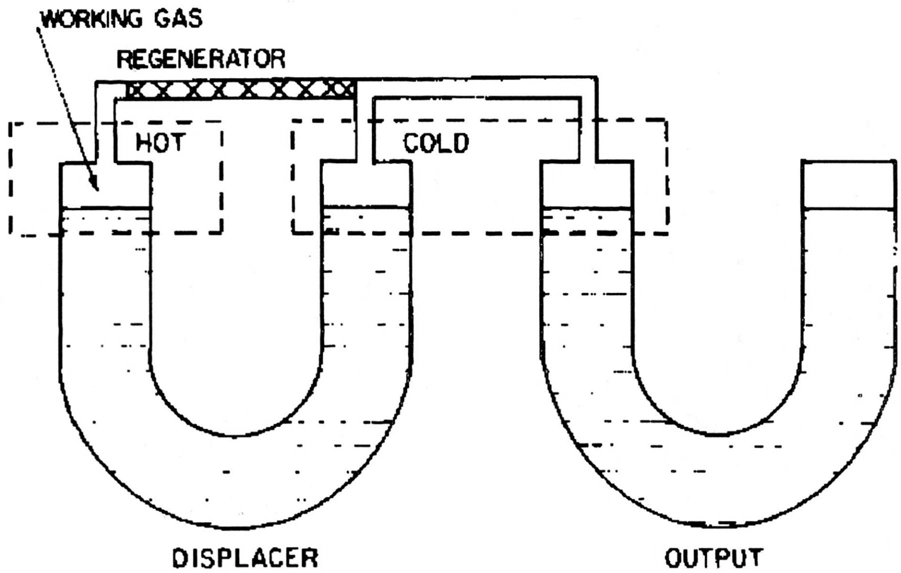


Figure 6: Basic Stirling Fluidyne
(West, 1983)

STIRLING ENGINE COMPONENTS

A schematic of a liquid piston Stirling engine with Holtzapple's modifications is presented in the next figure. Each riser and bounce chamber is partially filled with water. The volume between the water and the top of the riser is occupied by helium, a high thermoconductive gas (Nivarthi, 1993).

A gas burner is attached to the hot leg of the engine. When ignited, the burner heats the gas and causes the gas to expand. This expansion temporarily forces water out the hot leg and into the bounce chamber. A fraction also fills the ambient leg in the adjoining heat pump. The gas in the heat pump is compressed and does work. Gravity then reverses the direction of the water flow. The expanding gas in the heat pump chills the column as the water returns to the engine. This completes one cycle of the Stirling engine.

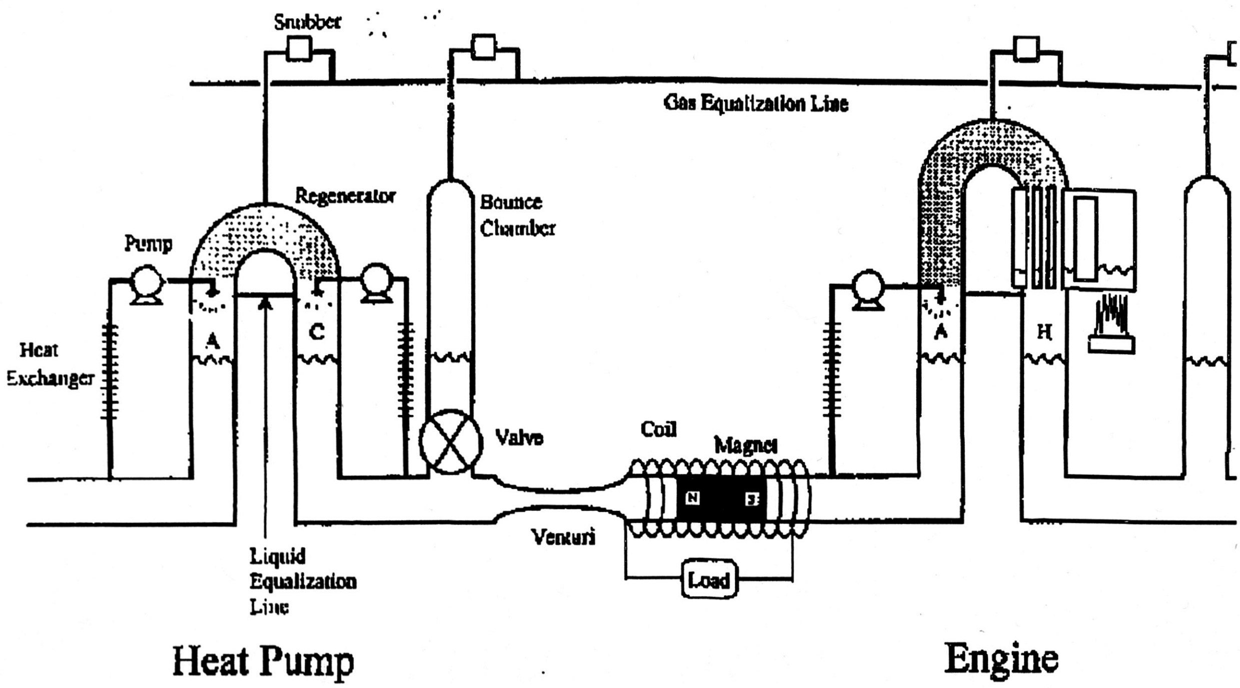


Figure 7: Stirling Engine Diagram (PROPRIETARY INFO)

BOUNCE CHAMBER

The purpose of the bounce chamber is to help tune the engine. This "tuning" is similar to timing an automobile engine to fire so many degrees off top dead center. A poorly tuned engine will not produce much work. The bounce chamber is typically placed closer to the hot or cold leg of the engine or heat pump, respectively. This allows the hot or cold piston to have greater mobility than the ambient piston. This modification increases the engine's potential power output.

REGENERATOR

The heat pump and engine are composed of two risers connected by a regenerator elbow at the top. The regenerator is filled with metal shavings or wire and maintains the same pressure in both risers (Nivarthi, 1993).

The function of the regenerator may best be explained by first considering what would happen in its absence. As the displacer piston is moved from right to left, hot gas would flow through the connecting tube and into the cold end of the cylinder; when it arrived there, it would be cooled down and the heat extracted from it during this cooling process would have to be carried away by the cooling water or air which was being used to keep the right-hand end of the displacer cylinder at a low temperature. This heat would therefore be wasted - and, of course, wasting heat reduces the efficiency of the engine.

With the regenerator present, however, there is a steady fall in temperature along the regenerator, from left to right, as the gas gives up heat to the regenerator material. By the time the gas emerges into the cold end of the cylinder, therefore, it has already been cooled and no extra heat has to be carried away by the coolant.

When the cold gas flows back to the hot cylinder, its temperature gradually increases as it picks up the heat left in the regenerator during its journey to the cold end. This heat is thus not wasted; the regenerator operates as a kind of heat store, and the efficiency is therefore increased (West, 1983).

SNUBBER

The snubber is simply a valve that connects each heat pump, engine, and bounce chamber to a gas equalization line. It maintains the same mean gas pressure in each riser over time (Holtzapple, 1993).

DEVELOPMENT OF A LIQUID PISTON STIRLING ENGINE MODEL

The development of a computer program to model a liquid piston quadraplex Stirling engine began by converting an earlier program developed by previous students from FORTRAN to Microsoft Visual Basic. Visual Basis is a programming language embedded within Microsoft Excel. Essentially, Visual Basic is a programming language which can be used to write Window's applications.

Excel Visual Basic offers several advantages over other programming languages. It is easy to learn and is very user friendly. This in turn helps to create a friendly application for the end user. Raw data generated by a Visual Basic program is readily convertible into graphical formats that are much easier to interpret and analyze. Real time parameter modifications can also be made while the program is running.

To show just how friendly Visual Basic model is, a brief overview of how to use it is presented here. When the file is opened from Excel, the program automatically takes the user to an interface screen, shown in Figure 8 below, where the user has several choices. To initiate a new run, the user clicks on the "One Run" button. This kicks the program off and asks the user a few questions about saving the generated data before continuing on to the next screen.

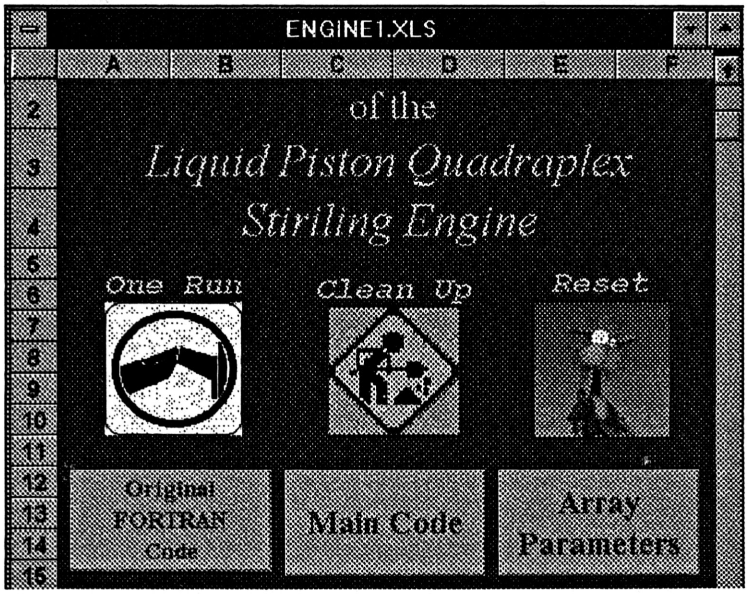


Figure 8: Interface Screen

Next, the user enters the engine parameters into the appropriate boxes on the schematic screen shown in Figure 9. Other less often modified engine parameters can be changed on the parameters screen (see Figure 10). This screen contains all the engine dimensions in addition to operating specifications and other constants. When the "Engage" button is pressed on the engine schematic screen the main program starts running. At any time before the engine reaches steady state, the user can quit and cancel the program or change views by pressing ESC. If the user wants the change views, the view changer screen shown in Figure 11 is pulled up. This screen offers a selection of views and options to change the speed of the oscillating model. One of the views available is presented in Figure 12. This screen displays the liquid piston heights for all the chambers and the current P-V diagram. With a little effort and patience, one can grasp how key design parameters of a Stirling engine effect its performance.

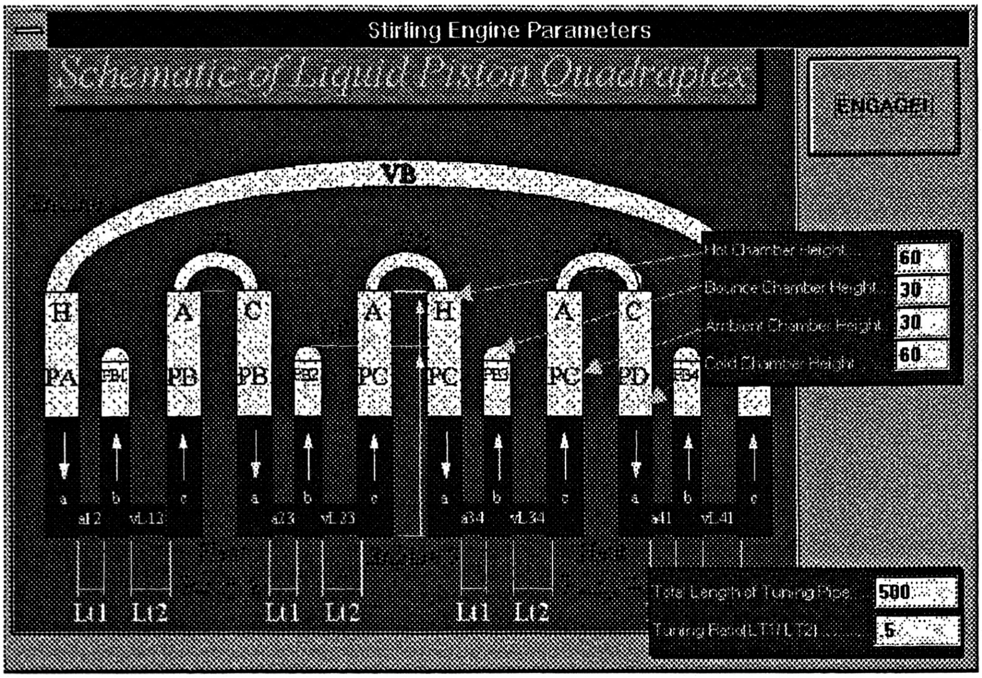
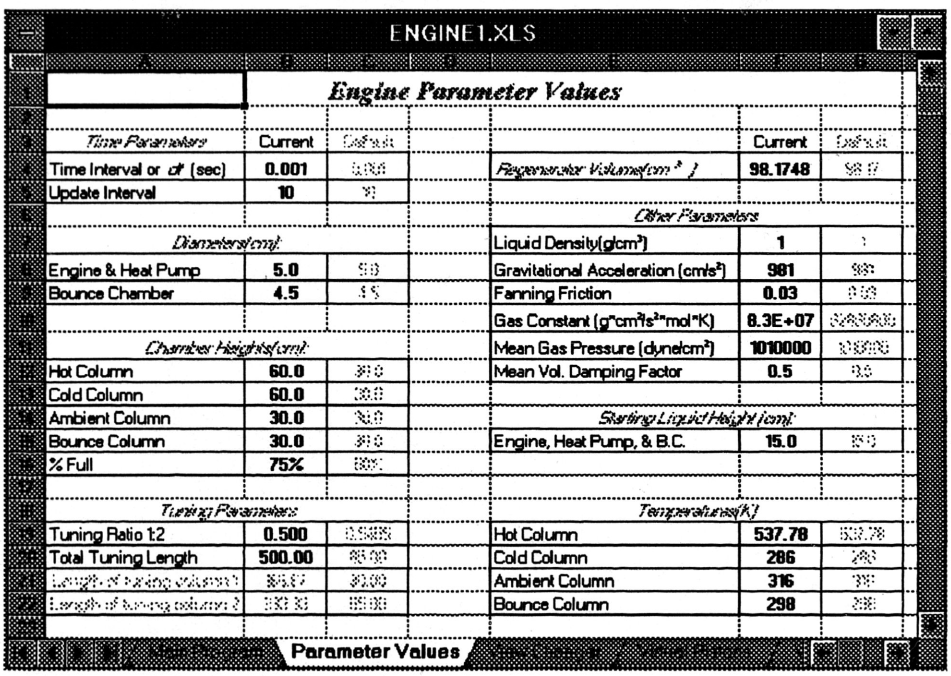


Figure 9: Schematic Screen (PROPRIETARY INFO)



ENGINE1.XLS

Engine Parameter Values

Time Parameters	Current	Default
Time Interval or *dt* (sec)	**0.001**	[illegible]
Update Interval	**10**	[illegible]
Diameters(cm):		
Engine & Heat Pump	**5.0**	[illegible]
Bounce Chamber	**4.5**	[illegible]
Chamber Heights(cm):		
Hot Column	**60.0**	[illegible]
Cold Column	**60.0**	[illegible]
Ambient Column	**30.0**	[illegible]
Bounce Column	**30.0**	[illegible]
% Full	**75%**	[illegible]
Tuning Parameters		
Tuning Ratio 1:2	**0.500**	[illegible]
Total Tuning Length	**500.00**	[illegible]
Length of tuning column 1	[illegible]	[illegible]
Length of tuning column 2	[illegible]	[illegible]

	Current	Default
Regenerator Volume(cm³)	**98.1748**	[illegible]
Other Parameters		
Liquid Density(g/cm³)	**1**	[illegible]
Gravitational Acceleration (cm/s²)	**981**	[illegible]
Fanning Friction	**0.03**	[illegible]
Gas Constant (g*cm²/s²*mol*K)	**8.3E+07**	[illegible]
Mean Gas Pressure (dyne/cm²)	**1010000**	[illegible]
Mean Vol. Damping Factor	**0.5**	[illegible]
Starting Liquid Height (cm):		
Engine, Heat Pump, & B.C.	**15.0**	[illegible]
Temperatures(K)		
Hot Column	**537.78**	[illegible]
Cold Column	**286**	[illegible]
Ambient Column	**316**	[illegible]
Bounce Column	**298**	[illegible]

Parameter Values

Figure 10: Engine Parameters Screen

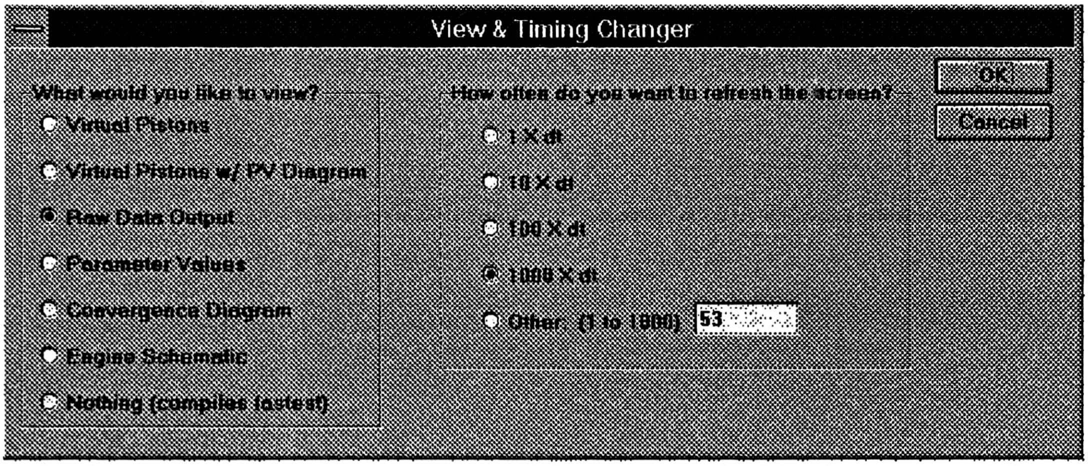


Figure 11: Change Views Screen

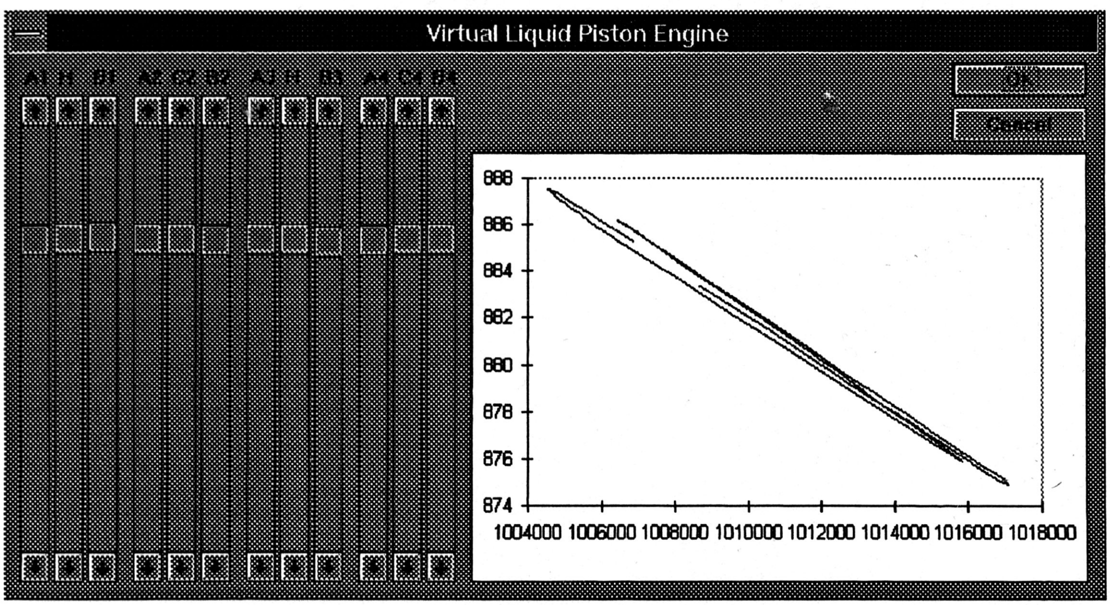


Figure 12: P-V Curve with Liquid Pistons Screen (PROPRIETARY INFO)

DEVELOPMENT OF OPTIMIZATION CODE

With all of the errors worked out of the computer model, engine parameter optimization could begin. To simplify this overwhelming task, only eight key engine parameters were selected for optimization. They are as follows:

1. Tuning ratio - T1/T2
2. Total length of the tuning pipe
3. Height of the hot chamber
4. Height of the cold chamber
5. Height of the ambient chamber
6. Height of the bounce chamber
7. Dead volume in the regenerator
8. Water level in the shortest chamber (%)

Initial optimization efforts involved trying random values for each parameter by hand. It quickly became apparent that a structured optimization approach would be required to obtain good results. A brute force array, varying each parameter over a reasonable operating range, was then considered. However, since the optimal value of one parameter could possibly be influenced by a change in any other, a large array would be required. In fact, if there are no independent variables, testing twenty values for each of the eight key variables would call for an array with over 25 billion elements.

Autonomous optimization programs based of numerical methods were considered next. One attempt involved the development of a master-slave Visual Basic program. The master computer could evaluate results and delegate new tasks to several slave computers over a network drive. This enabled a large number a computers to work in parallel toward a common solution.

Another, approach for optimizing the parameters involved using a "canned" FORTRAN routine based on numerical methods. A routine called Simann was selected from an internet library for this task. Even though this program was limited to serial operation, it had several significant advantages. Since it was FORTRAN based, the program ran faster. A single trial with the Visual Basic code takes approximately twenty minutes to half an hour. The FORTRAN version takes only five minutes. A FORTRAN program could also be run a larger computer, such as workstations and the University's VAX clusters. Use of the Cray super computer is currently being looked into as well.

SIMANN

Simann ("simulated annealing") is a global optimization method that distinguishes between different local optima. Simann, SA, tries to find the global optimum, minimum or maximum, of an N-dimensional function. It moves both up and downhill and as the optimization process proceeds, it focuses on the most promising area. To start, it randomly chooses a trial point within the step length VM (a vector of length N) of the user selected starting point. The function is evaluated at this trial point and its value is compared to its value at the initial point.

In a maximization problem, all uphill moves are accepted and the algorithm continues from that trial point. Downhill moves may be accepted; the decision is made by the Metropolis criteria. It uses TC (temperature) and the size of the downhill move in a probabilistic manner. The smaller T and the size of the downhill move are, the more likely that move will be accepted. If the trial is accepted, the algorithm moves on from that point. If it is rejected, another point is chosen instead for a trial evaluation. Each element of VM periodically adjusted so that half of all function evaluations in that direction are accepted.

A fall in T is imposed upon the system with the RT variable by T(i+1) = RT*T(i) where i is the ith iteration. Thus, as T declines, downhill moves are less likely to be accepted and the percentage of rejections rise. Given the scheme for the selection for VM, VM falls. Thus, as T declines, VM falls and SA focuses upon the most promising area for optimization.

Since the algorithm makes very few assumptions regarding the function to be optimized, it is quite robust with respect to non-quadratic surfaces. In fact, the degree of robustness can be adjusted by the user. Consult the appendix for further explanation concerning how to use the program efficiently.

RESULTS AND DISCUSSION

As mentioned earlier, a rigorous analysis of the generated results and their implications will not be discussed in detail. Rather, a brief overview of the progress made to date will be presented along with future expectations.

In the following figures, an old and poorly tuned engine is compared to a design with new parameters from the SA optimization routine. The old design typically produced 10-15 watts of power per engine, while the new design produces 95-100 watts (P_{ave} = 1 atm, Dia. of pipes = 5cm).

The following figure shows the liquid level in the three chambers of an engine. This data was collected for 2 seconds while the engine was oscillating at a steady state. Note that the new design's oscillations have a larger amplitude. This results in the production of more work.

