32

The Impact of Evaporatively Cooled Turbine Blades on Gas Turbine Performance

by

Federico Martinez-Tamayo

B.S., Aeronautical and Astronautical Engineering
Massachusetts Institute of Technology, 1990

Submitted to the Department of Aeronautical and Astronautical Engineering
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in
Aeronautical and Astronautical Engineering

at the
Massachusetts Institute of Technology
May 1995

Signature of Author.. [illegible]
Department of Aeronautical and Astronautical Engineering
February 17, 1995

Certified by................................ [illegible] ....
Professor Jack L. Kerrebrock
Thesis Supervisor

Accepted by .... [illegible] ....
Professor Harold Y. Wachman
Chairman, Department Graduate Committee

Abstract

Cycle analyses were performed for aircraft turbofan and ground based gas turbine engines in order to determine the effect that high cooling has on engine performance. Such high cooling might be the result of using heat pipe technology (evaporative cooling) to cool the turbine. Both cooled and uncooled engines were modeled. For the cooled aircraft engines, two types of cooling air ducting schemes were modeled. In the first scheme, cooling air is bled into the turbine at the point where it is used for cooling, and in the second scheme, the cooling air is injected into the bypass duct after it is used to cool the turbine. In addition, film cooling was modeled for comparison with the evaporatively cooled engines. For the ground based engines, a regenerated low pressure engine was modeled.

Two types of tests were conducted: in one, the turbine inlet temperature was varied from 1250 K to 2500 K while the turbine wall temperature was held constant at 1250 K, and in the second, the turbine inlet temperature was held constant at 2500 K while the turbine wall temperature was decreased from 1250 K. An increase in specific impulse of approximately 24% resulted from raising the turbine inlet temperature from 1650 K to 2500 K for the cooled high bypass engines at takeoff conditions. Similar benefits of increasing the turbine inlet temperature were found for the low bypass engines and for the high bypass engines at cruise conditions. When the turbine wall temperature was decreased, the performance of the cooled engines with coolant bled into to bypass duct in all cases degraded significantly less than the engines with coolant injected into the turbine. The regenerated ground based engine with cooling increased in efficiency by almost 15% when the turbine inlet temperature was increased from 1650 K to 2500 K, with a penalty of approximately 4.2% due to cooling at 2500 K.

Table of Contents

List of Figures

1.0 Introduction

The efficiency of gas turbines has been steadily improving as technological limits are pushed back. Primary among these is the limit on the turbine inlet temperature imposed by the turbine materials. Material technology advances, but slowly and at a high cost. Cooling of the turbine blades has provided additional help, but there are inherent limits in the cooling methods in use today. The size and shape of the blades limits the capacity of internal cooling systems, and film cooling becomes inefficient at high gas temperatures.

Evaporatively cooled turbines, which work on the same principles as standard heat pipes, hold great promise for improving gas turbine efficiency. In a heat pipe, inflowing heat vaporizes a liquid and is carried by the vapor to a condenser, where it is rejected. The liquid flows back along the walls to the evaporating region. Typically, a heat pipe has the advantages of a very high heat flux and a nearly constant temperature. In a turbine, these properties would translate to higher has temperature and greater efficiency. In addition, they might make it feasible to lower the wall temperature, allowing more flexibility in the choice of materials.

A functional evaporatively cooled turbine has yet to be demonstrated. Design concepts have been proposed, but the details of implementation have not been worked out. Ironing out these details will be a difficult and expensive task. Therefore, before embarking on an in-depth analysis and design study of an evaporatively cooled turbine, we must first ask whether or not such a turbine is desirable. Cycle analyses of engines with and without high cooling can help to answer this question from a thermodynamic standpoint. Specifically, they can tell us whether the penalties involved with extremely aggressive cooling of the turbine blades (energy extraction from the turbine flow, cooling air taken from the flowpath in the compressor) are offset by the benefits of higher turbine inlet temperature and lower blade temperature. This thermodynamic cycle analysis will be the focus of this report.

Two basic engine configurations were analyzed: a turbofan aircraft engine, and a ground-based engine with regeneration. The aircraft engine was modeled with three different ducting schemes for the cooling air. Two of these make use of evaporative cooling capabilities while the third is a standard film cooling scheme that has been included for comparison.