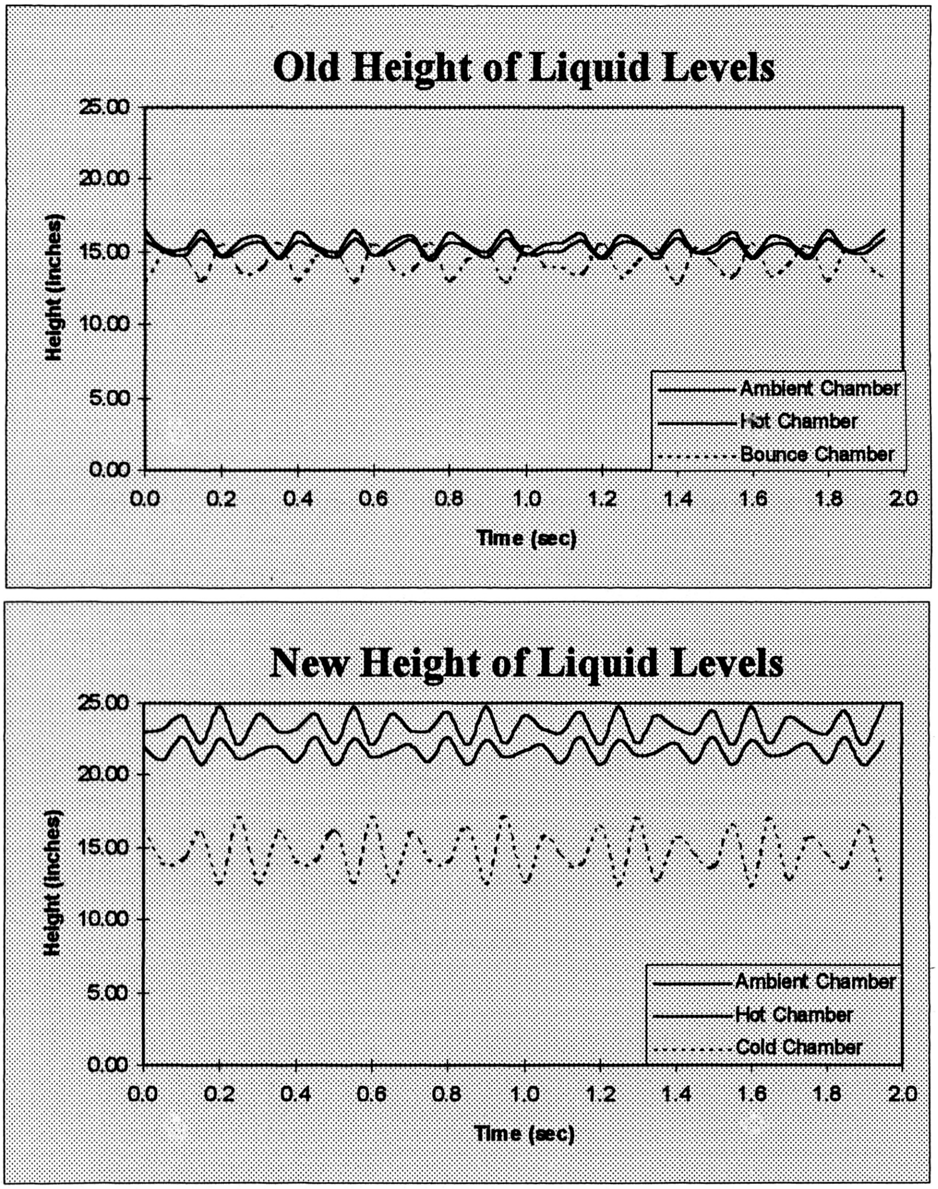


Figure 13: Optimal Liquid Level Comparison

The next figure compares the P-V curves from each design. The larger area within the oval indicates that more work was produced by the new design parameters.

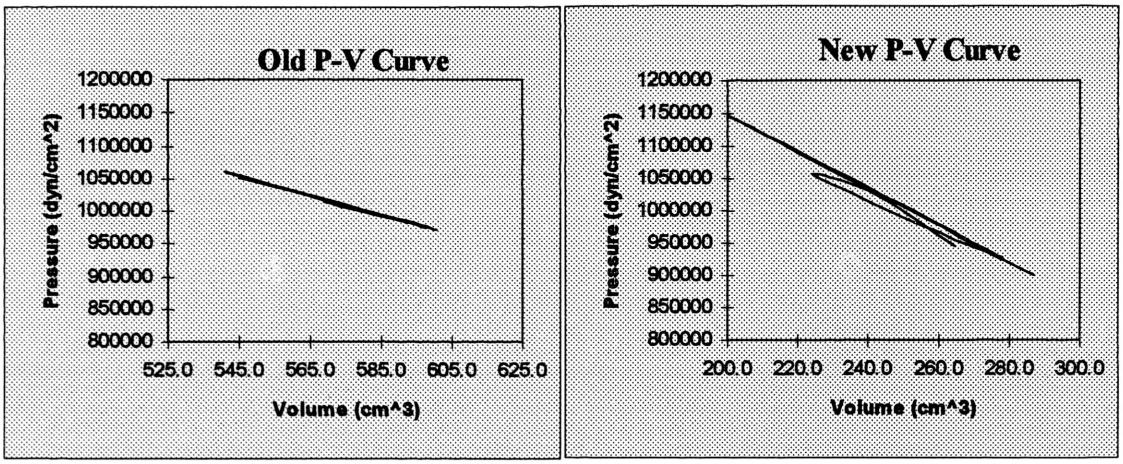


Figure 14: Optimal P-V Curve Comparison

Lastly, a comparison is made between the average power output for each engine design over time.

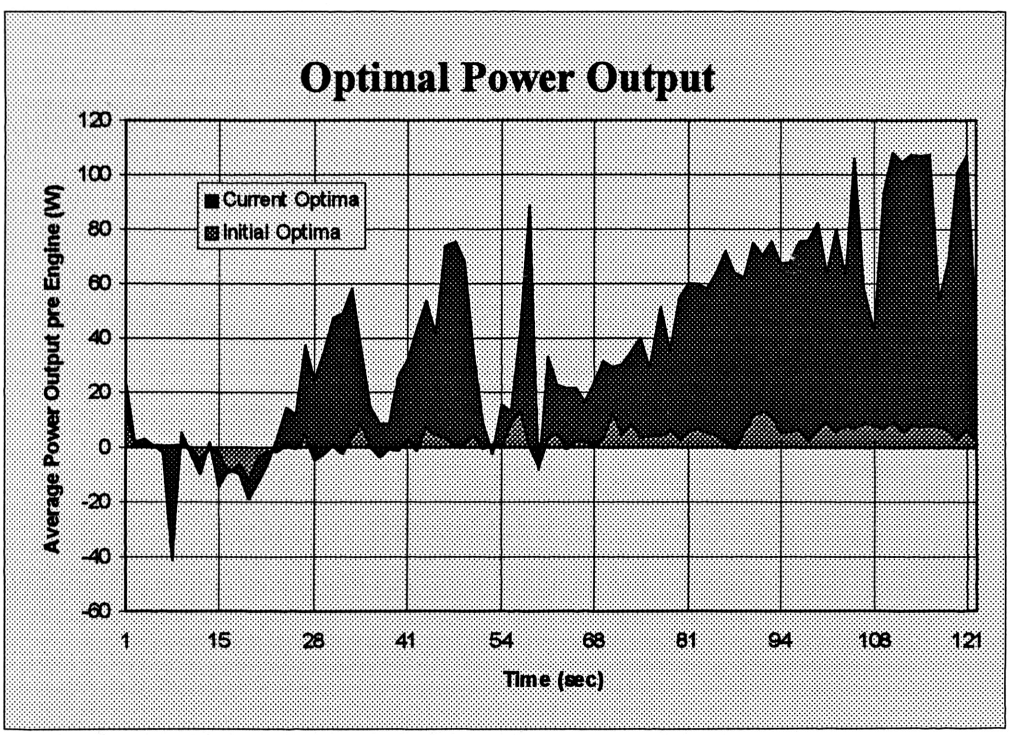


Figure 15: Optimal Power Output Comparison

CONCLUSIONS

Just over three thousand combinations have been tried to date. The initial results are quite encouraging. The engine's power output has as increased nine fold since the optimization process began. Future plans include further optimization trials. Thousands of more are still required to develop accurate trends and correlations relating optimal engine performance to key parameter values. With the parameters optimized, modifications to existing hardware will them be implemented. Experiments with the modified engine will hopefully show that its performance has increased significantly.

LITERATURE CITED

Corana, Al, "Minimizing Multimodal Functions of Continuous Variables with the 'Simulated Annealing' Algorithm." ACM Transactions on Mathematical Software, vol. 13, no. 3, September 1987, pp. 262-280.

Farmer, William, "A Computer Model of the Liquid Piston Quadraplex Stirling Engine," Unpublished Notes, Texas A&M University, College Station, TX, 1994.

Holtzapple, Mark, "Liquid Piston Stirling Heat Pumps," Unpublished Notes, Texas A&M University, College Station, TX, 1993.

Nivarthi, T., "A Model for a Quadraplex Liquid Piston Stirling Heat Pump," Unpublished Notes, Texas A&M University, College Station, TX, 1993.

West, C.D., Liquid Piston Stirling Engines. Van Nostrand Reinhold Company, New York, 1983.

Wurm, Jaroslav, John A. Kinast, Thomas R. Roose, William R. Staats, *Stirling and Vuilleumier Heat Pumps: Design and Applications.* McGraw-Hill, Inc., New York, 1990.

APPENDIX A: COMPUTER MODEL DEVELOPMENT

SAMPLE CALCULATIONS

Presented here is one example (part of pages and pages) calculation that went into the development of this computer model.

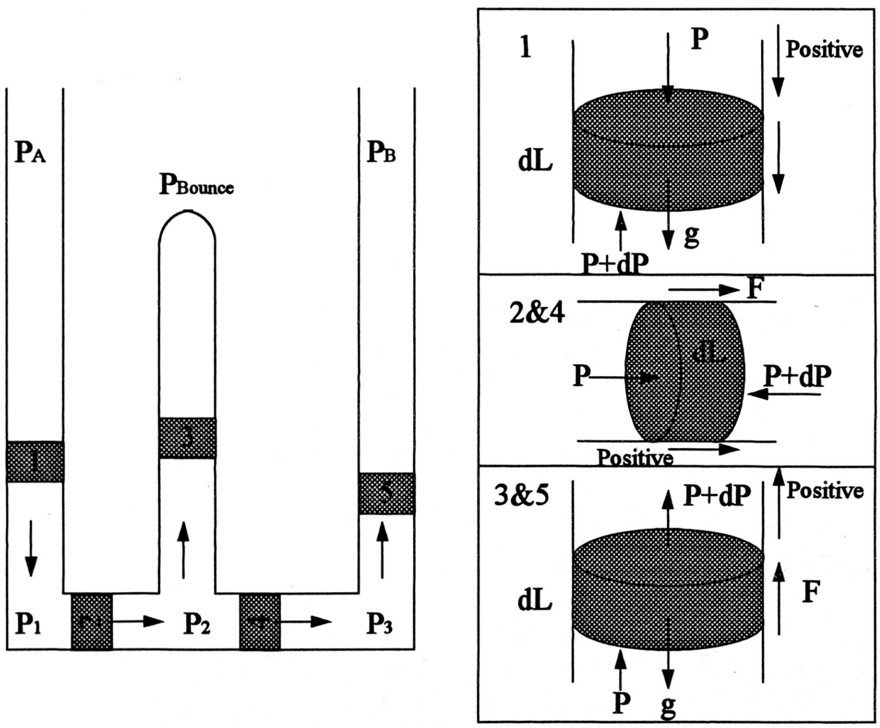


Figure 1A: Differential Fluid Element Diagram

Proprietary Information !

The derivation of P_2 is fairly complex. A system is first defined at the intersection of fluid elements 2, 3, and 4 shown in the above figure. By performing a mass balance on that system, an equation relating individual fluid element accelerations is obtained.

$$\dot{m}_1 = \dot{m}_3 + \dot{m}_5$$

Equation 1: $(\rho A v_l)_1 = (\rho A_{BC} v_l)_3 + (\rho A v_l)_5$

Subscript l refers to a liquid velocity. Since the fluid is assumed to be incompressible, the density variable in Equation 1 falls out.

Equation 2: $(A v_l)_1 = (A_{BC} v_l)_3 + (A v_l)_5$

Equation 2 is then differentiated with respect to time,

$$\left(A\frac{dv_l}{dt}\right)_1 = \left(A_{BC}\frac{dv_l}{dt}\right)_3 + \left(A\frac{dv_l}{dt}\right)_5$$

resulting in

$$(Aa)_1 = (A_{BC}a)_3 + (Aa)_5$$

By assuming $A_1 = A_5 = A$,

Equation 3: $a_1 = \frac{A_{BC}}{A} a_3 + a_5$

DERIVATION OF THE LIQUID ACCELERATION EQUATIONS

A force balance around volume element 1 is first defined.

Equation 4: $F = (dm)a_1 = (dm)g + PA - (P + dP)A + i_1 \frac{1}{2} f\rho\pi D(dL)v_{l1}^2$

where,

dm = mass of the differential volume element

dl = length of the differential volume element

and

$i_1 = -1$ if element 1 has a positve velocity vector

$i_1 = 1$ if element 1 has a negative velocity vector

The first term in Equation 4 corresponds to the force due to gravity. The second is the force due to the pressure difference across the volume element. The last term accounts for the frictional force, which is always acting against the element's velocity vector. The mass of the differential volume element is equivalent to

$$dm = \rho A dL$$

Substituting this into Equation 4 and solving for dP yields

$$(\rho A dL)a_1 = (\rho A dL)g - A dP + i_1 \frac{1}{2} f\rho\pi D(dL)v_{l1}^2$$

$$(\rho dL)a_1 = (\rho dL)g - dP + i_1 \frac{1}{2A} f\rho\pi D(dL)v_{l1}^2$$

Equation 5: $$dP = \left(\rho g + i_1 \frac{1}{2A} f\rho\pi D v_{l1}^2 - \rho a_1\right) dL$$

By inegrating Equation 5 from gas pressure P_A to liquid pressure P_1 ,or from the top to the bottom of the column, an equation representing the entire column is formulated.

$$\int_{P_A}^{P_1} dP = \left(\rho g + i_1 \frac{1}{2A} f\rho\pi D v_{l1}^2 - \rho a_1\right) \int_0^{L_1} dL$$

$$P_1 - P_A = \left(\rho g + i_1 \frac{1}{2A} f\rho\pi D v_{l1}^2 - \rho a_1\right) L_1$$

An expression for the liquid acceleration in the column is obtained by solving for a_1.

$$\rho g L_1 + \frac{i_1 2 f\rho v_{l1}^2 L_1}{D} - \rho a_1 L_1 = P_1 - P_A$$

Equation 6: $$a_1 = \frac{P_A - P_1}{\rho L_1} + g + i_1 \frac{2 f v_{l1}^2}{D}$$

In a similar manner, the liquid acceleration for volume element 2 is derived:

$$F = (dm)a_2 = PA - (P + dP)A + i_1 \frac{1}{2} f\rho\pi D(dL)v_{l1}^2$$

where,

dm = mass of the differential volume element

dl = length of the differential volume element

and

$i_1 = -1$ if element 1 has a positve velocity vector

$i_1 = 1$ if element 1 has a negative velocity vector

$$(\rho A dL)a_2 = -AdP + i_1 \frac{1}{2} f\rho\pi D(dL)v_{l1}^2$$

$$(\rho dL)a_2 = -dP + i_1 \frac{1}{2A} f\rho\pi D(dL)v_{l1}^2$$

$$dP = \left(i_1 \frac{1}{2A} f\rho\pi D v_{l1}^2 - \rho a_2\right) dL$$

$$\int_{P_1}^{P_2} dP = \left(i_1 \frac{1}{2A} f\rho\pi D v_{l1}^2 - \rho a_2\right) \int_0^{L_2} dL$$

$$P_2 - P_1 = \left(i_1 \frac{1}{2A} f\rho\pi D v_{l1}^2 - \rho a_2\right) L_2$$

$$\frac{i_1 2 f\rho v_{l1}^2 L_2}{D} - \rho a_2 L_2 = P_2 - P_1$$

Equation 7: $$a_2 = \frac{P_1 - P_2}{\rho L_2} + i_1 \frac{2 f v_{l1}^2}{D}$$

The derivation of the liquid acceleration for volume element 3 is nearly identical to that of volume element 1. However, in accordance with the previously defined sign convention for positive motion, elements 1 and 3 are moving in opposite directions. Also, the diameter of the bounce chamber is not assumed to be the same as the diameter of the heat pump and engine.

$$F = (dm)a_3 = -(dm)g + PA_{BC} - (P+dP)A_{BC} + i_3\frac{1}{2}f\rho\pi D_{BC}(dL)v_{13}^2$$

where,

dm = mass of the differential volume element

dl = length of the differential volume element

and

$i_3 = -1$ if element 1 has a positve velocity vector

$i_3 = 1$ if element 1 has a negative velocity vector

$$(\rho AdL)a_3 = -(\rho AdL)g - AdP + i_3\frac{1}{2}f\rho\pi D_{BC}(dL)v_{13}^2$$

$$(\rho dL)a_3 = -(\rho dL)g - dP + i_3\frac{1}{2A}f\rho\pi D_{BC}(dL)v_{13}^2$$

$$dP = \left(-\rho g + i_3\frac{1}{2A}f\rho\pi D_{BC}v_{13}^2 - \rho a_3\right)dL$$

$$\int_{P_2}^{P_{Bounce}} dP = \left(-\rho g + i_3\frac{1}{2A}f\rho\pi D_{BC}v_{13}^2 - \rho a_3\right)\int_0^{L_3} dL$$

$$P_{Bounce} - P_2 = \left(-\rho g + i_3\frac{1}{2A}f\rho\pi D_{BC}v_{13}^2 - \rho a_3\right)L_3$$

$$-\rho g L_3 + \frac{i_3 2f\rho v_{13}^2 L_3}{D_{BC}} - \rho a_3 L_3 = P_{Bounce} - P_2$$

Equation 8: $$a_3 = \frac{P_2 - P_{Bounce}}{\rho L_3} - g + i_3\frac{2fv_{13}^2}{D_{BC}}$$

The derivations for the acceleration of elements 4 and 5 are similar to the derivations of 1 and 2 respectively. Consequently, only the equations for the accelerations are presented here.

$$\textbf{Equation 9:}\quad a_4 = \frac{P_2 - P_3}{\rho L_4} + i_5 \frac{2 f v_{l5}^2}{D}$$

$$\textbf{Equation 10:}\quad a_5 = \frac{P_3 - P_B}{\rho L_5} - g + i_5 \frac{2 f v_{l5}^2}{D}$$

DERIVATION OF THE LIQUID PRESSURE EQUATIONS

The derivation of liquid pressure P_1 begins by equating Equation 6 and Equation 7.

$$a_1 = a_2$$

$$\frac{P_A - P_1}{\rho L_1} + g + i_1 \frac{2 f v_{l1}^2}{D} = \frac{P_1 - P_2}{\rho L_2} + i_1 \frac{2 f v_{l1}^2}{D}$$

$$\frac{P_A - P_1}{\rho L_1} + g = \frac{P_1 - P_2}{\rho L_2}$$

$$\frac{P_A}{\rho L_1} - \frac{P_1}{\rho L_1} + g = \frac{P_1}{\rho L_2} - \frac{P_2}{\rho L_2}$$

$$\frac{P_A}{\rho L_1} + g + \frac{P_1}{\rho L_2} = \frac{P_1}{\rho}\left(\frac{1}{L_2} + \frac{1}{L_1}\right)$$

$$\frac{P_A L_2 + g \rho L_1 L_2 + P_2 L_1}{\rho L_1 L_2} = P_1\left(\frac{L_1 + L_2}{\rho L_1 L_2}\right)$$

$$\textbf{Equation 11:}\quad P_1 = \frac{P_A L_2 + g \rho L_1 L_2 + P_2 L_1}{L_1 + L_2}$$

The derivation of P_3 is similar to that of P_1. Equation 9 and Equation 10 are equated, and P_3 is isolated.

Equation 12: $P_3 = \dfrac{P_B L_4 + g\rho L_4 L_5 + P_2 L_5}{L_4 + L_5}$

The derivation of the equation for P_2 is more difficult. Equation 6, Equation 8, and Equation 9 are substituted into Equation 3 and P_2 is isolated.

$$a_1 = \frac{A_{BC}}{A} a_3 + a_5$$

$$\frac{P_A - P_1}{\rho L_1} + g + i_1 \frac{2 f v_{l1}^2}{D} = \frac{D_{BC}^2}{D^2}\left(\frac{P_2 - P_{Bounce}}{\rho L_3} - g + i_3 \frac{2 f v_{l3}^2}{D_{BC}}\right) + \frac{P_3 - P_B}{\rho L_5} - g + i_5 \frac{2 f v_{l5}^2}{D}$$

$$\frac{P_A - P_1}{\rho L_1} + g + i_1 \frac{2 f v_{l1}^2}{D} = \frac{D_{BC}^2}{D^2}\left(\frac{P_2 - P_{Bounce}}{\rho L_3}\right) - g\frac{D_{BC}^2}{D^2} + i_3 \frac{2 f D_{BC} v_{l3}^2}{D^2} + \frac{P_3 - P_B}{\rho L_5} - g + i_5 \frac{2 f v_{l5}^2}{D}$$

$$\frac{D_{BC}^2}{D^2}\left(\frac{P_2 - P_{Bounce}}{\rho L_3}\right) = \frac{P_A - P_1}{\rho L_1} + g + i_1 \frac{2 f v_{l1}^2}{D} + g\frac{D_{BC}^2}{D^2} - i_3 \frac{2 f D_{BC} v_{l3}^2}{D^2} - \frac{P_3 - P_B}{\rho L_5} + g - i_5 \frac{2 f v_{l5}^2}{D}$$

$$P_2 - P_{Bounce} = \frac{D^2 L_3}{D_{BC}^2 L_1}(P_A - P_1) + g\frac{D^2 \rho L_3}{D_{BC}^2} + i_1 \frac{2\rho L_3 f D v_{l1}^2}{D_{BC}^2} + g\rho L_3 - i_3 \frac{2\rho L_3 f v_{l3}^2}{D_{BC}} - (P_3 - P_B)\frac{L_3 D^2}{L_5 D_{BC}^2} + g\frac{D^2 \rho L_3}{D_{BC}^2} - i_5 \frac{2\rho L_3 f D v_{l5}^2}{D_{BC}^2}$$

$$P_2 = P_{Bounce} + \rho g L_3\left(2\frac{D^2}{D_{BC}^2} + 1\right) + 2\rho L_3 f\left(i_1 \frac{D v_{l1}^2}{D_{BC}^2} - i_3 \frac{v_{l3}^2}{D_{BC}} - i_5 \frac{D v_{l5}^2}{D_{BC}^2}\right) + P_A\left(\frac{L_3}{L_1}\right)\left(\frac{D^2}{D_{BC}^2}\right) + P_B\left(\frac{L_3}{L_5}\right)\left(\frac{D^2}{D_{BC}^2}\right) - P_1\left(\frac{L_3}{L_1}\right)\left(\frac{D^2}{D_{BC}^2}\right) - P_3\left(\frac{L_3}{L_5}\right)\left(\frac{D^2}{D_{BC}^2}\right)$$

Substitutions for P_1 and P_3 in terms of P_2 are then made using Equation 11 and Equation 12.

$$P_2 = P_{Bounce} + \rho g L_3\left(2\frac{D^2}{D_{BC}^2} + 1\right) + 2\rho L_3 f\left(i_1 \frac{D v_{l1}^2}{D_{BC}^2} - i_3 \frac{v_{l3}^2}{D_{BC}} - i_5 \frac{D v_{l5}^2}{D_{BC}^2}\right) + P_A\left(\frac{L_3}{L_1}\right)\left(\frac{D^2}{D_{BC}^2}\right) +$$

$$P_B\left(\frac{L_3}{L_5}\right)\left(\frac{D^2}{D_{BC}^2}\right)-\left(\frac{P_A L_2+g\rho L_1 L_2+P_2 L_1}{L_1+L_2}\right)\left(\frac{L_3}{L_1}\right)\left(\frac{D^2}{D_{BC}^2}\right)-\left(\frac{P_B L_4+g\rho L_4 L_5+P_2 L_5}{L_4+L_5}\right)\left(\frac{L_3}{L_5}\right)\left(\frac{D^2}{D_{BC}^2}\right)$$

$$P_2\left(1+\left(\frac{L_B}{L_1+L_2}\right)\left(\frac{D^2}{D_{BC}^2}\right)+\left(\frac{L_B}{L_4+L_5}\right)\left(\frac{D^2}{D_{BC}^2}\right)\right)=P_{Bounce}+\rho g L_3\left(2\frac{D^2}{D_{BC}^2}+1\right)+2\rho L_3 f\left(i_1\frac{Dv_{l1}^2}{D_{BC}^2}-i_3\frac{v_{l3}^2}{D_{BC}}-i_5\frac{Dv_{l5}^2}{D_{BC}^2}\right)+$$

$$P_A\left(\frac{L_3}{L_1}\right)\left(\frac{D^2}{D_{BC}^2}\right)\left(1-\left(\frac{L_2}{L_1+L_2}\right)\right)+P_B\left(\frac{L_3}{L_5}\right)\left(\frac{D^2}{D_{BC}^2}\right)\left(1-\left(\frac{L_4}{L_4+L_5}\right)\right)-\rho g L_B\left(\left(\frac{L_2}{L_1+L_2}\right)+\left(\frac{L_4}{L_4+L_5}\right)\right)\left(\frac{D^2}{D_{BC}^2}\right)$$

$$P_2=\frac{P_{Bounce}+\rho g L_3\left(2\frac{D^2}{D_{BC}^2}+1\right)+2\rho L_3 f\left(i_1\frac{Dv_{l1}^2}{D_{BC}^2}-i_3\frac{v_{l3}^2}{D_{BC}}-i_5\frac{Dv_{l5}^2}{D_{BC}^2}\right)+P_A\left(\frac{L_3}{L_1}\right)\left(\frac{D^2}{D_{BC}^2}\right)\left(1-\left(\frac{L_2}{L_1+L_2}\right)\right)+P_B\left(\frac{L_3}{L_5}\right)\left(\frac{D^2}{D_{BC}^2}\right)\left(1-\left(\frac{L_4}{L_4+L_5}\right)\right)-\rho g L_B\left(\left(\frac{L_2}{L_1+L_2}\right)+\left(\frac{L_4}{L_4+L_5}\right)\right)\left(\frac{D^2}{D_{BC}^2}\right)}{\left(1+\left(\frac{L_B}{L_1+L_2}\right)\left(\frac{D^2}{D_{BC}^2}\right)+\left(\frac{L_B}{L_4+L_5}\right)\left(\frac{D^2}{D_{BC}^2}\right)\right)}$$