2.0 Turbofan Aircraft Engine

A diagram of a turbofan engine, including the station numbering used in this report, is given in Figure 1, "Turbofan Engine." In order to cool the turbine, a fraction ε of the mass flow entering the core is bled from some point in the compressor. Therefore, in an aircraft engine, the issue of where to exhaust this air after it has absorbed heat in the turbine becomes a critical one. Two basic ducting schemes were modeled. In one, shown schematically in Figure 2, "Cooling Flow Injected Into Turbine," the cooling air is injected directly into the main flow in the turbine. The air required to cool each stage is injected at that stage so that it can be expanded in the remaining stages and produce work. A more unusual scheme, shown in Figure 3, "Cooling Flow Injected Into Bypass Duct," involves routing the cooling air from the turbine to the bypass duct to be injected just downstream of the fan and expanded in the fan nozzle. This latter approach has both advantages and disadvantages over the turbine injection scheme. On the one hand, the pressure requirements on the cooling air are much lower when it is dumped into the relatively low pressure bypass duct. The temperature of the cooling air is therefore lower and less air is required. On the other hand, the cooling air must be ducted from the turbine through the rotating hub to the inertial fan nozzle. At high turbine inlet temperatures, this volume of cooling air can become large.

This section will explain the details of the models used to analyze the cooled aircraft engines.