APPENDIX B: VISUAL BASIC MODEL

```
Option Base 1
Option Explicit
'
'
    Type Piston

'   Declares the variable type "Piston" used in calculating the mean
volume

        Counter(1 To 12) As Integer
        A1(0 To 20) As Single    'index '0' is reserved for the average
        A2(0 To 20) As Single    'index '0' is reserved for the average
        A3(0 To 20) As Single    'index '0' is reserved for the average
        A4(0 To 20) As Single    'index '0' is reserved for the average
        H1(0 To 20) As Single    'index '0' is reserved for the average
        C2(0 To 20) As Single    'index '0' is reserved for the average
        H3(0 To 20) As Single    'index '0' is reserved for the average
        C4(0 To 20) As Single    'index '0' is reserved for the average
        B1(0 To 20) As Single    'index '0' is reserved for the average
        B2(0 To 20) As Single    'index '0' is reserved for the average
        B3(0 To 20) As Single    'index '0' is reserved for the average
        B4(0 To 20) As Single    'index '0' is reserved for the average
    End Type



Function Engine(Parameters() As Single, Results() As Single) As Integer

'*************Start Error Handler****************

        On Error GoTo ChangeViews
        Application.EnableCancelKey = xlErrorHandler


'***************************************************************
'*****           DECLARATION OF VARIABLES                  *****
'*****     All units are expressed in CGS system           *****
'***************************************************************

'   Pi     -   Pi (any questions?)
Dim Pi As Single

'   g      -   gravitaional acceleration constant
Dim g As Single

'   f       -   Fanning friction factor
Dim f As Single

'   d       -   density of the working fluid (water)
Dim d As Single

'   R       -   gas constant
Dim R As Single

'   Dia***  -   Diameters for the following pipes:
'           -   Tuning pipes, Ambient, hot, and cold columns
```

```
'       _BC  -   Bounce Chamber columns
Dim Dia, Dia_BC As Single

'   Area*** -    Cross-sectional area for the following pipes:
'            -   Tuning pipes, Ambient, hot, and cold columns
'        _BC -   Bounce Chamber columns
Dim Area, Area_BC As Single

'   Dt       -   time interval between iterations
Dim Dt As Single

'   L        -   height of the bounce, ambient, hot, or cold columns
Dim LB, LA, LH, LC As Single

'   LL       -   initial height of the water in all of the columns
Dim LL As Single

'   T*       -   temperature of the following columns:
'       H    -   hot
'       A    -   ambient
'       C    -   cold
'       B    -   bounce
Dim TH, TA, TB, TC As Single

'   Pmean    -   mean gas pressure in the gas equalization line
Dim Pmean As Single

'   VB       -   volume of the regenerator between the ambient and
hot/cold columns
Dim VB As Single

'   Damp     -   damping factor for mean gas volume calculations
Dim Damp As Single

'   L**      -   height of liquid in the ambient, hot, cold, or bounce
chamber
Dim LA1, LH1, LB1, LA2, LC2, LB2, LA3, LH3, LB3, LA4, LC4, LB4 As Single

'   vl***    -   liquid velocities in the columns.  see figure 5 p. 15
Dim vl12a, vl12b, vl12c, vl23a, vl23b, vl23c As Single
Dim vl34a, vl34b, vl34c, vl41a, vl41b, vl41c As Single

'   Vo**     -   volume of gas in each column
Dim VoH1, VoH3, VoC2, VoC4, VoB1, VoB2 As Single
Dim VoB3, VoB4, VoA1, VoA2, VoA3, VoA4 As Single

'   Vo**o    -   initial volume of gas in each column
Dim VoH1o, VoH3o, VoC2o, VoC4o, VoB1o, VoB2o As Single
Dim VoB3o, VoB4o, VoA1o, VoA2o, VoA3o, VoA4o As Single

'   P**      -   pressure of gas in each column
Dim PA, PB, PC, PD, PB1, PB2, PB3, PB4 As Single

'   P**o     -   initial pressure of gas in each column
Dim PAo, PBo, PCo, PDo, PB1o, PB2o, PB3o, PB4o As Single

'   j        -   mean gas volume counter
Dim j As Integer

'   MeanVol -    set of variables used in the calculation of the mean
volume
'                of gas present in each chamber
Dim MeanVol As Piston
```

```
'   t          -   time in seconds
Dim t As Single

'   TimeInterval     -   determines how often the view is updated
Dim TimeInterval As Single

'   m,n,u,v -    counters used for the calculation of the area
'                inside the PV curve
Dim m, n, u, v As Integer

'   p,w        -   counters used for the calculation of power output
'                  from engines 1 & 3
Dim p, w As Integer

'   Tick1,Tick3,TickPV  -   counters used for printing out the power
'                           generated from engines 1 & 3
Dim Tick1, Tick3, TickPV As Integer

'   Number_of_cycles     -   counter used to average the power from x
cycles
Dim Number_of_cycles As Integer

'   PP*        -   intermediate variables used in the calculation of
PA,PB,PC,PD
Dim PP1, PP2, PP3, PP4 As Single

'   P1_,P2_,P3_ -   pressure of liquid in column
Dim P1A, P1B, P1C, P1D As Single
Dim P2A, P2B, P2C, P2D As Single
Dim P3A, P3B, P3C, P3D As Single

'   i_1,i_3,i_5 -   direction of liquid in column
Dim i_1, i_3, i_5 As Single

'   X***       -   intermediate variables used in the calculation of
P1_,P2_,P3_
Dim X11, X12, X13, X14, X15, X21 As Single
Dim X16, X17, X18, X19, X110, X22 As Single
Dim X111, X112, X113, X114, X115, X23 As Single
Dim X116, X117, X118, X119, X120, X24 As Single

'   Lt1,Lt2 -    lengths of the tuning pipes
Dim Lt1, Lt2 As Single

'   a***       -   liquid acceleration terms for each column of water
Dim a12a, a23a, a34a, a41a As Single
Dim a12b, a23b, a34b, a41b As Single
Dim a12c, a23c, a34c, a41c As Single

'   vl***o     -   liquid velocity buffer
Dim vl12ao, vl23ao, vl34ao, vl41ao As Single
Dim vl12bo, vl23bo, vl34bo, vl41bo As Single
Dim vl12co, vl23co, vl34co, vl41co As Single

'   Chamber,Up,Down -   counters used in the calculation of the mean gas
volume
Dim Chamber As Integer
Dim Up(12), Down(12) As Integer
Dim SumVol(12), sum As Single

'              -   Variables used to calculate the power output from
engines 1 & 3
Dim PosArea, NegArea, AvePower1 As Single
Dim PosArea3, NegArea3, AvePower3 As Single
```

```
Dim PosCurve(1 To 2, 0 To 10000), NegCurve(1 To 2, 0 To 10000), Power(1
To 1000)
Dim PosCurve3(1 To 2, 0 To 10000), NegCurve3(1 To 2, 0 To 10000),
Power3(1 To 1000)
Dim Initial_t1, Initial_t3 As Single
Dim i, ii As Integer

'   neg, sup    -   counters used to check if blowout or negative height
has occurred
Dim neg, sup As Integer

'   PrintRow    -   counter used for printing the output file
Dim PrintRow, PrintRow2 As Integer

'   PrintBuffer -   the number of seconds raw data prints out after
convergence
Dim PrintBuffer As Single

'   View    -   string of the sreen viewed while the program is running
Dim View As String

'   Answer      -   variable used in message boxes
Dim Answer

'           -   variables used to determine convergance
Dim Converged As Boolean
Dim Check1, Check2, Check3, Finished As Single
Dim CycleNumber, CycleNumber_Max As Integer

'           -   variables used to calculate engine period
Dim Hertz, t_initial As Single
Dim CycleNumber_o As Integer

'   Full        -   Shortest column is x% full
Dim Full As Single

'****************************************************************
'****************************************************************
'   Unloading Parameter Data from Passed Array
'****************************************************************
'****************************************************************

Lt1 = Parameters(1) 'Tuning Length 1
Lt2 = Parameters(2) 'Tuning Length 2
LH = Parameters(3) 'Hot Chamber Height
LC = Parameters(4) 'Cold Chamber Height
LA = Parameters(5) 'Ambient Chamber Height
LB = Parameters(6) 'Bounce Chamber Height


'****************************************************************
'   Reading Parameter Data from Spreadsheet "Initial Conditions"
'****************************************************************

Pi = 3.1415927

g = Worksheets("Parameter Values").Range("F8").Value
f = Worksheets("Parameter Values").Range("F9").Value
d = Worksheets("Parameter Values").Range("F7").Value
R = Worksheets("Parameter Values").Range("F11").Value
Dia = Worksheets("Parameter Values").Range("B8").Value
Dia_BC = Worksheets("Parameter Values").Range("B9").Value
Dt = Worksheets("Parameter Values").Range("B4").Value
TH = Worksheets("Parameter Values").Range("F19").Value
```

```
TA = Worksheets("Parameter Values").Range("F21").Value
TC = Worksheets("Parameter Values").Range("F20").Value
TB = Worksheets("Parameter Values").Range("F22").Value
Pmean = Worksheets("Parameter Values").Range("F11").Value
VB = Worksheets("Parameter Values").Range("F4").Value
Damp = Worksheets("Parameter Values").Range("F12").Value
Full = Worksheets("Parameter Values").Range("B16").Value

'*****************************************************************
'   Cross-sectional Area Calculations
'*****************************************************************

Area = Pi * Dia ^ 2 / 4
Area_BC = Pi * Dia_BC ^ 2 / 4


'*****************************************************************
'   Initializing of liquid heights
'*****************************************************************

If LH <= LA And LH <= LC And LH <= LB Then LL = LH * Full
If LA <= LH And LA <= LC And LA <= LB Then LL = LA * Full
If LC <= LA And LC <= LH And LC <= LB Then LL = LC * Full
If LB <= LA And LB <= LH And LB <= LC Then LL = LB * Full

    LA1 = LL
    LH1 = LL
    LB1 = LL + 0.2 ' Infinitesimal offset to start engine oscillation
    LA2 = LL
    LC2 = LL
    LB2 = LL
    LA3 = LL
    LH3 = LL
    LB3 = LL - 0.2 ' Infinitesimal offset to start engine oscillation
    LA4 = LL
    LC4 = LL
    LB4 = LL

'*****************************************************************
'   Initializing of liquid velocities in each column
'*****************************************************************

    vl12a = 0
    vl12b = 0
    vl12c = 0
    vl23a = 0
    vl23b = 0
    vl23c = 0
    vl34a = 0
    vl34b = 0
    vl34c = 0
    vl41a = 0
    vl41b = 0
    vl41c = 0

'*****************************************************************
'   Initializing of gas volumes in each column
'*****************************************************************

    VoA1 = Area * (LA - LA1)
    VoH1 = Area * (LH - LH1)
    VoB1 = Area_BC * (LB - LB1)
    VoA2 = Area * (LA - LA2)
    VoC2 = Area * (LC - LC2)
```

```
    VoB2 = Area_BC * (LB - LB2)
    VoA3 = Area * (LA - LA3)
    VoH3 = Area * (LH - LH3)
    VoB3 = Area_BC * (LB - LB3)
    VoA4 = Area * (LA - LA4)
    VoC4 = Area * (LC - LC4)
    VoB4 = Area_BC * (LB - LB4)

'*****************************************************************
'   Initializing of gas pressures in each column
'*****************************************************************

    PA = Pmean
    PB = Pmean
    PC = Pmean
    PD = Pmean
    PB1 = Pmean
    PB2 = Pmean
    PB3 = Pmean
    PB4 = Pmean

'*****************************************************************
'   INITIALIZATION OF TIME AND OTHER COUNTERS
'*****************************************************************

For j = 1 To 12
    MeanVol.Counter(j) = 1
Next j

For j = 0 To 20
    MeanVol.A1(j) = Area * (LA - LL)
    MeanVol.A2(j) = Area * (LA - LL)
    MeanVol.A3(j) = Area * (LA - LL)
    MeanVol.A4(j) = Area * (LA - LL)
    MeanVol.H1(j) = Area * (LH - LL)
    MeanVol.C2(j) = Area * (LC - LL)
    MeanVol.H3(j) = Area * (LH - LL)
    MeanVol.C4(j) = Area * (LC - LL)
    MeanVol.B1(j) = Area_BC * (LB - LL)
    MeanVol.B2(j) = Area_BC * (LB - LL)
    MeanVol.B3(j) = Area_BC * (LB - LL)
    MeanVol.B4(j) = Area_BC * (LB - LL)
Next j

t = 0
m = 0
n = 0
p = 1
u = 0
v = 0
w = 1
Tick1 = 0
Tick3 = 0
TickPV = 0
Number_of_cycles = Worksheets("Parameter Values").Range("F25").Value
PrintBuffer = Worksheets("Parameter Values").Range("F28").Value
PrintRow = 3
PrintRow2 = 2
Converged = False
Finished = 1000
Engine = 0
CycleNumber = 0
CycleNumber_o = 0
CycleNumber_Max = Worksheets("Parameter Values").Range("F29").Value
```

```
'*****************************************************************
'   SELECT DEFAULT SHEET VIEWED WHILE RUNNING PROGRAM
'*****************************************************************

Sheets("Interface").Select
'Sheets("Virtual Pistons").Select
'Sheets("Power Data").Select
'Sheets("Raw Data").Select

'*****************************************************************
'   BEGINNING OF MAIN LOOP
'*****************************************************************

Do Until t >= (Finished + PrintBuffer)

    TimeInterval = Worksheets("Parameter Values").Range("B5").Value
    View = Worksheets("Parameter Values").Range("A25").Value
'*****************************************************************
'   INITIALIZATION OF GAS VOLUMES
'*****************************************************************

    VoH1o = VoH1
    VoA1o = VoA1
    VoC2o = VoC2
    VoH3o = VoH3
    VoA3o = VoA3
    VoC4o = VoC4
    VoA2o = VoA2
    VoA4o = VoA4
    VoB1o = VoB1
    VoB2o = VoB2
    VoB3o = VoB3
    VoB4o = VoB4

'*****************************************************************
'   INITIALIZATION OF GAS PRESSURES
'*****************************************************************

    PAo = PA
    PBo = PB
    PCo = PC
    PDo = PD
    PB1o = PB1
    PB2o = PB2
    PB3o = PB3
    PB4o = PB4

'*****************************************************************
'   CALCULATION OF GAS VOLUMES
'*****************************************************************

    VoA1 = Area * (LA - LA1)
    VoH1 = Area * (LH - LH1)
    VoB1 = Area_BC * (LB - LB1)
    VoA2 = Area * (LA - LA2)
    VoC2 = Area * (LC - LC2)
    VoB2 = Area_BC * (LB - LB2)
    VoA3 = Area * (LA - LA3)
    VoH3 = Area * (LH - LH3)
    VoB3 = Area_BC * (LB - LB3)
    VoA4 = Area * (LA - LA4)
    VoC4 = Area * (LC - LC4)
    VoB4 = Area_BC * (LB - LB4)
```

```
'*****************************************************************
'   CALCULATION OF GAS PRESSURES VIA THE IDEAL GAS LAW
'*****************************************************************

    PP1 = Pmean * (MeanVol.H1(0) / TH + 2 * VB / (TH + TA) +
MeanVol.A1(0) / TA)
    PP3 = Pmean * (MeanVol.H3(0) / TH + 2 * VB / (TH + TA) +
MeanVol.A3(0) / TA)
    PA = PP1 / (VoH1 / TH + 2 * VB / (TH + TA) + VoA1 / TA)
    PC = PP3 / (VoH3 / TH + 2 * VB / (TH + TA) + VoA3 / TA)
    PP2 = Pmean * (MeanVol.A2(0) / TA + 2 * VB / (TA + TC) +
MeanVol.C2(0) / TC)
    PP4 = Pmean * (MeanVol.A4(0) / TA + 2 * VB / (TA + TC) +
MeanVol.C4(0) / TC)
    PB = PP2 / (VoA2 / TA + 2 * VB / (TA + TC) + VoC2 / TC)
    PD = PP4 / (VoA4 / TA + 2 * VB / (TA + TC) + VoC4 / TC)
    PB1 = Pmean * MeanVol.B1(0) / VoB1
    PB2 = Pmean * MeanVol.B2(0) / VoB2
    PB3 = Pmean * MeanVol.B3(0) / VoB3
    PB4 = Pmean * MeanVol.B4(0) / VoB4

'*****************************************************************
'   CALCULATION OF LIQUID PRESSURES
'*****************************************************************

' P2A
    i_1 = Sgn(vl12a)
    i_3 = Sgn(vl12b)
    i_5 = Sgn(vl12c)

    X11 = PB1 + d * g * LB1 * (2 * (Dia / Dia_BC) * (Dia / Dia_BC) + 1)
    X12 = (PA - PA * Lt1 / (LH1 + Lt1)) * (LB1 / LH1) * (Dia / Dia_BC) *
(Dia / Dia_BC)
    X13 = (PB - PB * Lt2 / (LA2 + Lt2)) * (LB1 / LA2) * (Dia / Dia_BC) *
(Dia / Dia_BC)
    X14 = (d * g * LB1 * Lt1 / (LH1 + Lt1) + d * g * LB1 * Lt2 / (LA2 +
Lt2)) * (Dia / Dia_BC) * (Dia / Dia_BC)
    X15 = 1 + LB1 / (LH1 + Lt1) * (Dia / Dia_BC) * (Dia / Dia_BC) + LB1
/ (LA2 + Lt2) * (Dia / Dia_BC) * (Dia / Dia_BC)
    X21 = 2 * f * d * LB1 * (-i_3 * vl12b ^ 2 / Dia_BC - i_5 * Dia *
vl12c ^ 2 / (Dia_BC ^ 2) + i_1 * Dia * vl12a ^ 2 / (Dia_BC ^ 2))
    P2A = (X11 + X12 + X13 - X14 + X21) / X15
' P2B
    i_1 = Sgn(vl23a)
    i_3 = Sgn(vl23b)
    i_5 = Sgn(vl23c)

    X16 = PB2 + d * g * LB2 * (2 * (Dia / Dia_BC) * (Dia / Dia_BC) + 1)
    X17 = (PB - PB * Lt1 / (LC2 + Lt1)) * (LB2 / LC2) * (Dia / Dia_BC) *
(Dia / Dia_BC)
    X18 = (PC - PC * Lt2 / (LA3 + Lt2)) * (LB2 / LA3) * (Dia / Dia_BC) *
(Dia / Dia_BC)
    X19 = (d * g * LB2 * Lt1 / (LC2 + Lt1) + d * g * LB2 * Lt2 / (LA3 +
Lt2)) * (Dia / Dia_BC) * (Dia / Dia_BC)
    X110 = 1 + LB2 / (LC2 + Lt1) * (Dia / Dia_BC) * (Dia / Dia_BC) + LB2
/ (LA3 + Lt2) * (Dia / Dia_BC) * (Dia / Dia_BC)
    X22 = 2 * f * d * LB2 * (-i_3 * vl23b ^ 2 / Dia_BC - i_5 * Dia *
vl23c ^ 2 / (Dia_BC ^ 2) + i_1 * Dia * vl23a ^ 2 / (Dia_BC ^ 2))
    P2B = (X16 + X17 + X18 - X19 + X22) / X110
' P2C
    i_1 = Sgn(vl34a)
    i_3 = Sgn(vl34b)
    i_5 = Sgn(vl34c)
```

```
    X111 = PB3 + d * g * LB3 * (2 * (Dia / Dia_BC) * (Dia / Dia_BC) + 1)
    X112 = (PC - PC * Lt1 / (LH3 + Lt1)) * (LB3 / LH3) * (Dia / Dia_BC)
* (Dia / Dia_BC)
    X113 = (PD - PD * Lt2 / (LA4 + Lt2)) * (LB3 / LA4) * (Dia / Dia_BC)
* (Dia / Dia_BC)
    X114 = (d * g * LB3 * Lt1 / (LH3 + Lt1) + d * g * LB3 * Lt2 / (LA4 +
Lt2)) * (Dia / Dia_BC) * (Dia / Dia_BC)
    X115 = 1 + LB3 / (LH3 + Lt1) * (Dia / Dia_BC) * (Dia / Dia_BC) + LB3
/ (LA4 + Lt2) * (Dia / Dia_BC) * (Dia / Dia_BC)
    X23 = 2 * f * d * LB3 * (-i_3 * vl34b ^ 2 / Dia_BC - i_5 * Dia *
vl34c ^ 2 / (Dia_BC ^ 2) + i_1 * Dia * vl34a ^ 2 / (Dia_BC ^ 2))
    P2C = (X111 + X112 + X113 - X114 + X23) / X115
' P2D
    i_1 = Sgn(vl41a)
    i_3 = Sgn(vl41b)
    i_5 = Sgn(vl41c)

    X116 = PB4 + d * g * LB4 * (2 * (Dia / Dia_BC) * (Dia / Dia_BC) + 1)
    X117 = (PD - PD * Lt1 / (LC4 + Lt1)) * (LB4 / LC4) * (Dia / Dia_BC)
* (Dia / Dia_BC)
    X118 = (PA - PA * Lt2 / (LA1 + Lt2)) * (LB4 / LA1) * (Dia / Dia_BC)
* (Dia / Dia_BC)
    X119 = (d * g * LB4 * Lt1 / (LC4 + Lt1) + d * g * LB4 * Lt2 / (LA1 +
Lt2)) * (Dia / Dia_BC) * (Dia / Dia_BC)
    X120 = 1 + LB4 / (LC4 + Lt1) * (Dia / Dia_BC) * (Dia / Dia_BC) + LB4
/ (LA1 + Lt2) * (Dia / Dia_BC) * (Dia / Dia_BC)
    X24 = 2 * f * d * LB4 * (-i_3 * vl41b ^ 2 / Dia_BC - i_5 * Dia *
vl41c ^ 2 / (Dia_BC ^ 2) + i_1 * Dia * vl41a ^ 2 / (Dia_BC ^ 2))
    P2D = (X116 + X117 + X118 - X119 + X24) / X120

    P1A = (PA * Lt1 + g * d * LH1 * Lt1 + P2A * LH1) / (LH1 + Lt1)
    P1B = (PB * Lt1 + g * d * LC2 * Lt1 + P2B * LC2) / (LC2 + Lt1)
    P1C = (PC * Lt1 + g * d * LH3 * Lt1 + P2C * LH3) / (LH3 + Lt1)
    P1D = (PD * Lt1 + g * d * LC4 * Lt1 + P2D * LC4) / (LC4 + Lt1)

'     P3A = (g * d * Lt2 * LA2 + PB * Lt2 + P2A * LA2) / (Lt2 + LA2)
'     P3B = (g * d * Lt2 * LA3 + PC * Lt2 + P2B * LA3) / (Lt2 + LA3)
'     P3C = (g * d * Lt2 * LA4 + PD * Lt2 + P2C * LA4) / (Lt2 + LA4)
'     P3D = (g * d * Lt2 * LA1 + PA * Lt2 + P2D * LA1) / (Lt2 + LA1)

'*****************************************************************
'   CALCULATION OF LIQUID ACCELERATIONS
'*****************************************************************

    If (vl12a <= 0) Then a12a = (PA - P1A) / (d * LH1) + g + 2 * f *
vl12a * vl12a / Dia
    If (vl12a > 0) Then a12a = (PA - P1A) / (d * LH1) + g - 2 * f *
vl12a * vl12a / Dia
    If (vl23a <= 0) Then a23a = (PB - P1B) / (d * LC2) + g + 2 * f *
vl23a * vl23a / Dia
    If (vl23a > 0) Then a23a = (PB - P1B) / (d * LC2) + g - 2 * f *
vl23a * vl23a / Dia
    If (vl34a <= 0) Then a34a = (PC - P1C) / (d * LH3) + g + 2 * f *
vl34a * vl34a / Dia
    If (vl34a > 0) Then a34a = (PC - P1C) / (d * LH3) + g - 2 * f *
vl34a * vl34a / Dia
    If (vl41a <= 0) Then a41a = (PD - P1D) / (d * LC4) + g + 2 * f *
vl41a * vl41a / Dia
    If (vl41a > 0) Then a41a = (PD - P1D) / (d * LC4) + g - 2 * f *
vl41a * vl41a / Dia
    If (vl12b <= 0) Then a12b = (P2A - PB1) / (d * LB1) - g + 2 * f *
vl12b * vl12b / Dia_BC
```

```
    If (vl12b > 0) Then a12b = (P2A - PB1) / (d * LB1) - g - 2 * f *
vl12b * vl12b / Dia_BC
    If (vl23b <= 0) Then a23b = (P2B - PB2) / (d * LB2) - g + 2 * f *
vl23b * vl23b / Dia_BC
    If (vl23b > 0) Then a23b = (P2B - PB2) / (d * LB2) - g - 2 * f *
vl23b * vl23b / Dia_BC
    If (vl34b <= 0) Then a34b = (P2C - PB3) / (d * LB3) - g + 2 * f *
vl34b * vl34b / Dia_BC
    If (vl34b > 0) Then a34b = (P2C - PB3) / (d * LB3) - g - 2 * f *
vl34b * vl34b / Dia_BC
    If (vl41b <= 0) Then a41b = (P2D - PB4) / (d * LB4) - g + 2 * f *
vl41b * vl41b / Dia_BC
    If (vl41b > 0) Then a41b = (P2D - PB4) / (d * LB4) - g - 2 * f *
vl41b * vl41b / Dia_BC

    a12c = a12a - (Dia_BC / Dia) * (Dia_BC / Dia) * a12b
    a23c = a23a - (Dia_BC / Dia) * (Dia_BC / Dia) * a23b
    a34c = a34a - (Dia_BC / Dia) * (Dia_BC / Dia) * a34b
    a41c = a41a - (Dia_BC / Dia) * (Dia_BC / Dia) * a41b

    t = t + Dt

'*****************************************************************
'   CALCULATION OF VELOCITY BUFFER
'*****************************************************************

    vl12ao = vl12a
    vl12bo = vl12b
    vl12co = vl12c
    vl23ao = vl23a
    vl23bo = vl23b
    vl23co = vl23c
    vl34ao = vl34a
    vl34bo = vl34b
    vl34co = vl34c
    vl41ao = vl41a
    vl41bo = vl41b
    vl41co = vl41c

'*****************************************************************
'   CALCULATION OF LIQUID VELOCITIES
'*****************************************************************

    vl12a = vl12ao + a12a * Dt
    vl12b = vl12bo + a12b * Dt
    vl12c = vl12co + a12c * Dt
    vl23a = vl23ao + a23a * Dt
    vl23b = vl23bo + a23b * Dt
    vl23c = vl23co + a23c * Dt
    vl34a = vl34ao + a34a * Dt
    vl34b = vl34bo + a34b * Dt
    vl34c = vl34co + a34c * Dt
    vl41a = vl41ao + a41a * Dt
    vl41b = vl41bo + a41b * Dt
    vl41c = vl41co + a41c * Dt

'*****************************************************************
'   CALCULATION OF LIQUID POSITIONS IN EACH COLUMN
'*****************************************************************

    LH1 = LH1 - vl12ao * Dt - 0.5 * a12a * Dt * Dt
    LB1 = LB1 + vl12bo * Dt + 0.5 * a12b * Dt * Dt
    LA2 = LA2 + vl12co * Dt + 0.5 * a12c * Dt * Dt
    LC2 = LC2 - vl23ao * Dt - 0.5 * a23a * Dt * Dt
```

```
    LB2 = LB2 + vl23bo * Dt + 0.5 * a23b * Dt * Dt
    LA3 = LA3 + vl23co * Dt + 0.5 * a23c * Dt * Dt
    LH3 = LH3 - vl34ao * Dt - 0.5 * a34a * Dt * Dt
    LB3 = LB3 + vl34bo * Dt + 0.5 * a34b * Dt * Dt
    LA4 = LA4 + vl34co * Dt + 0.5 * a34c * Dt * Dt
    LC4 = LC4 - vl41ao * Dt - 0.5 * a41a * Dt * Dt
    LB4 = LB4 + vl41bo * Dt + 0.5 * a41b * Dt * Dt
    LA1 = LA1 + vl41co * Dt + 0.5 * a41c * Dt * Dt

'*****************************************************************
'   CALCULATION OF MEAN GAS VOLUME IN EACH CHAMBER
'*****************************************************************

'***********************    Ambient Chamber 1    ***********************

Chamber = 1

    If VoA1 > VoA1o Then                '   Down Stroke
        If (Up(Chamber) > 0) Then   '   One cycle complete
            MeanVol.A1(MeanVol.Counter(Chamber)) = SumVol(Chamber) /
(Up(Chamber) + Down(Chamber))
            sum = 0
            For j = 1 To 20
            sum = MeanVol.A1(j) / 20 + sum
            Next
            MeanVol.A1(0) = (sum - MeanVol.A1(0)) * Damp + sum
            MeanVol.Counter(Chamber) = MeanVol.Counter(Chamber) + 1
            If MeanVol.Counter(Chamber) = 21 Then
MeanVol.Counter(Chamber) = 1
            Up(Chamber) = 0
            Down(Chamber) = 0
            SumVol(Chamber) = 0
        End If
        Down(Chamber) = Down(Chamber) + 1
        SumVol(Chamber) = SumVol(Chamber) + VoA1
    Else                                '   Up Stroke
        SumVol(Chamber) = SumVol(Chamber) + VoA1
        Up(Chamber) = Up(Chamber) + 1
    End If

'***********************    Hot Chamber 1    ***********************

Chamber = 2

     If VoH1 > VoH1o Then               '   Down Stroke
        If (Up(Chamber) > 0) Then ' One cycle complete

            '   Calculate frequency of engine
            If CycleNumber = CycleNumber_o + 9 Then
                Hertz = 1 / ((t - t_initial) / 10)
                CycleNumber_o = CycleNumber
                t_initial = t
                Worksheets("Power Data").Range("C7").Value = Hertz
                Worksheets("Parameter Values").Range("B28").Value =
Hertz
            End If
            CycleNumber = CycleNumber + 1
            Worksheets("Convergence").Range("K3").Value = CycleNumber
            Worksheets("Power Data").Range("C9").Value = CycleNumber
            Worksheets("Parameter Values").Range("B29").Value =
CycleNumber
            '
            MeanVol.H1(MeanVol.Counter(Chamber)) = SumVol(Chamber) /
(Up(Chamber) + Down(Chamber))
```

```
            sum = 0
            For j = 1 To 20
            sum = MeanVol.H1(j) / 20 + sum
            Next
            MeanVol.H1(0) = (sum - MeanVol.H1(0)) * Damp + sum
            MeanVol.Counter(Chamber) = MeanVol.Counter(Chamber) + 1
            If MeanVol.Counter(Chamber) = 21 Then
MeanVol.Counter(Chamber) = 1
            Up(Chamber) = 0
            Down(Chamber) = 0
            SumVol(Chamber) = 0
        End If
        Down(Chamber) = Down(Chamber) + 1
        SumVol(Chamber) = SumVol(Chamber) + VoH1
    Else                                '   Up Stroke
        SumVol(Chamber) = SumVol(Chamber) + VoH1
        Up(Chamber) = Up(Chamber) + 1
    End If

'***********************    Bounce Chamber 1    **********************

Chamber = 3

     If VoB1 > VoB1o Then              '   Down Stroke
        If (Up(Chamber) > 0) Then ' One cycle complete
            MeanVol.B1(MeanVol.Counter(Chamber)) = SumVol(Chamber) /
(Up(Chamber) + Down(Chamber))
            sum = 0
            For j = 1 To 20
            sum = MeanVol.B1(j) / 20 + sum
            Next
            MeanVol.B1(0) = (sum - MeanVol.B1(0)) * Damp + sum
            MeanVol.Counter(Chamber) = MeanVol.Counter(Chamber) + 1
            If MeanVol.Counter(Chamber) = 21 Then
MeanVol.Counter(Chamber) = 1
            Up(Chamber) = 0
            Down(Chamber) = 0
            SumVol(Chamber) = 0
        End If
        Down(Chamber) = Down(Chamber) + 1
        SumVol(Chamber) = SumVol(Chamber) + VoB1
    Else                                '   Up Stroke
        SumVol(Chamber) = SumVol(Chamber) + VoB1
        Up(Chamber) = Up(Chamber) + 1
    End If

'***********************    Ambient Chamber 2    **********************

Chamber = 4

     If VoA2 > VoA2o Then              '   Down Stroke
        If (Up(Chamber) > 0) Then ' One cycle complete
            MeanVol.A2(MeanVol.Counter(Chamber)) = SumVol(Chamber) /
(Up(Chamber) + Down(Chamber))
            sum = 0
            For j = 1 To 20
            sum = MeanVol.A2(j) / 20 + sum
            Next
            MeanVol.A2(0) = (sum - MeanVol.A2(0)) * Damp + sum
            MeanVol.Counter(Chamber) = MeanVol.Counter(Chamber) + 1
            If MeanVol.Counter(Chamber) = 21 Then
MeanVol.Counter(Chamber) = 1
            Up(Chamber) = 0
            Down(Chamber) = 0
```

```
            SumVol(Chamber) = 0
        End If
        Down(Chamber) = Down(Chamber) + 1
        SumVol(Chamber) = SumVol(Chamber) + VoA2
    Else                                '   Up Stroke
        SumVol(Chamber) = SumVol(Chamber) + VoA2
        Up(Chamber) = Up(Chamber) + 1
    End If

'***********************    Cold Chamber 2    ***********************

Chamber = 5

     If VoC2 > VoC2o Then            '    Down Stroke
        If (Up(Chamber) > 0) Then ' One cycle complete
            MeanVol.C2(MeanVol.Counter(Chamber)) = SumVol(Chamber) /
(Up(Chamber) + Down(Chamber))
            sum = 0
            For j = 1 To 20
            sum = MeanVol.C2(j) / 20 + sum
            Next
            MeanVol.C2(0) = (sum - MeanVol.C2(0)) * Damp + sum
            MeanVol.Counter(Chamber) = MeanVol.Counter(Chamber) + 1
            If MeanVol.Counter(Chamber) = 21 Then
MeanVol.Counter(Chamber) = 1
            Up(Chamber) = 0
            Down(Chamber) = 0
            SumVol(Chamber) = 0
        End If
        Down(Chamber) = Down(Chamber) + 1
        SumVol(Chamber) = SumVol(Chamber) + VoC2
    Else                                '   Up Stroke
        SumVol(Chamber) = SumVol(Chamber) + VoC2
        Up(Chamber) = Up(Chamber) + 1
    End If

'***********************    Bounce Chamber 2    ***********************

Chamber = 6

     If VoB2 > VoB2o Then            '    Down Stroke
        If (Up(Chamber) > 0) Then ' One cycle complete
            MeanVol.B2(MeanVol.Counter(Chamber)) = SumVol(Chamber) /
(Up(Chamber) + Down(Chamber))
            sum = 0
            For j = 1 To 20
            sum = MeanVol.B2(j) / 20 + sum
            Next
            MeanVol.B2(0) = (sum - MeanVol.B2(0)) * Damp + sum
            MeanVol.Counter(Chamber) = MeanVol.Counter(Chamber) + 1
            If MeanVol.Counter(Chamber) = 21 Then
MeanVol.Counter(Chamber) = 1
            Up(Chamber) = 0
            Down(Chamber) = 0
            SumVol(Chamber) = 0
        End If
        Down(Chamber) = Down(Chamber) + 1
        SumVol(Chamber) = SumVol(Chamber) + VoB2
    Else                                '   Up Stroke
        SumVol(Chamber) = SumVol(Chamber) + VoB2
        Up(Chamber) = Up(Chamber) + 1
    End If

'***********************    Ambient Chamber 3    ***********************
```

```
Chamber = 7

     If VoA3 > VoA3o Then          '     Down Stroke
        If (Up(Chamber) > 0) Then ' One cycle complete
            MeanVol.A3(MeanVol.Counter(Chamber)) = SumVol(Chamber) /
(Up(Chamber) + Down(Chamber))
            sum = 0
            For j = 1 To 20
            sum = MeanVol.A3(j) / 20 + sum
            Next
            MeanVol.A3(0) = (sum - MeanVol.A3(0)) * Damp + sum
            MeanVol.Counter(Chamber) = MeanVol.Counter(Chamber) + 1
            If MeanVol.Counter(Chamber) = 21 Then
MeanVol.Counter(Chamber) = 1
            Up(Chamber) = 0
            Down(Chamber) = 0
            SumVol(Chamber) = 0
        End If
        Down(Chamber) = Down(Chamber) + 1
        SumVol(Chamber) = SumVol(Chamber) + VoA3
    Else                              '   Up Stroke
        SumVol(Chamber) = SumVol(Chamber) + VoA3
        Up(Chamber) = Up(Chamber) + 1
    End If

'***********************     Hot Chamber 3     **********************

Chamber = 8

     If VoH3 > VoH3o Then          '     Down Stroke
        If (Up(Chamber) > 0) Then ' One cycle complete
            MeanVol.H3(MeanVol.Counter(Chamber)) = SumVol(Chamber) /
(Up(Chamber) + Down(Chamber))
            sum = 0
            For j = 1 To 20
            sum = MeanVol.H3(j) / 20 + sum
            Next
            MeanVol.H3(0) = (sum - MeanVol.H3(0)) * Damp + sum
            MeanVol.Counter(Chamber) = MeanVol.Counter(Chamber) + 1
            If MeanVol.Counter(Chamber) = 21 Then
MeanVol.Counter(Chamber) = 1
            Up(Chamber) = 0
            Down(Chamber) = 0
            SumVol(Chamber) = 0
        End If
        Down(Chamber) = Down(Chamber) + 1
        SumVol(Chamber) = SumVol(Chamber) + VoH3
    Else                              '   Up Stroke
        SumVol(Chamber) = SumVol(Chamber) + VoH3
        Up(Chamber) = Up(Chamber) + 1
    End If

'***********************     Bounce Chamber 3     **********************

Chamber = 9

     If VoB3 > VoB3o Then          '     Down Stroke
        If (Up(Chamber) > 0) Then ' One cycle complete
            MeanVol.B3(MeanVol.Counter(Chamber)) = SumVol(Chamber) /
(Up(Chamber) + Down(Chamber))
            sum = 0
            For j = 1 To 20
            sum = MeanVol.B3(j) / 20 + sum
```

```
            Next
            MeanVol.B3(0) = (sum - MeanVol.B3(0)) * Damp + sum
            MeanVol.Counter(Chamber) = MeanVol.Counter(Chamber) + 1
            If MeanVol.Counter(Chamber) = 21 Then
MeanVol.Counter(Chamber) = 1
            Up(Chamber) = 0
            Down(Chamber) = 0
            SumVol(Chamber) = 0
        End If
        Down(Chamber) = Down(Chamber) + 1
        SumVol(Chamber) = SumVol(Chamber) + VoB3
    Else                            '   Up Stroke
        SumVol(Chamber) = SumVol(Chamber) + VoB3
        Up(Chamber) = Up(Chamber) + 1
    End If

'***********************   Ambient Chamber 4   ***********************

Chamber = 10

     If VoA4 > VoA4o Then           '   Down Stroke
        If (Up(Chamber) > 0) Then ' One cycle complete
            MeanVol.A4(MeanVol.Counter(Chamber)) = SumVol(Chamber) /
(Up(Chamber) + Down(Chamber))
            sum = 0
            For j = 1 To 20
            sum = MeanVol.A4(j) / 20 + sum
            Next
            MeanVol.A4(0) = (sum - MeanVol.A4(0)) * Damp + sum
            MeanVol.Counter(Chamber) = MeanVol.Counter(Chamber) + 1
            If MeanVol.Counter(Chamber) = 21 Then
MeanVol.Counter(Chamber) = 1
            Up(Chamber) = 0
            Down(Chamber) = 0
            SumVol(Chamber) = 0
        End If
        Down(Chamber) = Down(Chamber) + 1
        SumVol(Chamber) = SumVol(Chamber) + VoA4
    Else                            '   Up Stroke
        SumVol(Chamber) = SumVol(Chamber) + VoA4
        Up(Chamber) = Up(Chamber) + 1
    End If

'***********************   Cold Chamber 4   ***********************

Chamber = 11

     If VoC4 > VoC4o Then           '   Down Stroke
        If (Up(Chamber) > 0) Then ' One cycle complete
            MeanVol.C4(MeanVol.Counter(Chamber)) = SumVol(Chamber) /
(Up(Chamber) + Down(Chamber))
            sum = 0
            For j = 1 To 20
            sum = MeanVol.C4(j) / 20 + sum
            Next
            MeanVol.C4(0) = (sum - MeanVol.C4(0)) * Damp + sum
            MeanVol.Counter(Chamber) = MeanVol.Counter(Chamber) + 1
            If MeanVol.Counter(Chamber) = 21 Then
MeanVol.Counter(Chamber) = 1
            Up(Chamber) = 0
            Down(Chamber) = 0
            SumVol(Chamber) = 0
        End If
        Down(Chamber) = Down(Chamber) + 1
```

```
        SumVol(Chamber) = SumVol(Chamber) + VoC4
    Else                                '   Up Stroke
        SumVol(Chamber) = SumVol(Chamber) + VoC4
        Up(Chamber) = Up(Chamber) + 1
    End If

'***********************    Bounce Chamber 4    ***********************

Chamber = 12

     If VoB4 > VoB4o Then           '    Down Stroke
        If (Up(Chamber) > 0) Then ' One cycle complete
            MeanVol.B4(MeanVol.Counter(Chamber)) = SumVol(Chamber) /
(Up(Chamber) + Down(Chamber))
            sum = 0
            For j = 1 To 20
            sum = MeanVol.B4(j) / 20 + sum
            Next
            MeanVol.B4(0) = (sum - MeanVol.B4(0)) * Damp + sum
            MeanVol.Counter(Chamber) = MeanVol.Counter(Chamber) + 1
            If MeanVol.Counter(Chamber) = 21 Then
MeanVol.Counter(Chamber) = 1
            Up(Chamber) = 0
            Down(Chamber) = 0
            SumVol(Chamber) = 0
        End If
        Down(Chamber) = Down(Chamber) + 1
        SumVol(Chamber) = SumVol(Chamber) + VoB4
    Else                                '   Up Stroke
        SumVol(Chamber) = SumVol(Chamber) + VoB4
        Up(Chamber) = Up(Chamber) + 1
    End If
'****************************************************************

'****************************************************************
'   CALCULATION OF POWER PER CYCLE
'****************************************************************

    If ((VoH1 + VoA1) > (VoH1o + VoA1o)) Then       ' Positive Curve
        If (n > 0) Then
            PosArea = PV_Area(PosCurve, m)
            NegArea = PV_Area(NegCurve, n)
            Power(p) = (NegArea - PosArea) / (t - Initial_t1) _
                        / 10000000#
            Initial_t1 = t
            m = 0
            n = 0
            p = p + 1
            If p = Number_of_cycles + 1 Then
                AvePower1 = 0
                For ii = 1 To Number_of_cycles
                    AvePower1 = Power(ii) / Number_of_cycles + AvePower1
                Next ii
                'POWERPOWERPOWERPOWERPOWERPOWERPOWERPOWERPOWERPOWERPOWER
                DialogSheets("Virtual Pistons").EditBoxes("Edit Box 53")
_
                .Text = AvePower1
                Worksheets("Power Data").Cells(6, 8).Value = AvePower1
                Worksheets("Power Data").Cells(Tick1 + 7, 8).Value =
AvePower1
                Worksheets("Power Data").Cells(Tick1 + 7, 7).Value = t
                Tick1 = Tick1 + 1
                'POWERPOWERPOWERPOWERPOWERPOWERPOWERPOWERPOWERPOWERPOWER
                p = 1
```

```
            End If
        End If
        PosCurve(1, m) = PA
        PosCurve(2, m) = VoH1 + VoA1
        m = m + 1
    End If
    If ((VoH1 + VoA1) < (VoH1o + VoA1o)) Then       '   Negative Curve
        NegCurve(1, n) = PA
        NegCurve(2, n) = VoH1o + VoA1o
        n = n + 1
    End If

    '   Engine 3 Power

    If ((VoH3 + VoA3) > (VoH3o + VoA3o)) Then          '    Positvie
Curve
        If (u > 0) Then
            PosArea3 = PV_Area(PosCurve3, v)
            NegArea3 = PV_Area(NegCurve3, u)
            Power3(w) = (NegArea3 - PosArea3) / (t - Initial_t3) _
                        / 10000000#
            Initial_t3 = t
            v = 0
            u = 0
            w = w + 1
            If w = Number_of_cycles + 1 Then
                '
                AvePower3 = 0
                For ii = 1 To Number_of_cycles
                    AvePower3 = Power3(ii) / Number_of_cycles +
AvePower3
                Next ii
                'POWERPOWERPOWERPOWERPOWERPOWERPOWERPOWERPOWERPOWER
                DialogSheets("Virtual Pistons").EditBoxes("Edit Box 54")
_
                .Text = AvePower3
                Worksheets("Power Data").Cells(6, 10).Value = AvePower3
                Worksheets("Power Data").Cells(Tick3 + 7, 10).Value =
AvePower3
                Worksheets("Power Data").Cells(Tick3 + 7, 9).Value = t
                Tick3 = Tick3 + 1
                'POWERPOWERPOWERPOWERPOWERPOWERPOWERPOWERPOWERPOWER
                w = 1
            End If
        End If
        PosCurve3(1, v) = PC
        PosCurve3(2, v) = VoH3 + VoA3
        v = v + 1
    End If
    If ((VoH3 + VoA3) < (VoH3o + VoA3o)) Then          '   Negative
Curve
        NegCurve3(1, u) = PC
        NegCurve3(2, u) = VoH3o + VoA3o
        u = u + 1
    End If

'****************************************************************
'   NEGATIVE HEIGHT CHECK
'****************************************************************

      If (LH1 < 0) Then neg = neg + 1
      If (LB1 < 0) Then neg = neg + 1
      If (LA2 < 0) Then neg = neg + 1
      If (LC2 < 0) Then neg = neg + 1
```

```
      If (LB2 < 0) Then neg = neg + 1
      If (LA3 < 0) Then neg = neg + 1
      If (LH3 < 0) Then neg = neg + 1
      If (LB3 < 0) Then neg = neg + 1
      If (LA4 < 0) Then neg = neg + 1
      If (LC4 < 0) Then neg = neg + 1
      If (LB4 < 0) Then neg = neg + 1
      If (LA1 < 0) Then neg = neg + 1
      If (neg >= 1) Then
        Engine = -1
        Exit Do
      End If

'****************************************************************
'   BLOWOUT CHECK
'****************************************************************

      If (LH1 > LH) Then sup = sup + 1
      If (LB1 > LB) Then sup = sup + 1
      If (LA2 > LA) Then sup = sup + 1
      If (LC2 > LC) Then sup = sup + 1
      If (LB2 > LB) Then sup = sup + 1
      If (LA3 > LA) Then sup = sup + 1
      If (LH3 > LH) Then sup = sup + 1
      If (LB3 > LB) Then sup = sup + 1
      If (LA4 > LA) Then sup = sup + 1
      If (LC4 > LC) Then sup = sup + 1
      If (LB4 > LB) Then sup = sup + 1
      If (LA1 > LA) Then sup = sup + 1
      If (sup >= 1) Then
        Engine = 1
        Exit Do
      End If

'****************************************************************
'   UPDATE CURRENT VIEW
'****************************************************************

If (t * 1000 Mod TimeInterval = 0) Then

    If View = "Power Data" Or View = "Virtual Pistons" Or View =
"Virtual Pistons II" Then
    Worksheets("Power Data").Cells(3, 1).Value = LA1
    Worksheets("Power Data").Cells(3, 2).Value = LH1
    Worksheets("Power Data").Cells(3, 3).Value = LB1
    Worksheets("Power Data").Cells(3, 4).Value = LA2
    Worksheets("Power Data").Cells(3, 5).Value = LC2
    Worksheets("Power Data").Cells(3, 6).Value = LB2
    Worksheets("Power Data").Cells(3, 7).Value = LA3
    Worksheets("Power Data").Cells(3, 8).Value = LH3
    Worksheets("Power Data").Cells(3, 9).Value = LB3
    Worksheets("Power Data").Cells(3, 10).Value = LA4
    Worksheets("Power Data").Cells(3, 11).Value = LC4
    Worksheets("Power Data").Cells(3, 12).Value = LB4
    Worksheets("Power Data").Cells(5, 3).Value = t
    Worksheets("Power Data").Cells(TickPV + 8, 1).Value = PA
    Worksheets("Power Data").Cells(TickPV + 8, 2).Value = VoH1 + VoA1
    TickPV = TickPV + 1
        If TickPV > 30 Then TickPV = 0
    End If

    If View = "Virtual Pistons" Then
    DialogSheets("Virtual Pistons").EditBoxes("Edit Box 19").Text = t
    End If
```

```
End If

'*****************************************************************
'   PRINT OUT RESULTS
'*****************************************************************


    i = 50

    If Converged And (t * 1000 Mod i = 0) Then

    PrintRow = PrintRow + 1
    Worksheets("Raw Data").Cells(PrintRow, 1).Value = t
    Worksheets("Raw Data").Cells(PrintRow, 2).Value = LA1
    Worksheets("Raw Data").Cells(PrintRow, 3).Value = LH1
    Worksheets("Raw Data").Cells(PrintRow, 4).Value = LB1
    Worksheets("Raw Data").Cells(PrintRow, 5).Value = LA2
    Worksheets("Raw Data").Cells(PrintRow, 6).Value = LC2
    Worksheets("Raw Data").Cells(PrintRow, 7).Value = LB2
    Worksheets("Raw Data").Cells(PrintRow, 8).Value = LA3
    Worksheets("Raw Data").Cells(PrintRow, 9).Value = LH3
    Worksheets("Raw Data").Cells(PrintRow, 10).Value = LB3
    Worksheets("Raw Data").Cells(PrintRow, 11).Value = LA4
    Worksheets("Raw Data").Cells(PrintRow, 12).Value = LC4
    Worksheets("Raw Data").Cells(PrintRow, 13).Value = LB4
    Worksheets("Raw Data").Cells(PrintRow, 14).Value = MeanVol.A1(0)
    Worksheets("Raw Data").Cells(PrintRow, 15).Value = MeanVol.H1(0)
    Worksheets("Raw Data").Cells(PrintRow, 16).Value = MeanVol.B1(0)
    Worksheets("Raw Data").Cells(PrintRow, 17).Value = MeanVol.A2(0)
    Worksheets("Raw Data").Cells(PrintRow, 18).Value = MeanVol.C2(0)
    Worksheets("Raw Data").Cells(PrintRow, 19).Value = MeanVol.B2(0)
    Worksheets("Raw Data").Cells(PrintRow, 20).Value = MeanVol.A3(0)
    Worksheets("Raw Data").Cells(PrintRow, 21).Value = MeanVol.H3(0)
    Worksheets("Raw Data").Cells(PrintRow, 22).Value = MeanVol.B3(0)
    Worksheets("Raw Data").Cells(PrintRow, 23).Value = MeanVol.A4(0)
    Worksheets("Raw Data").Cells(PrintRow, 24).Value = MeanVol.C4(0)
    Worksheets("Raw Data").Cells(PrintRow, 25).Value = MeanVol.B4(0)

    Worksheets("Raw Data").Cells(PrintRow, 26).Value = PA
    Worksheets("Raw Data").Cells(PrintRow, 27).Value = VoA1 + VoH1
    Worksheets("Raw Data").Cells(PrintRow, 28).Value = PB
    Worksheets("Raw Data").Cells(PrintRow, 29).Value = VoA2 + VoC2
    Worksheets("Raw Data").Cells(PrintRow, 30).Value = PA
    Worksheets("Raw Data").Cells(PrintRow, 31).Value = VoA3 + VoH3
    Worksheets("Raw Data").Cells(PrintRow, 32).Value = PB
    Worksheets("Raw Data").Cells(PrintRow, 33).Value = VoA4 + VoC4

    Worksheets("Raw Data").Cells(PrintRow, 34).Value = Power(p)
    Worksheets("Raw Data").Cells(PrintRow, 35).Value = Power3(w)



    End If

'*****************************************************************
'   CONVERGENCE CHECK
'*****************************************************************

'    i = 100
```

```
'
'
'     If (t * 1000 Mod i = 0) And Converged = False Then
'     PrintRow2 = PrintRow2 + 1
'     Worksheets("Convergence").Cells(PrintRow2, 1).Value = t
'     Worksheets("Convergence").Cells(PrintRow2, 2).Value = MeanVol.A1(0)
'     Worksheets("Convergence").Cells(PrintRow2, 3).Value = MeanVol.H1(0)
'     Worksheets("Convergence").Cells(PrintRow2, 4).Value = MeanVol.B1(0)
'     If PrintRow2 >= 202 Then PrintRow2 = 2
'     Check1 = Worksheets("Convergence").Cells(3, 8).Value
'     Check2 = Worksheets("Convergence").Cells(3, 9).Value
'     Check3 = Worksheets("Convergence").Cells(3, 10).Value
'         If CycleNumber > 200 And Check1 < 25 And Check2 < 25 And Check3
< 25 Or t > Finished Then
'             Converged = True
'             Finished = t
'             Beep
'             Beep
 '            Beep
'             Sheets("Raw Data").Select
'         End If
'     End If

    If CycleNumber > 1000 Then
            Converged = True
            Finished = t
            Sheets("Raw Data").Select
    End If

'*****************************************************************
'   REPEAT LOOP
'*****************************************************************

Loop


'*****************************************************************
'   Quit computation and return results
'*****************************************************************

    If Engine = 0 Then

    Results(1) = Finished

    For i = 4 To PrintBuffer * 10 + 4
        Results(2) = Worksheets("Raw Data").Cells(i, 46).Value _
                     / (PrintBuffer * 10) + Results(2)
        Results(3) = Worksheets("Raw Data").Cells(i, 47).Value _
                     / (PrintBuffer * 10) + Results(3)
    Next i

    End If

Exit Function

ChangeViews:
    If Err = 18 Then
        Answer = MsgBox("Would you like to view a different screen or
change refresh rate?", vbYesNoCancel)
        If Answer = vbCancel Then  'Continue at the point of
interruption
        Resume
        ElseIf Answer = vbNo Then  'End Program
```

```
            Exit Function
        Else    'Enable dialog box to change to view
            ViewChanger
            Resume
        End If
    Else  'Handle other errors
        MsgBox Error(Err)
        Exit Function
    End If


End Function




Function PV_Area(Curve, num)

    Dim j As Integer

For j = 1 To num
    PV_Area = PV_Area + Abs(((Curve(1, j - 1) + Curve(1, j)) / 2) * _
            (Curve(2, j - 1) - Curve(2, j)))
Next j
End Function


Sub ViewChanger()

    DialogSheets("View Changer").Show

        If DialogSheets("View Changer").OptionButtons(1).Value = xlOn
Then
            Sheets("Virtual Pistons").Select
            Worksheets("Parameter Values").Range("A25").Value = "Virtual
Pistons"

        ElseIf DialogSheets("View Changer").OptionButtons(2).Value =
xlOn Then
            Sheets("Virtual Pistons II").Select
            Worksheets("Parameter Values").Range("A25").Value = "Virtual
Pistons II"

        ElseIf DialogSheets("View Changer").OptionButtons(3).Value =
xlOn Then
            Sheets("Power Data").Select
            Worksheets("Parameter Values").Range("A25").Value = "Power
Data"

        ElseIf DialogSheets("View Changer").OptionButtons(4).Value =
xlOn Then
            Sheets("Parameter Values").Select
            Worksheets("Parameter Values").Range("A25").Value =
"Parameter Values"

        ElseIf DialogSheets("View Changer").OptionButtons(5).Value =
xlOn Then
            Sheets("Convergence").Select
            Worksheets("Parameter Values").Range("A25").Value =
"Convergence"
```

```
        ElseIf DialogSheets("View Changer").OptionButtons(6).Value =
xlOn Then
                Sheets("Parameters").Select
                Worksheets("Parameter Values").Range("A25").Value =
"Parameters"

        ElseIf DialogSheets("View Changer").OptionButtons(7).Value =
xlOn Then
                Sheets("Interface").Select
                Worksheets("Parameter Values").Range("A25").Value =
"Interface"

        End If

        If DialogSheets("View Changer").OptionButtons(8).Value = xlOn
Then
                Worksheets("Parameter Values").Range("B5").Value = 1

        ElseIf DialogSheets("View Changer").OptionButtons(9).Value =
xlOn Then
                Worksheets("Parameter Values").Range("B5").Value = 10

        ElseIf DialogSheets("View Changer").OptionButtons(10).Value =
xlOn Then
                Worksheets("Parameter Values").Range("B5").Value = 100

        ElseIf DialogSheets("View Changer").OptionButtons(11).Value =
xlOn Then
                Worksheets("Parameter Values").Range("B5").Value = 1000

        ElseIf DialogSheets("View Changer").OptionButtons(12).Value =
xlOn Then
                Worksheets("Parameter Values").Range("B5").Value = _
                DialogSheets("View Changer").EditBoxes("Edit Box 21").Text

        End If

End Sub
```