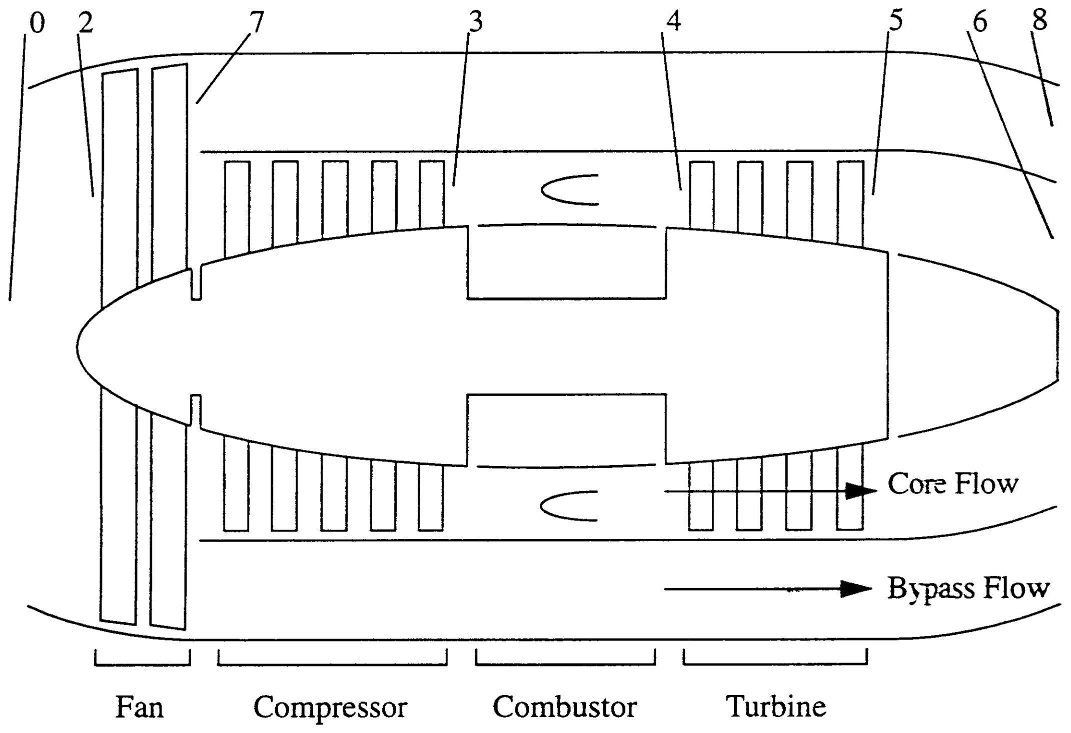

FIGURE 1. Turbofan Engine

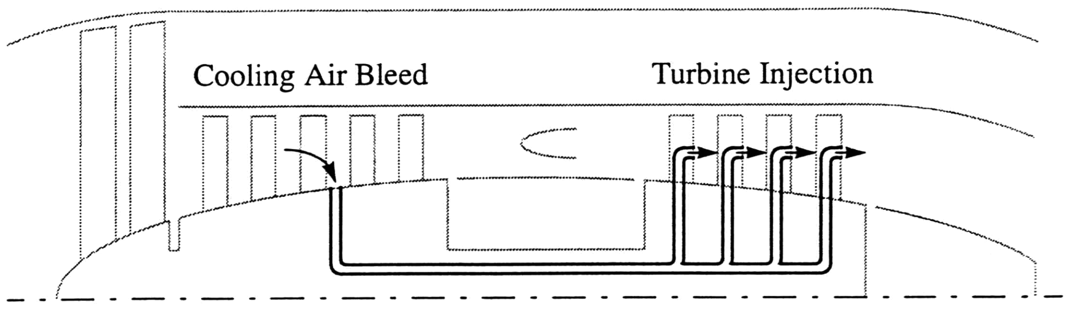

FIGURE 2. Cooling Flow Injected Into Turbine

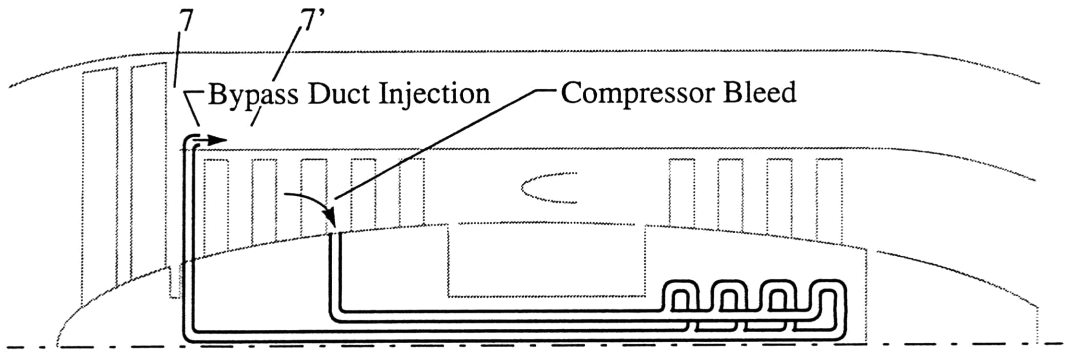

FIGURE 3. Cooling Flow Injected Into Bypass Duct

2.1 Thrust and Specific Impulse

The fan thrust of a turbofan engine is the sum of the rate of change in momentum in the bypass flow and the difference between exit pressure and atmospheric pressure acting on the fan exit area (in the following expressions, ε_f is equal to the mass fraction of the cooling air ε if it is injected into the fan; if the cooling air is injected into the turbine, $\varepsilon_f = 0$):

$$F_8 = (\alpha + \varepsilon_f)\dot{m}u_8 - \alpha\dot{m}u_0 + A_8(p_8 - p_0), \tag{1}$$

where α is the bypass ratio. The dimensionless fan thrust is then

$$\frac{F_8}{\dot{m}a_0} = (\alpha + \varepsilon_f)\frac{u_8}{a_0} - \alpha\frac{u_8}{a_0} + \frac{A_8 p_8}{\dot{m}a_0}\left(1 - \frac{p_0}{p_8}\right). \tag{2}$$

The fan exit area, A_8, can be obtained from an expression for the mass flow rate at the exit of the fan duct,

$$\dot{m}\,(\alpha+\varepsilon_f) = \frac{p_8}{RT_8}u_8A_8 . \tag{3}$$

Equation (2) can be rewritten with A_8 taken from Equation (3) to give the following expression for the fan thrust

$$\frac{F_8}{\dot{m}a_0} = (\alpha+\varepsilon_f)\frac{u_8}{a_0} - \alpha\frac{u_0}{a_0} + \frac{\alpha+\varepsilon_f}{\gamma_c}\frac{T_8 a_0}{T_0 u_8}\left(1-\frac{p_o}{p_8}\right) . \tag{4}$$

Similarly, the core thrust is found to be

$$\frac{F_6}{\dot{m}a_0} = (1+f-\varepsilon_f)\frac{u_6}{a_0} - \frac{u_0}{a_0} + \frac{1+f-\varepsilon_f}{\gamma_c}\frac{T_6 a_0}{T_0 u_6}\left(1-\frac{p_o}{p_6}\right) , \tag{5}$$

where f is the fuel-air ratio. The total thrust F is simply the sum of F_6 and F_8, or per unit total airflow,

$$\frac{F}{\dot{m}a_0(1+\alpha)} = \frac{1}{1+\alpha}\left(\frac{F_6}{\dot{m}a_0} + \frac{F_8}{\dot{m}a_0}\right) . \tag{6}$$

No real simplification results from combining Equation (4) and Equation (5) into Equation (6), so they will be left in their current form.