APPENDIX C: FORTRAN OPTIMIZATION CODE

```
C ABSTRACT:
C   Simulated annealing is a global optimization method that
distinguishes
C   between different local optima. Starting from an initial point, the
C   algorithm takes a step and the function is evaluated. When
minimizing a
C   function, any downhill step is accepted and the process repeats from
this
C   new point. An uphill step may be accepted. Thus, it can escape from
local
C   optima. This uphill decision is made by the Metropolis criteria. As
the
C   optimization process proceeds, the length of the steps decline and
the
C   algorithm closes in on the global optimum. Since the algorithm makes
very
C   few assumptions regarding the function to be optimized, it is quite
C   robust with respect to non-quadratic surfaces. The degree of
robustness
C   can be adjusted by the user. In fact, simulated annealing can be
used as
C   a local optimizer for difficult functions.
C
C   This implementation of simulated annealing was used in "Global
Optimization
C   of Statistical Functions with Simulated Annealing," Goffe, Ferrier
and
C   Rogers, Journal of Econometrics, vol. 60, no. 1/2, Jan./Feb. 1994,
pp.
C   65-100. Briefly, we found it competitive, if not superior, to
multiple
C   restarts of conventional optimization routines for difficult
optimization
C   problems.
C
C   For more information on this routine, contact its author:
C   Bill Goffe, bgoffe@whale.st.usm.edu
C
      PROGRAM SIMANN
C  This file is an example of the Corana et al. simulated annealing
C  algorithm for multimodal and robust optimization as implemented
C  and modified by Goffe, Ferrier and Rogers. Counting the above line
C  ABSTRACT as 1, the routine itself (SA), with its supplementary
C  routines, is on lines 232-990. A multimodal example from Judge et al.
C  (FCN) is on lines 150-231. The rest of this file (lines 1-149) is a
C  driver routine with values appropriate for the Judge example. Thus,
this
C  example is ready to run.
C
C  To understand the algorithm, the documentation for SA on lines 236-
C  484 should be read along with the parts of the paper that describe
C  simulated annealing. Then the following lines will then aid the user
C  in becomming proficient with this implementation of simulated
C  annealing.
C
C  Learning to use SA:
C      Use the sample function from Judge with the following suggestions
```

```
C  to get a feel for how SA works. When you've done this, you should be
C  ready to use it on most any function with a fair amount of expertise.
C     1. Run the program as is to make sure it runs okay. Take a look at
C        the intermediate output and see how it optimizes as temperature
C        (T) falls. Notice how the optimal point is reached and how
C        falling T reduces VM.
C     2. Look through the documentation to SA so the following makes a
C        bit of sense. In line with the paper, it shouldn't be that hard
C        to figure out. The core of the algorithm is described on pp. 68-
70
C        and on pp. 94-95. Also see Corana et al. pp. 264-9.
C     3. To see how it selects points and makes decisions about uphill
C      and downhill moves, set IPRINT = 3 (very detailed intermediate
C        output) and MAXEVL = 100 (only 100 function evaluations to limit
C        output).
C     4. To see the importance of different temperatures, try starting
C        with a very low one (say T = 10E-5). You'll see (i) it never
C        escapes from the local optima (in annealing terminology, it
C        quenches) & (ii) the step length (VM) will be quite small. This
C        is a key part of the algorithm: as temperature (T) falls, step
C        length does too. In a minor point here, note how VM is quickly
C        reset from its initial value. Thus, the input VM is not very
C        important. This is all the more reason to examine VM once the
C        algorithm is underway.
C     5. To see the effect of different parameters and their effect on
C        the speed of the algorithm, try RT = .95 & RT = .1. Notice the
C        vastly different speed for optimization. Also try NT = 20. Note
C        that this sample function is quite easy to optimize, so it will
C        tolerate big changes in these parameters. RT and NT are the
C        parameters one should adjust to modify the runtime of the
C        algorithm and its robustness.
C     6. Try constraining the algorithm with either LB or UB.

      PARAMETER (N = 8, NEPS = 4)

      DOUBLE PRECISION  LB(N), UB(N), X(N), XOPT(N), C(N), VM(N),
     1                  FSTAR(NEPS), XP(N), T, EPS, RT, FOPT

      INTEGER  NACP(N), NS, NT, NFCNEV, IER, ISEED1, ISEED2,
     1         MAXEVL, IPRINT, NACC, NOBDS

      LOGICAL  MAX

      EXTERNAL FCN

C  Set underflows to zero on IBM mainframes.
C     CALL XUFLOW(0)

      OPEN (7, FILE='SIMDAT.DAT', STATUS='UNKNOWN')


C  Set input parameters.
      MAX = .TRUE.
      EPS = 1.0D-6
      RT = .85
      ISEED1 = 1
      ISEED2 = 2
      NS = 20
      NT = 5
      MAXEVL = 50
      IPRINT = 3


      LB(1) = .1
```

```
      UB(1) = 10.
      LB(2) = 10.
      UB(2) = 1000.
      DO I= 3 , 6
           LB(I)=15.
           UB(I)=200.
      End Do
      LB(7) = 10.
      UB(7) = 500.
      LB(8) = .5
      UB(8) = .95
      Do I= 1, N
           C(I) = 2.0
      End Do

C  Note start at local, but not global, optima of the Judge function.
      X(1) =  .5
      X(2) = 100.
      X(3) = 30.
      X(4) = 30.
      X(5) = 30.
      X(6) = 30.
      X(7) = 100.
      X(8) = .5

C  Set input values of the input/output parameters.
      T = 5.0
      DO 20, I = 1, N
         VM(I) = 1.0
 20   CONTINUE

      WRITE(7,1000) N, MAX, T, RT, EPS, NS, NT, NEPS, MAXEVL, IPRINT,
     1              ISEED1, ISEED2

      CALL PRTVEC(X,N,'STARTING VALUES')
      CALL PRTVEC(VM,N,'INITIAL STEP LENGTH')
      CALL PRTVEC(LB,N,'LOWER BOUND')
      CALL PRTVEC(UB,N,'UPPER BOUND')
      CALL PRTVEC(C,N,'C VECTOR')
      WRITE(7,'(/,''  ****   END OF DRIVER ROUTINE OUTPUT    ****''
     1          /,''  ****   BEFORE CALL TO SA.              ****'')')

      CALL SA(N,X,MAX,RT,EPS,NS,NT,NEPS,MAXEVL,LB,UB,C,IPRINT,ISEED1,
     1        ISEED2,T,VM,XOPT,FOPT,NACC,NFCNEV,NOBDS,IER,
     2        FSTAR,XP,NACP)

      WRITE(7,'(/,''  ****   RESULTS AFTER SA   ****   '')')
      CALL PRTVEC(XOPT,N,'SOLUTION')
      CALL PRTVEC(VM,N,'FINAL STEP LENGTH')
      WRITE(7,1001) FOPT, NFCNEV, NACC, NOBDS, T, IER

1000  FORMAT(/,' SIMULATED ANNEALING EXAMPLE',/,
     1       /,' NUMBER OF PARAMETERS: ',I3,'   MAXIMAZATION: ',L5,
     2       /,' INITIAL TEMP: ', G8.2, '   RT: ',G8.2, '   EPS: ',G8.2,
     3       /,' NS: ',I3, '   NT: ',I2, '   NEPS: ',I2,
     4       /,' MAXEVL: ',I10, '   IPRINT: ',I1, '   ISEED1: ',I4,
     5       '   ISEED2: ',I4)
1001  FORMAT(/,' OPTIMAL FUNCTION VALUE: ',G20.13
     1       /,' NUMBER OF FUNCTION EVALUATIONS:     ',I10,
     2       /,' NUMBER OF ACCEPTED EVALUATIONS:     ',I10,
     3       /,' NUMBER OF OUT OF BOUND EVALUATIONS: ',I10,
     4       /,' FINAL TEMP: ', G20.13,'  IER: ', I3)

      STOP
```

```
      END
C ********************************************************************
C ********************************************************************

      SUBROUTINE FCN(Var, X, Engine)

      Double Precision X(8),engine
      Integer Var

C************************************************************
C*****           DECLARATION OF VARIABLES               *****
C*****     All units are expressed in CGS system        *****
C************************************************************

C   Pi     -   Pi (any questions?)
      Real Pi

C   g      -   gravitaional acceleration constant
      Real g

C   f        -   Fanning friction factor
      Real f

C   d        -   density of the working fluid (water)
      Real d

C   R        -   gas constant
      Real R

C   Dia***   -   Diameters for the following pipes:
C            -   Tuning pipes, Ambient, hot, and cold columns
C       _BC  -   Bounce Chamber columns
      Real Dia, DiaBC

C   Area*** -   Cross-sectional area for the following pipes:
C           -   Tuning pipes, Ambient, hot, and cold columns
C       _BC -   Bounce Chamber columns
      Real Area, AreaBC

C   Dt       -   time interval between iterations
      Real Dt

C   L        -   height of the bounce, ambient, hot, or cold columns
      Real LB, LA, LH, LC

C   LL       -   initial height of the water in all of the columns
      Real LL
      Real Full

C   T*       -   temperature of the following columns:
C H - hot
C A - ambient
C C - cold
C B - bounce
      Real TH, TA, TB, TC

C   Pmean    -   mean gas pressure in the gas equalization line
      Real Pmean

C   VB          - volume of the regenerator between the ambient and
C           hot/cold columns
      Real VB

C   Damp     -   damping factor for mean gas volume calculations
```

```
      Real Damp

C   L**      -     height of liquid in the ambient, hot, cold, or bounce
C            chamber
      Real LA1, LH1, LB1, LA2, LC2, LB2, LA3, LH3, LB3, LA4, LC4, LB4

C   vl***    -   liquid velocities in the columns.  see figure 5 p. 15
      Real vl12a, vl12b, vl12c, vl23a, vl23b, vl23c
      Real vl34a, vl34b, vl34c, vl41a, vl41b, vl41c

C   Vo**     -   volume of gas in each column
      Real VoH1, VoH3, VoC2, VoC4, VoB1, VoB2
      Real VoB3, VoB4, VoA1, VoA2, VoA3, VoA4

C   Vo**o    -   initial volume of gas in each column
      Real VoH1o, VoH3o, VoC2o, VoC4o, VoB1o, VoB2o
      Real VoB3o, VoB4o, VoA1o, VoA2o, VoA3o, VoA4o

C   P**      -   pressure of gas in each column
      Real PA, PB, PC, PD, PB1, PB2, PB3, PB4

C   P**o     -   initial pressure of gas in each column
      Real PAo, PBo, PCo, PDo, PB1o, PB2o, PB3o, PB4o

C   j        -   mean gas volume counter
      Integer j

C   MeanVol -   set of variables used in the calculation of the mean
volume
C               of gas present in each chamber

      Integer MVCount(12)
      Real MVA1(21)
      Real MVA2(21)
      Real MVA3(21)
      Real MVA4(21)
      Real MVH1(21)
      Real MVC2(21)
      Real MVH3(21)
      Real MVC4(21)
      Real MVB1(21)
      Real MVB2(21)
      Real MVB3(21)
      Real MVB4(21)

C   t        -   time in seconds
      Real t

C   m,n,u,v -   counters used for the calculation of the area
C               inside the PV curve
      Integer m, n, u, v

C   p,w      -   counters used for the calculation of Power1 output
C               from engines 1 & 3
      Integer p, w

C   Powcycle     -   counter used to average the Power1 from x cycles
      Integer Powcycle

C   PP* -   intermediate variables used in the calculation of
PA,PB,PC,PD
      Real PP1, PP2, PP3, PP4

C   P1_,P2_,P3_ -   pressure of liquid in column
```

```
      Real P1A, P1B, P1C, P1D
      Real P2A, P2B, P2C, P2D
      Real P3A, P3B, P3C, P3D

C   i1,i3,i5 -    direction of liquid in column
      Real I1, I3, I5

C   X***-intermediate variables used in the calculation of P1_,P2_,P3_
      Real X11, X12, X13, X14, X15, X21
      Real X16, X17, X18, X19, X110, X22
      Real X111, X112, X113, X114, X115, X23
      Real X116, X117, X118, X119, X120, X24

C   Lt1,Lt2 -   lengths of the tuning pipes
      Real Lt1, Lt2

C   a***    -   liquid acceleration terms for each column of water
      Real a12a, a23a, a34a, a41a
      Real a12b, a23b, a34b, a41b
      Real a12c, a23c, a34c, a41c

C   vl***o  -   liquid velocity buffer
      Real vl12ao, vl23ao, vl34ao, vl41ao
      Real vl12bo, vl23bo, vl34bo, vl41bo
      Real vl12co, vl23co, vl34co, vl41co

C   Chamber,Up,Down -   counters used in the calculation of the mean gas
volume
      Integer Chamber
      Integer Up(12), Down(12)
      Real SumVol(12), sum

C-Variables used to calculate the Power1 output from engines 1 & 3
      Real PosArea, NegArea, AvePow1
      Real PosArea3, NegArea3, AvePow3
      Real PosCur1(2, 10000), NegCur1(2, 10000), Power1(1000)
      Real PosCur3(2, 10000), NegCur3(2, 10000), Power3(1000)
      Real Initial1, Initial3
      Integer ii

C   neg, sup  -   counters used to check if blowout or negative height
C      has occurred
      Integer neg, sup

C           -   variables used to determine convergance
      Integer CycleNum

C*****************************************************************
C*****************************************************************
C   Unloading Parameter Data from Passed Array
C*****************************************************************
C*****************************************************************


      Lt2 = X(2) / (1. + X(1))
      Lt1 = X(2) - Lt2
      LH = X(3)
      LC = X(4)
      LA = X(5)
      LB = X(6)
      VB = X(7)
      Full=X(8)
```

```
C*****************************************************************
C   Reading Parameter Data from Spreadsheet "Initial Conditions"
C*****************************************************************

      Pi = 3.1415927
      g = 981.
      f = 0.03
      d = 1.
      R = 82880600.
      Dia = 5.
      DiaBC = 4.5
      Dt = 0.001
      TH = 537.78
      TA = 316.
      TC = 286.
      TB = 289.
      Pmean = 1010000.
      Damp = 0.5

C*****************************************************************
C   Cross-sectional Area Calculations
C*****************************************************************

      Area = Pi * Dia ** 2. / 4.
      AreaBC = Pi * DiaBC ** 2. / 4.


C*****************************************************************
C   Initializing of liquid heights
C*****************************************************************

      If (LH .LE. LA) Then
           If (LH .LE. LC) Then
                IF (LH .LE. LB) Then
                     LL = LH * Full
                End If
           End If
      End IF

      If (LA .LE. LH) Then
           If (LA .LE. LC) Then
                IF (LA .LE. LB) Then
                     LL = LA * Full
                End If
           End If
      End IF

      If (LC .LE. LA) Then
           If (LC .LE. LH) Then
                IF (LC .LE. LB) Then
                     LL = LC * Full
                End If
           End If
      End IF

      If (LB .LE. LA) Then
           If (LB .LE. LC) Then
                IF (LB .LE. LH) Then
                     LL = LB * Full
                End If
           End If
      End IF

      LA1 = LL
```

```
      LH1 = LL
      LB1 = LL + 0.2
      LA2 = LL
      LC2 = LL
      LB2 = LL
      LA3 = LL
      LH3 = LL
      LB3 = LL - 0.2
      LA4 = LL
      LC4 = LL
      LB4 = LL

C*****************************************************************
C   Initializing of liquid velocities in each column
C*****************************************************************

      vl12a = 0.
      vl12b = 0.
      vl12c = 0.
      vl23a = 0.
      vl23b = 0.
      vl23c = 0.
      vl34a = 0.
      vl34b = 0.
      vl34c = 0.
      vl41a = 0.
      vl41b = 0.
      vl41c = 0.

C*****************************************************************
C   Initializing of gas volumes in each column
C*****************************************************************

      VoA1 = Area * (LA - LA1)
      VoH1 = Area * (LH - LH1)
      VoB1 = AreaBC * (LB - LB1)
      VoA2 = Area * (LA - LA2)
      VoC2 = Area * (LC - LC2)
      VoB2 = AreaBC * (LB - LB2)
      VoA3 = Area * (LA - LA3)
      VoH3 = Area * (LH - LH3)
      VoB3 = AreaBC * (LB - LB3)
      VoA4 = Area * (LA - LA4)
      VoC4 = Area * (LC - LC4)
      VoB4 = AreaBC * (LB - LB4)

C*****************************************************************
C   Initializing of gas pressures in each column
C*****************************************************************

      PA = Pmean
      PB = Pmean
      PC = Pmean
      PD = Pmean
      PB1 = Pmean
      PB2 = Pmean
      PB3 = Pmean
      PB4 = Pmean

C*****************************************************************
C   INITIALIZATION OF TIME AND OTHER COUNTERS
C*****************************************************************

      Do j = 1, 12
```

```
            MVCount(j) = 1
            Up(j) = 0
            Down(j) = 0
            SumVol(j) = 0.
      End Do

      Do j = 1 , 21
            MVA1(j) = Area * (LA - LL)
            MVA2(j) = Area * (LA - LL)
            MVA3(j) = Area * (LA - LL)
            MVA4(j) = Area * (LA - LL)
            MVH1(j) = Area * (LH - LL)
            MVC2(j) = Area * (LC - LL)
            MVH3(j) = Area * (LH - LL)
            MVC4(j) = Area * (LC - LL)
            MVB1(j) = AreaBC * (LB - LL)
            MVB2(j) = AreaBC * (LB - LL)
            MVB3(j) = AreaBC * (LB - LL)
            MVB4(j) = AreaBC * (LB - LL)
      End Do

      t = 0.
      m = 1
      n = 1
      p = 1
      u = 1
      v = 1
      w = 1
      Powcycle = 99
      Engine = 0.
      CycleNum = 0
      sum=0.
      PosArea=0.
      NegArea=0.
      AvePow1=0.
      AvePow3=0.
      Initial1=0.
      Initial3=0.
      neg=0
      sup=0

C*****************************************************************
C   BEGINNING OF MAIN LOOP
C*****************************************************************

1005  If (CycleNum .EQ. 1000) then
            GOTO 1009
      End IF

C*****************************************************************
C   INITIALIZATION OF GAS VOLUMES
C*****************************************************************

      VoH1o = VoH1
      VoA1o = VoA1
      VoC2o = VoC2
      VoH3o = VoH3
      VoA3o = VoA3
      VoC4o = VoC4
      VoA2o = VoA2
      VoA4o = VoA4
      VoB1o = VoB1
      VoB2o = VoB2
      VoB3o = VoB3
```

```
      VoB4o = VoB4

C*******************************************************************
C   INITIALIZATION OF GAS PRESSURES
C*******************************************************************

      PAo = PA
      PBo = PB
      PCo = PC
      PDo = PD
      PB1o = PB1
      PB2o = PB2
      PB3o = PB3
      PB4o = PB4

C*******************************************************************
C   CALCULATION OF GAS VOLUMES
C*******************************************************************

      VoA1 = Area * (LA - LA1)
      VoH1 = Area * (LH - LH1)
      VoB1 = AreaBC * (LB - LB1)
      VoA2 = Area * (LA - LA2)
      VoC2 = Area * (LC - LC2)
      VoB2 = AreaBC * (LB - LB2)
      VoA3 = Area * (LA - LA3)
      VoH3 = Area * (LH - LH3)
      VoB3 = AreaBC * (LB - LB3)
      VoA4 = Area * (LA - LA4)
      VoC4 = Area * (LC - LC4)
      VoB4 = AreaBC * (LB - LB4)

C*******************************************************************
C   CALCULATION OF GAS PRESSURES VIA THE IDEAL GAS LAW
C*******************************************************************

      PP1 = Pmean*(MVH1(21)/TH + 2. * VB / (TH + TA) + MVA1(21) / TA)
      PP3 = Pmean*(MVH3(21)/TH + 2. * VB / (TH + TA) + MVA3(21) / TA)
      PA = PP1/(VoH1 / TH + 2. * VB / (TH + TA) + VoA1 / TA)
      PC = PP3/(VoH3 / TH + 2. * VB / (TH + TA) + VoA3 / TA)
      PP2 = Pmean*(MVA2(21)/TA + 2. * VB / (TA + TC) + MVC2(21) / TC)
      PP4 = Pmean*(MVA4(21)/TA + 2. * VB / (TA + TC) + MVC4(21) / TC)
      PB = PP2/(VoA2 / TA + 2. * VB / (TA + TC) + VoC2 / TC)
      PD = PP4/(VoA4 / TA + 2. * VB / (TA + TC) + VoC4 / TC)
      PB1 = Pmean*MVB1(21) / VoB1
      PB2 = Pmean*MVB2(21) / VoB2
      PB3 = Pmean*MVB3(21) / VoB3
      PB4 = Pmean*MVB4(21) / VoB4

C*******************************************************************
C   CALCULATION OF LIQUID PRESSURES
C*******************************************************************

C P2A
      IF (vl12a.gt.0.) then
              I1 = 1
      Else
              I1 = -1
      End If

      IF (vl12b.gt.0.) then
            I3 = 1
      Else
            I3 = -1
```

```
      End If

      IF (vl12c.gt.0.) then
            I5 = 1
      Else
            I5 = -1
      End If

      X11 = PB1 + d * g * LB1 * (2. * (Dia/DiaBC)**2. + 1.)
      X12 = (PA - PA * Lt1 / (LH1 + Lt1)) * (LB1 / LH1) *
(Dia/DiaBC)**2.
      X13 = (PB - PB * Lt2 / (LA2 + Lt2)) * (LB1 / LA2) *
(Dia/DiaBC)**2.
      X14 = (d * g * LB1 * Lt1 / (LH1 + Lt1) + d * g * LB1 * Lt2 /
     1           (LA2 + Lt2)) * (Dia/DiaBC)**2.
      X15 = 1. + LB1 / (LH1 + Lt1) * (Dia/DiaBC)**2. + LB1 /
     1           (LA2 + Lt2) * (Dia/DiaBC)**2.
      X21 = 2. * f * d * LB1* (-I3 * vl12b**2. / DiaBC - I5 *
     1Dia * vl12c**2. / (DiaBC * DiaBC) + I1 * Dia * vl12a *
     2vl12a / (DiaBC * DiaBC))
      P2A = (X11 + X12 + X13 - X14 + X21) / X15

C P2B
      IF (vl23a.gt.0.) then
              I1 = 1
      Else
              I1 = -1
      End If

      IF (vl23b.gt.0.) then
            I3 = 1
      Else
            I3 = -1
      End If

      IF (vl23c.gt.0.) then
            I5 = 1
      Else
            I5 = -1
      End If

      X16 = PB2 + d * g * LB2 * (2. * (Dia/DiaBC)**2. + 1.)
      X17 = (PB - PB * Lt1 / (LC2 + Lt1)) * (LB2 / LC2) *
(Dia/DiaBC)**2.
      X18 = (PC - PC * Lt2 / (LA3 + Lt2)) * (LB2 / LA3) *
(Dia/DiaBC)**2.
      X19 = (d * g * LB2 * Lt1 / (LC2 + Lt1) + d * g * LB2 * Lt2 /
     1           (LA3 + Lt2)) * (Dia/DiaBC)**2.
      X110 = 1. + LB2 / (LC2+Lt1) * (Dia/DiaBC)**2. + LB2 /
     1           (LA3 + Lt2) * (Dia/DiaBC)**2.
      X22 = 2. * f * d * LB2 * (-I3 * vl23b**2. / DiaBC - I5 *
     1Dia * vl23c ** 2. / (DiaBC * DiaBC) + I1 * Dia * vl23a *
     2vl23a / (DiaBC * DiaBC))
      P2B = (X16 + X17 + X18 - X19 + X22) / X110

C P2C
      IF (vl34a.gt.0.) then
              I1 = 1
      Else
              I1 = -1
      End If

      IF (vl34b.gt.0.) then
            I3 = 1
```

```
      Else
            I3 = -1
      End If

      IF (vl34c.gt.0.) then
            I5 = 1
      Else
            I5 = -1
      End If

      X111 = PB3 + d * g * LB3 * (2. * (Dia/DiaBC)**2. + 1.)
      X112 = (PC - PC * Lt1 / (LH3 + Lt1)) * (LB3 / LH3) *
     1 (Dia/DiaBC)**2.
      X113 = (PD - PD * Lt2 / (LA4 + Lt2)) * (LB3 / LA4) *
     1 (Dia/DiaBC)**2.
      X114 = (d * g * LB3 * Lt1 / (LH3 + Lt1) + d * g * LB3 * Lt2 /
     1            (LA4 + Lt2)) * (Dia/DiaBC)**2.
      X115 = 1. + LB3 / (LH3 + Lt1) * (Dia/DiaBC)**2. + LB3 /
     1            (LA4 + Lt2) * (Dia/DiaBC)**2.
      X23 = 2. * f * d * LB3 * (-I3 * vl34b**2. / DiaBC - I5 *
     1 Dia * vl34c**2. / (DiaBC * DiaBC) + I1 * Dia *
     2 vl34a * vl34a / (DiaBC * DiaBC))
      P2C = (X111 + X112 + X113 - X114 + X23) / X115

C P2D
      IF (vl41a.gt.0.) then
              I1 = 1
      Else
              I1 = -1
      End If

      IF (vl41b.gt.0.) then
            I3 = 1
      Else
            I3 = -1
      End If

      IF (vl41c.gt.0.) then
            I5 = 1
      Else
            I5 = -1
      End If

      X116 = PB4 + d * g * LB4 * (2. * (Dia/DiaBC)**2.      + 1.)
      X117 = (PD - PD * Lt1 / (LC4 + Lt1)) * (LB4 / LC4) *
     1 (Dia/DiaBC)**2.
      X118 = (PA - PA * Lt2 / (LA1 + Lt2)) * (LB4 / LA1) *
     1 (Dia/DiaBC)**2.
      X119 = (d * g * LB4 * Lt1 / (LC4 + Lt1) + d * g * LB4 * Lt2 /
     1            (LA1 + Lt2)) * (Dia/DiaBC)**2.
      X120 = 1. + LB4 / (LC4 + Lt1) * (Dia/DiaBC)**2. + LB4 /
     1            (LA1 + Lt2) * (Dia/DiaBC)**2.
      X24 = 2. * f * d * LB4 * (-I3 * vl41b**2. / DiaBC - I5 *
     1 Dia * vl41c**2. / (DiaBC * DiaBC) + I1 * Dia * vl41a *
     2 vl41a / (DiaBC * DiaBC))
      P2D = (X116 + X117 + X118 - X119 + X24) / X120

      P1A = (PA * Lt1 + g * d * LH1 * Lt1 + P2A * LH1) / (LH1 + Lt1)
      P1B = (PB * Lt1 + g * d * LC2 * Lt1 + P2B * LC2) / (LC2 + Lt1)
      P1C = (PC * Lt1 + g * d * LH3 * Lt1 + P2C * LH3) / (LH3 + Lt1)
      P1D = (PD * Lt1 + g * d * LC4 * Lt1 + P2D * LC4) / (LC4 + Lt1)

      P3A = (g * d * Lt2 * LA2 + PB * Lt2 + P2A * LA2) / (Lt2 + LA2)
      P3B = (g * d * Lt2 * LA3 + PC * Lt2 + P2B * LA3) / (Lt2 + LA3)
```

```
      P3C = (g * d * Lt2 * LA4 + PD * Lt2 + P2C * LA4) / (Lt2 + LA4)
      P3D = (g * d * Lt2 * LA1 + PA * Lt2 + P2D * LA1) / (Lt2 + LA1)

C*****************************************************************
C   CALCULATION OF LIQUID ACCELERATIONS
C*****************************************************************

      If (vl12a .LE. 0.) Then
      a12a = (PA - P1A) / (d * LH1) + g + 2. * f * vl12a * vl12a / Dia
      End If
      If (vl12a .GT. 0.) Then
      a12a = (PA - P1A) / (d * LH1) + g - 2. * f * vl12a * vl12a / Dia
      End If
      If (vl23a .LE. 0.) Then
      a23a = (PB - P1B) / (d * LC2) + g + 2. * f * vl23a * vl23a / Dia
      End If
      If (vl23a .GT. 0.) Then
      a23a = (PB - P1B) / (d * LC2) + g - 2. * f * vl23a * vl23a / Dia
      End If
      If (vl34a .LE. 0.) Then
      a34a = (PC - P1C) / (d * LH3) + g + 2. * f * vl34a * vl34a / Dia
      End If
      If (vl34a .GT. 0.) Then
      a34a = (PC - P1C) / (d * LH3) + g - 2. * f * vl34a * vl34a / Dia
      End If
      If (vl41a .LE. 0.) Then
      a41a = (PD - P1D) / (d * LC4) + g + 2. * f * vl41a * vl41a / Dia
      End If
      If (vl41a .GT. 0.) Then
      a41a = (PD - P1D) / (d * LC4) + g - 2. * f * vl41a * vl41a / Dia
      End If
      If (vl12b .LE. 0.) Then
      a12b = (P2A - PB1) / (d * LB1) - g + 2. * f * vl12b **2. / DiaBC
      End If
      If (vl12b .GT. 0.) Then
      a12b = (P2A - PB1) / (d * LB1) - g - 2. * f * vl12b **2. / DiaBC
      End If
      If (vl23b .LE. 0.) Then
      a23b = (P2B - PB2) / (d * LB2) - g + 2. * f * vl23b **2. / DiaBC
      End If
      If (vl23b .GT. 0.) Then
      a23b = (P2B - PB2) / (d * LB2) - g - 2. * f * vl23b **2. / DiaBC
      End If
      If (vl34b .LE. 0.) Then
      a34b = (P2C - PB3) / (d * LB3) - g + 2. * f * vl34b **2. / DiaBC
      End If
      If (vl34b .GT. 0.) Then
      a34b = (P2C - PB3) / (d * LB3) - g - 2. * f * vl34b **2. / DiaBC
      End If
      If (vl41b .LE. 0.) Then
      a41b = (P2D - PB4) / (d * LB4) - g + 2. * f * vl41b **2. / DiaBC
      End If
      If (vl41b .GT. 0.) Then
      a41b = (P2D - PB4) / (d * LB4) - g - 2. * f * vl41b **2. / DiaBC
      End If

      a12c = a12a - (DiaBC / Dia) * (DiaBC / Dia) * a12b
      a23c = a23a - (DiaBC / Dia) * (DiaBC / Dia) * a23b
      a34c = a34a - (DiaBC / Dia) * (DiaBC / Dia) * a34b
      a41c = a41a - (DiaBC / Dia) * (DiaBC / Dia) * a41b

      t = t + Dt

C*****************************************************************
```

```
C   CALCULATION OF VELOCITY BUFFER
C*****************************************************************

      vl12ao = vl12a
      vl12bo = vl12b
      vl12co = vl12c
      vl23ao = vl23a
      vl23bo = vl23b
      vl23co = vl23c
      vl34ao = vl34a
      vl34bo = vl34b
      vl34co = vl34c
      vl41ao = vl41a
      vl41bo = vl41b
      vl41co = vl41c

C*****************************************************************
C   CALCULATION OF LIQUID VELOCITIES
C*****************************************************************

      vl12a = vl12ao + a12a * Dt
      vl12b = vl12bo + a12b * Dt
      vl12c = vl12co + a12c * Dt
      vl23a = vl23ao + a23a * Dt
      vl23b = vl23bo + a23b * Dt
      vl23c = vl23co + a23c * Dt
      vl34a = vl34ao + a34a * Dt
      vl34b = vl34bo + a34b * Dt
      vl34c = vl34co + a34c * Dt
      vl41a = vl41ao + a41a * Dt
      vl41b = vl41bo + a41b * Dt
      vl41c = vl41co + a41c * Dt

C*****************************************************************
C   CALCULATION OF LIQUID POSITIONS IN EACH COLUMN
C*****************************************************************

      LH1 = LH1 - vl12ao * Dt - 0.5 * a12a * Dt * Dt
      LB1 = LB1 + vl12bo * Dt + 0.5 * a12b * Dt * Dt
      LA2 = LA2 + vl12co * Dt + 0.5 * a12c * Dt * Dt
      LC2 = LC2 - vl23ao * Dt - 0.5 * a23a * Dt * Dt
      LB2 = LB2 + vl23bo * Dt + 0.5 * a23b * Dt * Dt
      LA3 = LA3 + vl23co * Dt + 0.5 * a23c * Dt * Dt
      LH3 = LH3 - vl34ao * Dt - 0.5 * a34a * Dt * Dt
      LB3 = LB3 + vl34bo * Dt + 0.5 * a34b * Dt * Dt
      LA4 = LA4 + vl34co * Dt + 0.5 * a34c * Dt * Dt
      LC4 = LC4 - vl41ao * Dt - 0.5 * a41a * Dt * Dt
      LB4 = LB4 + vl41bo * Dt + 0.5 * a41b * Dt * Dt
      LA1 = LA1 + vl41co * Dt + 0.5 * a41c * Dt * Dt

C*****************************************************************
C   CALCULATION OF MEAN GAS VOLUME IN EACH CHAMBER
C*****************************************************************

C***********************     Ambient Chamber 1     **********************

      Chamber = 1

      If (VoA1 .GT. VoA1o) Then
           If (Up(Chamber) .GT. 0) Then
                MVA1(MVCount(Chamber)) = SumVol(Chamber) /
     1                 (Up(Chamber) + Down(Chamber))
                sum = 0.
                DO j = 1 , 20
```

```
                    sum = MVA1(j) / 20. + sum
                End DO
                MVA1(21) = (sum - MVA1(21)) * Damp + sum
                MVCount(Chamber) = MVCount(Chamber) + 1
                If (MVCount(Chamber) .EQ. 21) Then
                     MVCount(Chamber) = 1
                End IF
                Up(Chamber) = 0
                Down(Chamber) = 0
                SumVol(Chamber) = 0.
           End If
           Down(Chamber) = Down(Chamber) + 1
           SumVol(Chamber) = SumVol(Chamber) + VoA1
      Else
           SumVol(Chamber) = SumVol(Chamber) + VoA1
           Up(Chamber) = Up(Chamber) + 1
      End If

C***********************     Hot Chamber 1     **********************

      Chamber = 2

      If (VoH1 .GT. VoH1o) Then
           If (Up(Chamber) .GT. 0) Then
                MVH1(MVCount(Chamber)) = SumVol(Chamber) /
     1                (Up(Chamber) + Down(Chamber))
                sum = 0.
                Do j = 1 , 20.
                     sum = MVH1(j) / 20. + sum
                End Do
                MVH1(21) = (sum - MVH1(21)) * Damp + sum
                MVCount(Chamber) = MVCount(Chamber) + 1
                If (MVCount(Chamber) .EQ. 21) Then
                     MVCount(Chamber) = 1
                End If
                Up(Chamber) = 0
                Down(Chamber) = 0
                SumVol(Chamber) = 0.
           End If
           Down(Chamber) = Down(Chamber) + 1
           SumVol(Chamber) = SumVol(Chamber) + VoH1
      Else
           SumVol(Chamber) = SumVol(Chamber) + VoH1
           Up(Chamber) = Up(Chamber) + 1
      End If

C***********************     Bounce Chamber 1     **********************

      Chamber = 3

      If (VoB1 .GT. VoB1o) Then
           If (Up(Chamber) .GT. 0) Then
                MVB1(MVCount(Chamber)) = SumVol(Chamber) /
     1                (Up(Chamber) + Down(Chamber))
                sum = 0.
                Do j = 1 ,20
                     sum = MVB1(j) / 20. + sum
                End Do
                MVB1(21) = (sum - MVB1(21)) * Damp + sum
                MVCount(Chamber) = MVCount(Chamber) + 1
                If (MVCount(Chamber) .EQ. 21)  Then
                     MVCount(Chamber) = 1
                End If
                Up(Chamber) = 0
```

```
               Down(Chamber) = 0
               SumVol(Chamber) = 0.
          End If
          Down(Chamber) = Down(Chamber) + 1
          SumVol(Chamber) = SumVol(Chamber) + VoB1
     Else
          SumVol(Chamber) = SumVol(Chamber) + VoB1
          Up(Chamber) = Up(Chamber) + 1
     End If

C***********************    Ambient Chamber 2    ***********************

     Chamber = 4

     If (VoA2 .GT. VoA2o) Then
          If (Up(Chamber) .GT. 0) Then
               MVA2(MVCount(Chamber)) = SumVol(Chamber) /
     1                (Up(Chamber) + Down(Chamber))
               sum = 0.
               Do j = 1 , 20
                     sum = MVA2(j) / 20. + sum
               End Do
               MVA2(21) = (sum - MVA2(21)) * Damp + sum
               MVCount(Chamber) = MVCount(Chamber) + 1
               If (MVCount(Chamber) .EQ. 21) Then
                    MVCount(Chamber) = 1
               End If
               Up(Chamber) = 0
               Down(Chamber) = 0
               SumVol(Chamber) = 0.
          End If
          Down(Chamber) = Down(Chamber) + 1
          SumVol(Chamber) = SumVol(Chamber) + VoA2
     Else
          SumVol(Chamber) = SumVol(Chamber) + VoA2
          Up(Chamber) = Up(Chamber) + 1
     End If

C***********************    Cold Chamber 2    ***********************

     Chamber = 5

     If (VoC2 .GT. VoC2o) Then
          If (Up(Chamber) .GT. 0) Then
               MVC2(MVCount(Chamber)) = SumVol(Chamber) /
     1                (Up(Chamber) + Down(Chamber))
               sum = 0.
               Do j = 1 , 20
                    sum = MVC2(j) / 20. + sum
               End Do
               MVC2(21) = (sum - MVC2(21)) * Damp + sum
               MVCount(Chamber) = MVCount(Chamber) + 1
               If (MVCount(Chamber) .EQ. 21) Then
                    MVCount(Chamber) = 1
               End If
               Up(Chamber) = 0
               Down(Chamber) = 0
               SumVol(Chamber) = 0.
          End If
          Down(Chamber) = Down(Chamber) + 1
          SumVol(Chamber) = SumVol(Chamber) + VoC2
     Else
          SumVol(Chamber) = SumVol(Chamber) + VoC2
          Up(Chamber) = Up(Chamber) + 1
```

```
      End If
C***********************    Bounce Chamber 2    ***********************
      Chamber = 6

      If (VoB2 .GT. VoB2o) Then
           If (Up(Chamber) .GT. 0) Then
                MVB2(MVCount(Chamber)) = SumVol(Chamber) /
     1                   (Up(Chamber) + Down(Chamber))
                sum = 0.
                Do j = 1 , 20
                      sum = MVB2(j) / 20. + sum
                End DO
                MVB2(21) = (sum - MVB2(21)) * Damp + sum
                MVCount(Chamber) = MVCount(Chamber) + 1
                If (MVCount(Chamber) .EQ. 21) Then
                     MVCount(Chamber) = 1
                End If
                Up(Chamber) = 0
                Down(Chamber) = 0
                SumVol(Chamber) = 0.
           End If
           Down(Chamber) = Down(Chamber) + 1
           SumVol(Chamber) = SumVol(Chamber) + VoB2
      Else
           SumVol(Chamber) = SumVol(Chamber) + VoB2
           Up(Chamber) = Up(Chamber) + 1
      End If
C***********************    Ambient Chamber 3    ***********************
      Chamber = 7

      If (VoA3 .GT. VoA3o) Then
           If (Up(Chamber) .GT. 0) Then
                MVA3(MVCount(Chamber)) = SumVol(Chamber) /
     1                   (Up(Chamber) + Down(Chamber))
                sum = 0.
                Do j = 1 , 20
                     sum = MVA3(j) / 20. + sum
                End Do
                MVA3(21) = (sum - MVA3(21)) * Damp + sum
                MVCount(Chamber) = MVCount(Chamber) + 1
                If (MVCount(Chamber) .EQ. 21) Then
                     MVCount(Chamber) = 1
                End If
                Up(Chamber) = 0
                Down(Chamber) = 0
                SumVol(Chamber) = 0.
           End If
           Down(Chamber) = Down(Chamber) + 1
           SumVol(Chamber) = SumVol(Chamber) + VoA3
      Else
           SumVol(Chamber) = SumVol(Chamber) + VoA3
           Up(Chamber) = Up(Chamber) + 1
      End If
C***********************    Hot Chamber 3    **********************
      Chamber = 8

      If (VoH3 .GT. VoH3o) Then
              If (Up(Chamber) .GT. 0) Then
```

```
                MVH3(MVCount(Chamber)) = SumVol(Chamber) /
     1                 (Up(Chamber) + Down(Chamber))
                sum = 0.
                Do j = 1 , 20
                     sum = MVH3(j) / 20. + sum
                End Do
                MVH3(21) = (sum - MVH3(21)) * Damp + sum
                MVCount(Chamber) = MVCount(Chamber) + 1
                If (MVCount(Chamber) .EQ. 21) Then
                     MVCount(Chamber) = 1
                End If
                Up(Chamber) = 0
                Down(Chamber) = 0
                SumVol(Chamber) = 0.
           End If
           Down(Chamber) = Down(Chamber) + 1
           SumVol(Chamber) = SumVol(Chamber) + VoH3
      Else
           SumVol(Chamber) = SumVol(Chamber) + VoH3
           Up(Chamber) = Up(Chamber) + 1
      End If

C***********************    Bounce Chamber 3    **********************

      Chamber = 9

      If (VoB3 .GT. VoB3o) Then
           If (Up(Chamber) .GT. 0) Then
                MVB3(MVCount(Chamber)) = SumVol(Chamber) /
     1                 (Up(Chamber) + Down(Chamber))
                sum = 0.
                Do j = 1 , 20
                     sum = MVB3(j) / 20. + sum
                End Do
                MVB3(21) = (sum - MVB3(21)) * Damp + sum
                MVCount(Chamber) = MVCount(Chamber) + 1
                If (MVCount(Chamber).EQ. 21) Then
                     MVCount(Chamber) = 1
                End If
                Up(Chamber) = 0
                Down(Chamber) = 0
                SumVol(Chamber) = 0.
           End If
           Down(Chamber) = Down(Chamber) + 1
           SumVol(Chamber) = SumVol(Chamber) + VoB3
      Else
           SumVol(Chamber) = SumVol(Chamber) + VoB3
           Up(Chamber) = Up(Chamber) + 1
      End If

C***********************    Ambient Chamber 4    **********************

      Chamber = 10

      If (VoA4 .GT. VoA4o) Then
           If (Up(Chamber) .GT. 0) Then
                MVA4(MVCount(Chamber)) = SumVol(Chamber) /
     1                 (Up(Chamber) + Down(Chamber))
                sum = 0.
                Do j = 1 , 20
                     sum = MVA4(j) / 20. + sum
                End DO
                MVA4(21) = (sum - MVA4(21)) * Damp + sum
                MVCount(Chamber) = MVCount(Chamber) + 1
```

```
                If (MVCount(Chamber) .EQ. 21) Then
                     MVCount(Chamber) = 1
                End If
                Up(Chamber) = 0
                Down(Chamber) = 0
                SumVol(Chamber) = 0.
           End If
           Down(Chamber) = Down(Chamber) + 1
           SumVol(Chamber) = SumVol(Chamber) + VoA4
      Else
           SumVol(Chamber) = SumVol(Chamber) + VoA4
           Up(Chamber) = Up(Chamber) + 1
      End If

C***********************    Cold Chamber 4    ***********************

      Chamber = 11

      If (VoC4 .GT. VoC4o) Then
           If (Up(Chamber) .GT. 0) Then
                MVC4(MVCount(Chamber)) = SumVol(Chamber) /
     1                 (Up(Chamber) + Down(Chamber))
                sum = 0.
                Do j = 1 , 20
                     sum = MVC4(j) / 20. + sum
                End DO
                MVC4(21) = (sum - MVC4(21)) * Damp + sum
                MVCount(Chamber) = MVCount(Chamber) + 1
                If (MVCount(Chamber) .EQ. 21) Then
                     MVCount(Chamber) = 1
                End If
                Up(Chamber) = 0
                Down(Chamber) = 0
                SumVol(Chamber) = 0.
           End If
           Down(Chamber) = Down(Chamber) + 1
           SumVol(Chamber) = SumVol(Chamber) + VoC4
      Else
           SumVol(Chamber) = SumVol(Chamber) + VoC4
           Up(Chamber) = Up(Chamber) + 1
      End If

C***********************    Bounce Chamber 4    ***********************

      Chamber = 12

      If (VoB4 .GT. VoB4o) Then
           If (Up(Chamber) .GT. 0) Then
                MVB4(MVCount(Chamber)) = SumVol(Chamber) /
     1                 (Up(Chamber) + Down(Chamber))
                sum = 0.
                Do j = 1 , 20
                     sum = MVB4(j) / 20. + sum
                End Do
                MVB4(21) = (sum - MVB4(21)) * Damp + sum
                MVCount(Chamber) = MVCount(Chamber) + 1
                If (MVCount(Chamber) .EQ. 21) Then
                     MVCount(Chamber) = 1
                End If
                Up(Chamber) = 0
                Down(Chamber) = 0
                SumVol(Chamber) = 0.
           End If
           Down(Chamber) = Down(Chamber) + 1
```

```
           SumVol(Chamber) = SumVol(Chamber) + VoB4
      Else
           SumVol(Chamber) = SumVol(Chamber) + VoB4
           Up(Chamber) = Up(Chamber) + 1
      End If

C*****************************************************************
C   CALCULATION OF Power PER CYCLE
C*****************************************************************

      If ((VoH1 + VoA1) .GT. (VoH1o + VoA1o)) Then
           If (n .GT. 1) Then
                CycleNum = CycleNum+1
                PosArea = PVArea(PosCur1, m, PosArea)
                NegArea = PVArea(NegCur1, n, NegArea)
                Power1(p) = (NegArea-PosArea)/(t-Initial1)/10000000
                Initial1 = t
                m = 1
                n = 1
                p = p + 1
                If (p .EQ. (Powcycle + 1)) Then
                     AvePow1 = 0.
                     Do ii = 1 , Powcycle
                          AvePow1 = Power1(ii)/Powcycle+AvePow1
                     End Do
                     p = 1
                End If
           End If
           PosCur1(1, m) = PA
           PosCur1(2, m) = VoH1 + VoA1
           m = m + 1
      End If
      If ((VoH1 + VoA1) .LT. (VoH1o + VoA1o)) Then
           NegCur1(1, N) = PA
           NegCur1(2, N) = VoH1o + VoA1o
           n = n + 1
      End If

C   Engine 3 Power1

      If ((VoH3 + VoA3) .GT. (VoH3o + VoA3o)) Then
           If (u .GT. 1) Then
                PosArea3 = PVArea(PosCur3, v, PosArea3)
                NegArea3 = PVArea(NegCur3, u, NegArea3)
                Power3(w)=(NegArea3-PosArea3)/(t-Initial3)/10000000
                Initial3 = t
                v = 1
                u = 1
                w = w + 1
                If (w .EQ. (Powcycle + 1)) Then
                     AvePow3 = 0.
                     Do ii = 1 , Powcycle
                         AvePow3=Power3(ii)/Powcycle+AvePow3
                     End Do
                           w = 1
                End If
           End If
           PosCur3(1, v) = PC
           PosCur3(2, v) = VoH3 + VoA3
           v = v + 1
      End If
      If ((VoH3 + VoA3) .LT. (VoH3o + VoA3o)) Then
           NegCur3(1, u) = PC
           NegCur3(2, u) = VoH3o + VoA3o
```

```
          u = u + 1
      End If

C*****************************************************************
C   NEGATIVE HEIGHT CHECK
C*****************************************************************

      If (LH1 .LT. 0.) Then
      neg = neg + 1
      End If
      If (LB1 .LT. 0.) Then
      neg = neg + 1
      End If
      If (LA2 .LT. 0.) Then
      neg = neg + 1
      End If
      If (LC2 .LT. 0.) Then
      neg = neg + 1
      End If
      If (LB2 .LT. 0.) Then
      neg = neg + 1
      End If
      If (LA3 .LT. 0.) Then
      neg = neg + 1
      End If
      If (LH3 .LT. 0.) Then
      neg = neg + 1
      End If
      If (LB3 .LT. 0.) Then
      neg = neg + 1
      End If
      If (LA4 .LT. 0.) Then
      neg = neg + 1
      End If
      If (LC4 .LT. 0.) Then
      neg = neg + 1
      End If
      If (LB4 .LT. 0.) Then
      neg = neg + 1
      End If
      If (LA1 .LT. 0.) Then
      neg = neg + 1
      End If
      If (neg .GE. 1) Then
           Engine = -10
           GoTo 1009
      End If

C*****************************************************************
C   BLOWOUT CHECK
C*****************************************************************

      If (LH1 .GT. LH) Then
      sup = sup + 1
      End If
      If (LB1 .GT. LB) Then
      sup = sup + 1
      End If
      If (LA2 .GT. LA) Then
      sup = sup + 1
      End If
      If (LC2 .GT. LC) Then
      sup = sup + 1
      End If
```

```
      If (LB2 .GT. LB) Then
      sup = sup + 1
      End If
      If (LA3 .GT. LA) Then
      sup = sup + 1
      End If
      If (LH3 .GT. LH) Then
      sup = sup + 1
      End If
      If (LB3 .GT. LB) Then
      sup = sup + 1
      End If
      If (LA4 .GT. LA) Then
      sup = sup + 1
      End If
      If (LC4 .GT. LC) Then
      sup = sup + 1
      End If
      If (LB4 .GT. LB) Then
      sup = sup + 1
      End If
      If (LA1 .GT. LA) Then
      sup = sup + 1
      End If
      If (sup .GE. 1) Then
           Engine = -10
           GoTo 1009
      End If
C*****************************************************************
C   REPEAT LOOP
C*****************************************************************

      GoTo 1005


C*****************************************************************
C   Quit computation and return results
C*****************************************************************

1009  Engine = (AvePow1 + AvePow3) / 2.
      VAR=VAR
      Return
      End


      Function PVArea(Curve, num, Area)
      Real Curve(2,*), Area
      Integer jj, num

      Area=0.
      Do jj = 2 , num
           Area = Area + Abs(((Curve(1, jj - 1) +
     1 Curve(1, jj)) / 2) * (Curve(2, jj - 1) - Curve(2, jj)))
      End Do
      PVArea=Area

      Return
      End

C ********************************************************************
C ********************************************************************
C ********************************************************************
```

```
C **********************************************************************

      SUBROUTINE SA(N,X,MAX,RT,EPS,NS,NT,NEPS,MAXEVL,LB,UB,C,IPRINT,
     1              ISEED1,ISEED2,T,VM,XOPT,FOPT,NACC,NFCNEV,NOBDS,IER,
     2              FSTAR,XP,NACP)

C  Version: 3.2
C  Date: 1/22/94.
C  Differences compared to Version 2.0:
C     1. If a trial is out of bounds, a point is randomly selected
C        from LB(i) to UB(i). Unlike in version 2.0, this trial is
C        evaluated and is counted in acceptances and rejections.
C        All corresponding documentation was changed as well.
C  Differences compared to Version 3.0:
C     1. If VM(i) > (UB(i) - LB(i)), VM is set to UB(i) - LB(i).
C        The idea is that if T is high relative to LB & UB, most
C        points will be accepted, causing VM to rise. But, in this
C        situation, VM has little meaning; particularly if VM is
C        larger than the acceptable region. Setting VM to this size
C        still allows all parts of the allowable region to be selected.
C  Differences compared to Version 3.1:
C     1. Test made to see if the initial temperature is positive.
C     2. WRITE statements prettied up.
C     3. References to paper updated.
C
C  Synopsis:
C  This routine implements the continuous simulated annealing global
C  optimization algorithm described in Corana et al.'s article
C  "Minimizing Multimodal Functions of Continuous Variables with the
C  "Simulated Annealing" Algorithm" in the September 1987 (vol. 13,
C  no. 3, pp. 262-280) issue of the ACM Transactions on Mathematical
C  Software.
C
C  A very quick (perhaps too quick) overview of SA:
C     SA tries to find the global optimum of an N dimensional function.
C  It moves both up and downhill and as the optimization process
C  proceeds, it focuses on the most promising area.
C     To start, it randomly chooses a trial point within the step length
C  VM (a vector of length N) of the user selected starting point. The
C  function is evaluated at this trial point and its value is compared
C  to its value at the initial point.
C     In a maximization problem, all uphill moves are accepted and the
C  algorithm continues from that trial point. Downhill moves may be
C  accepted; the decision is made by the Metropolis criteria. It uses T
C  (temperature) and the size of the downhill move in a probabilistic
C  manner. The smaller T and the size of the downhill move are, the more
C  likely that move will be accepted. If the trial is accepted, the
C  algorithm moves on from that point. If it is rejected, another point
C  is chosen instead for a trial evaluation.
C     Each element of VM periodically adjusted so that half of all
C  function evaluations in that direction are accepted.
C     A fall in T is imposed upon the system with the RT variable by
C  T(i+1) = RT*T(i) where i is the ith iteration. Thus, as T declines,
C  downhill moves are less likely to be accepted and the percentage of
C  rejections rise. Given the scheme for the selection for VM, VM falls.
C  Thus, as T declines, VM falls and SA focuses upon the most promising
C  area for optimization.
C
C  The importance of the parameter T:
C     The parameter T is crucial in using SA successfully. It influences
C  VM, the step length over which the algorithm searches for optima. For
C  a small intial T, the step length may be too small; thus not enough
C  of the function might be evaluated to find the global optima. The
user
```

```
C  should carefully examine VM in the intermediate output (set IPRINT =
C  1) to make sure that VM is appropriate. The relationship between the
C  initial temperature and the resulting step length is function
C  dependent.
C     To determine the starting temperature that is consistent with
C  optimizing a function, it is worthwhile to run a trial run first. Set
C  RT = 1.5 and T = 1.0. With RT > 1.0, the temperature increases and VM
C  rises as well. Then select the T that produces a large enough VM.
C
C  For modifications to the algorithm and many details on its use,
C  (particularly for econometric applications) see Goffe, Ferrier
C  and Rogers, "Global Optimization of Statistical Functions with
C  Simulated Annealing," Journal of Econometrics, vol. 60, no. 1/2,
C  Jan./Feb. 1994, pp. 65-100.
C  For more information, contact
C              Bill Goffe
C              Department of Economics and International Business
C              University of Southern Mississippi
C              Hattiesburg, MS  39506-5072
C              (601) 266-4484 (office)
C              (601) 266-4920 (fax)
C              bgoffe@whale.st.usm.edu (Internet)
C
C  As far as possible, the parameters here have the same name as in
C  the description of the algorithm on pp. 266-8 of Corana et al.
C
C  In this description, SP is single precision, DP is double precision,
C  INT is integer, L is logical and (N) denotes an array of length n.
C  Thus, DP(N) denotes a double precision array of length n.
C
C  Input Parameters:
C    Note: The suggested values generally come from Corana et al. To
C          drastically reduce runtime, see Goffe et al., pp. 90-1 for
C          suggestions on choosing the appropriate RT and NT.
C    N - Number of variables in the function to be optimized. (INT)
C    X - The starting values for the variables of the function to be
C        optimized. (DP(N))
C    MAX - Denotes whether the function should be maximized or
C          minimized. A true value denotes maximization while a false
C          value denotes minimization. Intermediate output (see IPRINT)
C          takes this into account. (L)
C    RT - The temperature reduction factor. The value suggested by
C         Corana et al. is .85. See Goffe et al. for more advice. (DP)
C    EPS - Error tolerance for termination. If the final function
C          values from the last neps temperatures differ from the
C          corresponding value at the current temperature by less than
C          EPS and the final function value at the current temperature
C          differs from the current optimal function value by less than
C          EPS, execution terminates and IER = 0 is returned. (EP)
C    NS - Number of cycles. After NS*N function evaluations, each
C         element of VM is adjusted so that approximately half of
C         all function evaluations are accepted. The suggested value
C         is 20. (INT)
C    NT - Number of iterations before temperature reduction. After
C         NT*NS*N function evaluations, temperature (T) is changed
C         by the factor RT. Value suggested by Corana et al. is
C         MAX(100, 5*N). See Goffe et al. for further advice. (INT)
C    NEPS - Number of final function values used to decide upon termi-
C           nation. See EPS. Suggested value is 4. (INT)
C    MAXEVL - The maximum number of function evaluations. If it is
C             exceeded, IER = 1. (INT)
C    LB - The lower bound for the allowable solution variables. (DP(N))
C    UB - The upper bound for the allowable solution variables. (DP(N))
C         If the algorithm chooses X(I) .LT. LB(I) or X(I) .GT. UB(I),
```

```
C        I = 1, N, a point is from inside is randomly selected. This
C        This focuses the algorithm on the region inside UB and LB.
C        Unless the user wishes to concentrate the search to a par-
C        ticular region, UB and LB should be set to very large positive
C        and negative values, respectively. Note that the starting
C        vector X should be inside this region. Also note that LB and
C        UB are fixed in position, while VM is centered on the last
C        accepted trial set of variables that optimizes the function.
C    C - Vector that controls the step length adjustment. The suggested
C        value for all elements is 2.0. (DP(N))
C    IPRINT - controls printing inside SA. (INT)
C            Values: 0 - Nothing printed.
C                    1 - Function value for the starting value and
C                        summary results before each temperature
C                        reduction. This includes the optimal
C                        function value found so far, the total
C                        number of moves (broken up into uphill,
C                        downhill, accepted and rejected), the
C                        number of out of bounds trials, the
C                        number of new optima found at this
C                        temperature, the current optimal X and
C                        the step length VM. Note that there are
C                        N*NS*NT function evalutations before each
C                        temperature reduction. Finally, notice is
C                        is also given upon achieveing the termination
C                        criteria.
C                    2 - Each new step length (VM), the current optimal
C                        X (XOPT) and the current trial X (X). This
C                        gives the user some idea about how far X
C                        strays from XOPT as well as how VM is adapting
C                        to the function.
C                    3 - Each function evaluation, its acceptance or
C                        rejection and new optima. For many problems,
C                        this option will likely require a small tree
C                        if hard copy is used. This option is best
C                        used to learn about the algorithm. A small
C                        value for MAXEVL is thus recommended when
C                        using IPRINT = 3.
C            Suggested value: 1
C            Note: For a given value of IPRINT, the lower valued
C                  options (other than 0) are utilized.
C    ISEED1 - The first seed for the random number generator RANMAR.
C             0 .LE. ISEED1 .LE. 31328. (INT)
C    ISEED2 - The second seed for the random number generator RANMAR.
C             0 .LE. ISEED2 .LE. 30081. Different values for ISEED1
C             and ISEED2 will lead to an entirely different sequence
C             of trial points and decisions on downhill moves (when
C             maximizing). See Goffe et al. on how this can be used
C             to test the results of SA. (INT)
C
C  Input/Output Parameters:
C    T - On input, the initial temperature. See Goffe et al. for advice.
C        On output, the final temperature. (DP)
C    VM - The step length vector. On input it should encompass the
C        region of interest given the starting value X. For point
C        X(I), the next trial point is selected is from X(I) - VM(I)
C        to  X(I) + VM(I). Since VM is adjusted so that about half
C        of all points are accepted, the input value is not very
C        important (i.e. is the value is off, SA adjusts VM to the
C        correct value). (DP(N))
C
C  Output Parameters:
C    XOPT - The variables that optimize the function. (DP(N))
C    FOPT - The optimal value of the function. (DP)
```

```
C    NACC - The number of accepted function evaluations. (INT)
C    NFCNEV - The total number of function evaluations. In a minor
C             point, note that the first evaluation is not used in the
C             core of the algorithm; it simply initializes the
C             algorithm. (INT).
C    NOBDS - The total number of trial function evaluations that
C            would have been out of bounds of LB and UB. Note that
C            a trial point is randomly selected between LB and UB.
C            (INT)
C    IER - The error return number. (INT)
C          Values: 0 - Normal return; termination criteria achieved.
C                  1 - Number of function evaluations (NFCNEV) is
C                      greater than the maximum number (MAXEVL).
C                  2 - The starting value (X) is not inside the
C                      bounds (LB and UB).
C                  3 - The initial temperature is not positive.
C                  99 - Should not be seen; only used internally.
C
C  Work arrays that must be dimensioned in the calling routine:
C       RWK1 (DP(NEPS))  (FSTAR in SA)
C       RWK2 (DP(N))     (XP    "  " )
C       IWK  (INT(N))    (NACP  "  " )
C
C  Required Functions (included):
C    EXPREP - Replaces the function EXP to avoid under- and overflows.
C             It may have to be modified for non IBM-type main-
C             frames. (DP)
C    RMARIN - Initializes the random number generator RANMAR.
C    RANMAR - The actual random number generator. Note that
C             RMARIN must run first (SA does this). It produces uniform
C             random numbers on [0,1]. These routines are from
C             Usenet's comp.lang.fortran. For a reference, see
C             "Toward a Universal Random Number Generator"
C             by George Marsaglia and Arif Zaman, Florida State
C             University Report: FSU-SCRI-87-50 (1987).
C             It was later modified by F. James and published in
C             "A Review of Pseudo-random Number Generators." For
C             further information, contact stuart@ads.com. These
C             routines are designed to be portable on any machine
C             with a 24-bit or more mantissa. I have found it produces
C             identical results on a IBM 3081 and a Cray Y-MP.
C
C  Required Subroutines (included):
C    PRTVEC - Prints vectors.
C    PRT1 ... PRT10 - Prints intermediate output.
C    FCN - Function to be optimized. The form is
C            SUBROUTINE FCN(N,X,F)
C            INTEGER N
C            DOUBLE PRECISION  X(N), F
C            ...
C            function code with F = F(X)
C            ...
C            RETURN
C            END
C          Note: This is the same form used in the multivariable
C          minimization algorithms in the IMSL edition 10 library.
C
C  Machine Specific Features:
C    1. EXPREP may have to be modified if used on non-IBM type main-
C       frames. Watch for under- and overflows in EXPREP.
C    2. Some FORMAT statements use G25.18; this may be excessive for
C       some machines.
C    3. RMARIN and RANMAR are designed to be protable; they should not
C       cause any problems.
```

```
C  Type all external variables.
      DOUBLE PRECISION  X(*), LB(*), UB(*), C(*), VM(*), FSTAR(*),
     1                  XOPT(*), XP(*), T, EPS, RT, FOPT
      INTEGER  NACP(*), N, NS, NT, NEPS, NACC, MAXEVL, IPRINT,
     1         NOBDS, IER, NFCNEV, ISEED1, ISEED2
      LOGICAL  MAX

C  Type all internal variables.
      DOUBLE PRECISION  F, FP, P, PP, RATIO
      INTEGER  NUP, NDOWN, NREJ, NNEW, LNOBDS, H, I, J, M
      LOGICAL  QUIT

C  Type all functions.
      DOUBLE PRECISION  EXPREP
      REAL  RANMAR

C  Initialize the random number generator RANMAR.
      CALL RMARIN(ISEED1,ISEED2)

C  Set initial values.
      NACC = 0
      NOBDS = 0
      NFCNEV = 0
      IER = 99

      DO 10, I = 1, N
         XOPT(I) = X(I)
         NACP(I) = 0
10    CONTINUE

      DO 20, I = 1, NEPS
         FSTAR(I) = 1.0D+20
20    CONTINUE

C  If the initial temperature is not positive, notify the user and
C  return to the calling routine.
      IF (T .LE. 0.0) THEN
       WRITE(7,'(/,''  THE INITIAL TEMPERATURE IS NOT POSITIVE. ''
     1             /,''  RESET THE VARIABLE T. ''/)')
         IER = 3
         RETURN
      END IF

C  If the initial value is out of bounds, notify the user and return
C  to the calling routine.
      DO 30, I = 1, N
         IF ((X(I) .GT. UB(I)) .OR. (X(I) .LT. LB(I))) THEN
            CALL PRT1
            IER = 2
            RETURN
         END IF
30    CONTINUE

C  Evaluate the function with input X and return value as F.
      CALL FCN(N,X,F)

C  If the function is to be minimized, switch the sign of the function.
C  Note that all intermediate and final output switches the sign back
C  to eliminate any possible confusion for the user.
      IF(.NOT. MAX) F = -F
      NFCNEV = NFCNEV + 1
      FOPT = F
      FSTAR(1) = F
```

```
      IF(IPRINT .GE. 1) CALL PRT2(MAX,N,X,F)

C  Start the main loop. Note that it terminates if (i) the algorithm
C  succesfully optimizes the function or (ii) there are too many
C  function evaluations (more than MAXEVL).
100   NUP = 0
      NREJ = 0
      NNEW = 0
      NDOWN = 0
      LNOBDS = 0

      DO 400, M = 1, NT
         DO 300, J = 1, NS
            DO 200, H = 1, N

C  Generate XP, the trial value of X. Note use of VM to choose XP.
               DO 110, I = 1, N
                  IF (I .EQ. H) THEN
                     XP(I) = X(I) + (RANMAR()*2.- 1.) * VM(I)
                  ELSE
                     XP(I) = X(I)
                  END IF

C  If XP is out of bounds, select a point in bounds for the trial.
                  IF((XP(I) .LT. LB(I)) .OR. (XP(I) .GT. UB(I))) THEN
                    XP(I) = LB(I) + (UB(I) - LB(I))*RANMAR()
                    LNOBDS = LNOBDS + 1
                    NOBDS = NOBDS + 1
                    IF(IPRINT .GE. 3) CALL PRT3(MAX,N,XP,X,FP,F)
                  END IF
110            CONTINUE

C  Evaluate the function with the trial point XP and return as FP.
               CALL FCN(N,XP,FP)
               IF(.NOT. MAX) FP = -FP
               NFCNEV = NFCNEV + 1
               IF(IPRINT .GE. 3) CALL PRT4(MAX,N,XP,X,FP,F)

C  If too many function evaluations occur, terminate the algorithm.
               IF(NFCNEV .GE. MAXEVL) THEN
                  CALL PRT5
                  IF (.NOT. MAX) FOPT = -FOPT
                  IER = 1
                  RETURN
               END IF

C  Accept the new point if the function value increases.
               IF(FP .GE. F) THEN
                  IF(IPRINT .GE. 3) THEN
                 WRITE(7,'(''  POINT ACCEPTED'')')
                  END IF
                  DO 120, I = 1, N
                     X(I) = XP(I)
120               CONTINUE
                  F = FP
                  NACC = NACC + 1
                  NACP(H) = NACP(H) + 1
                  NUP = NUP + 1

C  If greater than any other point, record as new optimum.
                  IF (FP .GT. FOPT) THEN
                  IF(IPRINT .GE. 3) THEN
                 WRITE(7,'(''  NEW OPTIMUM'')')
                  END IF
```

```
                     DO 130, I = 1, N
                        XOPT(I) = XP(I)
130                  CONTINUE
                     FOPT = FP
                     NNEW = NNEW + 1
                  END IF

C  If the point is lower, use the Metropolis criteria to decide on
C  acceptance or rejection.
               ELSE
                  P = EXPREP((FP - F)/T)
                  PP = RANMAR()
                  IF (PP .LT. P) THEN
                     IF(IPRINT .GE. 3) CALL PRT6(MAX)
                     DO 140, I = 1, N
                        X(I) = XP(I)
140                  CONTINUE
                     F = FP
                     NACC = NACC + 1
                     NACP(H) = NACP(H) + 1
                     NDOWN = NDOWN + 1
                  ELSE
                     NREJ = NREJ + 1
                     IF(IPRINT .GE. 3) CALL PRT7(MAX)
                  END IF
               END IF

200         CONTINUE
300      CONTINUE

C  Adjust VM so that approximately half of all evaluations are accepted.
         DO 310, I = 1, N
            RATIO = DFLOAT(NACP(I)) /DFLOAT(NS)
            IF (RATIO .GT. .6) THEN
               VM(I) = VM(I)*(1. + C(I)*(RATIO - .6)/.4)
            ELSE IF (RATIO .LT. .4) THEN
               VM(I) = VM(I)/(1. + C(I)*((.4 - RATIO)/.4))
            END IF
            IF (VM(I) .GT. (UB(I)-LB(I))) THEN
               VM(I) = UB(I) - LB(I)
            END IF
310      CONTINUE

         IF(IPRINT .GE. 2) THEN
            CALL PRT8(N,VM,XOPT,X)
         END IF

         DO 320, I = 1, N
            NACP(I) = 0
320      CONTINUE

400   CONTINUE

      IF(IPRINT .GE. 1) THEN
         CALL PRT9(MAX,N,T,XOPT,VM,FOPT,NUP,NDOWN,NREJ,LNOBDS,NNEW)
      END IF

C  Check termination criteria.
      QUIT = .FALSE.
      FSTAR(1) = F
      IF ((FOPT - FSTAR(1)) .LE. EPS) QUIT = .TRUE.
      DO 410, I = 1, NEPS
         IF (ABS(F - FSTAR(I)) .GT. EPS) QUIT = .FALSE.
410   CONTINUE
```

```
C  Terminate SA if appropriate.
      IF (QUIT) THEN
         DO 420, I = 1, N
            X(I) = XOPT(I)
420      CONTINUE
         IER = 0
         IF (.NOT. MAX) FOPT = -FOPT
       IF(IPRINT .GE. 1) CALL PRT10
         RETURN
      END IF

C  If termination criteria is not met, prepare for another loop.
      T = RT*T
      DO 430, I = NEPS, 2, -1
         FSTAR(I) = FSTAR(I-1)
430   CONTINUE
      F = FOPT
      DO 440, I = 1, N
         X(I) = XOPT(I)
440   CONTINUE

C  Loop again.
      GO TO 100

      END

      FUNCTION  EXPREP(RDUM)
C  This function replaces exp to avoid under- and overflows and is
C  designed for IBM 370 type machines. It may be necessary to modify
C  it for other machines. Note that the maximum and minimum values of
C  EXPREP are such that they has no effect on the algorithm.

      DOUBLE PRECISION  RDUM, EXPREP

      IF (RDUM .GT. 174.) THEN
         EXPREP = 3.69D+75
      ELSE IF (RDUM .LT. -180.) THEN
         EXPREP = 0.0
      ELSE
         EXPREP = EXP(RDUM)
      END IF

      RETURN
      END

      subroutine RMARIN(IJ,KL)
C  This subroutine and the next function generate random numbers. See
C  the comments for SA for more information. The only changes from the
C  orginal code is that (1) the test to make sure that RMARIN runs first
C  was taken out since SA assures that this is done (this test didn't
C  compile under IBM's VS Fortran) and (2) typing ivec as integer was
C  taken out since ivec isn't used. With these exceptions, all following
C  lines are original.

C This is the initialization routine for the random number generator
C     RANMAR()
C NOTE: The seed variables can have values between:    0 <= IJ <= 31328
C                                                      0 <= KL <= 30081
      real U(97), C, CD, CM
      integer I97, J97
      common /raset1/ U, C, CD, CM, I97, J97
      if( IJ .lt. 0  .or.  IJ .gt. 31328  .or.
     *    KL .lt. 0  .or.  KL .gt. 30081 ) then
```

```
           print '(A)', ' The first random number seed must have a value
     *between 0 and 31328'
           print '(A)',' The second seed must have a value between 0 and
     *30081'
              stop
      endif
      i = mod(IJ/177, 177) + 2
      j = mod(IJ    , 177) + 2
      k = mod(KL/169, 178) + 1
      l = mod(KL,     169)
      do 2 ii = 1, 97
         s = 0.0
         t = 0.5
         do 3 jj = 1, 24
            m = mod(mod(i*j, 179)*k, 179)
            i = j
            j = k
            k = m
            l = mod(53*l+1, 169)
            if (mod(l*m, 64) .ge. 32) then
               s = s + t
            endif
            t = 0.5 * t
3        continue
         U(ii) = s
2     continue
      C = 362436.0 / 16777216.0
      CD = 7654321.0 / 16777216.0
      CM = 16777213.0 /16777216.0
      I97 = 97
      J97 = 33
      return
      end

      function ranmar()
      real U(97), C, CD, CM
      integer I97, J97
      common /raset1/ U, C, CD, CM, I97, J97
         uni = U(I97) - U(J97)
         if( uni .lt. 0.0 ) uni = uni + 1.0
         U(I97) = uni
         I97 = I97 - 1
         if(I97 .eq. 0) I97 = 97
         J97 = J97 - 1
         if(J97 .eq. 0) J97 = 97
         C = C - CD
         if( C .lt. 0.0 ) C = C + CM
         uni = uni - C
         if( uni .lt. 0.0 ) uni = uni + 1.0
         RANMAR = uni
      return
      END

      SUBROUTINE PRT1
C  This subroutine prints intermediate output, as does PRT2 through
C  PRT10. Note that if SA is minimizing the function, the sign of the
C  function value and the directions (up/down) are reversed in all
C  output to correspond with the actual function optimization. This
C  correction is because SA was written to maximize functions and
C  it minimizes by maximizing the negative a function.

      WRITE(7,'(/,''  THE STARTING VALUE (X) IS OUTSIDE THE BOUNDS ''
     1          /,''  (LB AND UB). EXECUTION TERMINATED WITHOUT ANY''
     2          /,''  OPTIMIZATION. RESPECIFY X, UB OR LB SO THAT  ''
```

```
3          /,''  LB(I) .LT. X(I) .LT. UB(I), I = 1, N. ''/)')

 RETURN
 END

 SUBROUTINE PRT2(MAX,N,X,F)

 DOUBLE PRECISION  X(*), F
 INTEGER  N
 LOGICAL  MAX

 WRITE(7,'(''  '')')
 CALL PRTVEC(X,N,'INITIAL X')
 IF (MAX) THEN
  WRITE(7,'(''  INITIAL F: '',/, G25.18)') F
 ELSE
  WRITE(7,'(''  INITIAL F: '',/, G25.18)') -F
 END IF

 RETURN
 END

 SUBROUTINE PRT3(MAX,N,XP,X,FP,F)

 DOUBLE PRECISION  XP(*), X(*), FP, F
 INTEGER  N
 LOGICAL  MAX

 WRITE(7,'(''  '')')
 CALL PRTVEC(X,N,'CURRENT X')
 IF (MAX) THEN
  WRITE(7,'(''  CURRENT F: '',G25.18)') F
 ELSE
  WRITE(7,'(''  CURRENT F: '',G25.18)') -F
 END IF
 CALL PRTVEC(XP,N,'TRIAL X')
 WRITE(7,'(''  POINT REJECTED SINCE OUT OF BOUNDS'')')
 FP=FP
 RETURN
 END

 SUBROUTINE PRT4(MAX,N,XP,X,FP,F)

 DOUBLE PRECISION  XP(*), X(*), FP, F
 INTEGER  N
 LOGICAL  MAX

 WRITE(7,'(''  '')')
 CALL PRTVEC(X,N,'CURRENT X')
 IF (MAX) THEN
  WRITE(7,'(''  CURRENT F: '',G25.18)') F
  WRITE(*,'(''  CURRENT F: '',G25.18)') F
    CALL PRTVEC(XP,N,'TRIAL X')
  WRITE(7,'(''  RESULTING F: '',G25.18)') FP
  WRITE(*,'(''  RESULTING F: '',G25.18)') FP
 ELSE
  WRITE(7,'(''  CURRENT F: '',G25.18)') -F
  WRITE(*,'(''  CURRENT F: '',G25.18)') -F
    CALL PRTVEC(XP,N,'TRIAL X')
  WRITE(7,'(''  RESULTING F: '',G25.18)') -FP
  WRITE(*,'(''  RESULTING F: '',G25.18)') -FP
 END IF

 RETURN
```

```
      END

      SUBROUTINE PRT5

      WRITE(7,'(/,''  TOO MANY FUNCTION EVALUATIONS; CONSIDER ''
     1          /,''  INCREASING MAXEVL OR EPS, OR DECREASING ''
     2          /,''  NT OR RT. THESE RESULTS ARE LIKELY TO BE ''
     3          /,''  POOR.'',/)')

      RETURN
      END

      SUBROUTINE PRT6(MAX)

      LOGICAL  MAX

      IF (MAX) THEN
       WRITE(7,'(''  THOUGH LOWER, POINT ACCEPTED'')')
      ELSE
       WRITE(7,'(''  THOUGH HIGHER, POINT ACCEPTED'')')
      END IF

      RETURN
      END

      SUBROUTINE PRT7(MAX)

      LOGICAL  MAX

      IF (MAX) THEN
       WRITE(7,'(''  LOWER POINT REJECTED'')')
      ELSE
       WRITE(7,'(''  HIGHER POINT REJECTED'')')
      END IF

      RETURN
      END

      SUBROUTINE PRT8(N,VM,XOPT,X)

      DOUBLE PRECISION  VM(*), XOPT(*), X(*)
      INTEGER  N

      WRITE(7,'(/,
     1  '' INTERMEDIATE RESULTS AFTER STEP LENGTH ADJUSTMENT'',/)')
      CALL PRTVEC(VM,N,'NEW STEP LENGTH (VM)')
      CALL PRTVEC(XOPT,N,'CURRENT OPTIMAL X')
      CALL PRTVEC(X,N,'CURRENT X')
      WRITE(7,'('' '')')

      RETURN
      END

      SUBROUTINE PRT9(MAX,N,T,XOPT,VM,FOPT,NUP,NDOWN,NREJ,LNOBDS,NNEW)

      DOUBLE PRECISION  XOPT(*), VM(*), T, FOPT
      INTEGER  N, NUP, NDOWN, NREJ, LNOBDS, NNEW, TOTMOV
      LOGICAL  MAX

      TOTMOV = NUP + NDOWN + NREJ

      WRITE(7,'(/,
     1  '' INTERMEDIATE RESULTS BEFORE NEXT TEMPERATURE REDUCTION'',/)')
      WRITE(7,'(''  CURRENT TEMPERATURE:            '',G12.5)') T
```

```
      IF (MAX) THEN
       WRITE(7,'(''  MAX FUNCTION VALUE SO FAR:  '',G25.18)') FOPT
       WRITE(7,'(''  TOTAL MOVES:                '',I8)') TOTMOV
       WRITE(7,'(''     UPHILL:                  '',I8)') NUP
       WRITE(7,'(''     ACCEPTED DOWNHILL:       '',I8)') NDOWN
       WRITE(7,'(''     REJECTED DOWNHILL:       '',I8)') NREJ
       WRITE(7,'(''  OUT OF BOUNDS TRIALS:       '',I8)') LNOBDS
       WRITE(7,'(''  NEW MAXIMA THIS TEMPERATURE:'',I8)') NNEW
      ELSE
       WRITE(7,'(''  MIN FUNCTION VALUE SO FAR:  '',G25.18)') -FOPT
       WRITE(7,'(''  TOTAL MOVES:                '',I8)') TOTMOV
       WRITE(7,'(''     DOWNHILL:                '',I8)')  NUP
       WRITE(7,'(''     ACCEPTED UPHILL:         '',I8)')  NDOWN
       WRITE(7,'(''     REJECTED UPHILL:         '',I8)')  NREJ
       WRITE(7,'(''  TRIALS OUT OF BOUNDS:       '',I8)')  LNOBDS
       WRITE(7,'(''  NEW MINIMA THIS TEMPERATURE:'',I8)')  NNEW
      END IF
      CALL PRTVEC(XOPT,N,'CURRENT OPTIMAL X')
      CALL PRTVEC(VM,N,'STEP LENGTH (VM)')
      WRITE(7,'('' '')')

      RETURN
      END

      SUBROUTINE PRT10

      WRITE(7,'(/,''  SA ACHIEVED TERMINATION CRITERIA. IER = 0. '',/)')

      RETURN
      END

      SUBROUTINE PRTVEC(VECTOR,NCOLS,NAME)
C  This subroutine prints the double precision vector named VECTOR.
C  Elements 1 thru NCOLS will be printed. NAME is a character variable
C  that describes VECTOR. Note that if NAME is given in the call to
C  PRTVEC, it must be enclosed in quotes. If there are more than 10
C  elements in VECTOR, 10 elements will be printed on each line.

      INTEGER NCOLS
      DOUBLE PRECISION VECTOR(NCOLS)
      CHARACTER *(*) NAME

      WRITE(7,1001) NAME

      IF (NCOLS .GT. 10) THEN
         LINES = INT(NCOLS/10.)

         DO 100, I = 1, LINES
            LL = 10*(I - 1)
          WRITE(7,1000) (VECTOR(J),J = 1+LL, 10+LL)
  100    CONTINUE

       WRITE(7,1000) (VECTOR(J),J = 11+LL, NCOLS)
      ELSE
       WRITE(7,1000) (VECTOR(J),J = 1, NCOLS)
      END IF

 1000 FORMAT( 10(G12.5,1X))
 1001 FORMAT(/,25X,A)

      RETURN
      END
```