The specific impulse follows from Equation (6), since $I = F/(g\dot{m}_f)$:

$$I = \frac{a_0(1+\alpha)}{g}\left(\frac{F}{\dot{m}a_0(1+\alpha)}\right)\frac{1}{f} . \tag{7}$$

From the thrust equations, we see that for the case where the nozzles are ideally expanded ($p_6 = p_8 = p_0$) and the exit velocities are equalized, as may be true of an engine tailored for high efficiency, a transfer of mass from the core to the bypass flow has no direct effect on the thrust. It may, however, have an effect through a change in the optimum bypass ratio necessary to enforce the above conditions.

In order to calculate the thrust and specific impulse using Equation (6) and Equation (7), we must know, in addition to the flight conditions and the bypass ratio, expressions for f, ε, and the exit conditions (the velocities u_6/u_0 and u_8/u_0, the temperatures T_6/T_0 and T_8/T_0, and the pressures p_6/p_0 and p_8/p_0). We will derive expressions for these quantities next.

2.2 Exit Temperatures and Velocities

Using pressure and temperature balances through the engine from inlet to exhaust, we can find T_6/T_0, T_8/T_0, u_6/u_0, u_8/u_0, M_6 and M_0. From $T_{t_6} = T_6\left(1+\frac{\gamma_c-1}{2}M_6^2\right)$ we have

$$\frac{T_6}{T_0} = \frac{\theta_t \tau_t}{1 + \frac{\gamma_c - 1}{2} M_6^2} . \tag{8}$$

Likewise, for the fan duct, we have

$$\frac{T_8}{T_0} = \frac{\theta_0 \tau_f \tau_{cf}}{1 + \frac{\gamma_c - 1}{2} M_8^2} , \tag{9}$$

where $\tau_{cf} = T_{t_8}/T_{t_7}$. This variable is added to the customarily used set of temperature ratios to take into account the effect of adding cooling flow between the fan and the exhaust (see Figure 3, "Cooling Flow Injected Into Bypass Duct"). Similarly, $\pi_{cf} = p_{t_8}/p_{t_7}$. There are two cases to consider now. The exhaust nozzles may be choked, or they may be ideally expanded to atmospheric pressure. In the latter case, we can find the Mach numbers M_6^2 and M_8^2 with a pressure accounting through the core and the fan ducts, respectively, using the isentropic relation $P_t = P\left(1 + \frac{\gamma_c - 1}{2} M^2\right)^{\gamma/(\gamma - 1)}$,

$$M_6^2 = \frac{2}{\gamma_t - 1}\left[\left(\frac{P_0}{P_6}\delta_0 \pi_d \pi_b \pi_c \pi_t\right)^{\frac{\gamma_t - 1}{\gamma_t}} - 1\right] \tag{10}$$

$$M_8^2 = \frac{2}{\gamma_c - 1}\left[\left(\frac{P_0}{P_8}\delta_0 \pi_d \pi_f \pi_{cf}\right)^{\frac{\gamma_c - 1}{\gamma_c}} - 1\right] , \tag{11}$$

where $p_0/p_6 = p_0/p_8 = 1$. If the exhaust nozzles are choked, Equation (10) and Equation (11) can be used with $M_6 = M_8 = 1$ to find p_0/p_6 and p_0/p_8. Now, once we have the individual pressure and temperature ratios, we can find u_6/u_0 and u_8/u_0 with the definition of Mach number, resulting in

$$\frac{u_6}{a_0} = M_6 \sqrt{\frac{\gamma_t R_t T_6}{\gamma_c R_c T_0}} \tag{12}$$

$$\frac{u_8}{a_0} = M_8 \sqrt{\frac{T_8}{T_0}} . \tag{13}$$

If we take θ_t, M_0, π_d, and π_b as given, then to know the exit velocities, we must know the compressor pressure ratio π_c, the fan conditions τ_f, π_f, τ_{cf} and π_{cf}, and the turbine conditions τ_t and π_t. In addition, the thrust equations require f, ε and α.

2.3 Fuel-Air Ratio

The fuel-air ratio f can be found by balancing the energy in the flow at the outlet of the combustor with the sum of the combustor inlet flow energy and the fuel energy

$$c_{p_t}(\dot{m}+\dot{m}_f-\dot{m}_\varepsilon)\,T_{t_4} = c_{p_c}(\dot{m}-\dot{m}_\varepsilon)\,T_{t_3}+\eta_b\dot{m}_f h, \tag{14}$$

which can be rearranged to give

$$f = \frac{(1-\varepsilon)\left(c_{p_t}T_0\theta_t - c_{p_c}T_0\theta_0\tau_c\right)}{h\eta_b - c_{p_t}T_0\theta_t}. \tag{15}$$

We assume in this analysis that θ_t is fixed by the turbine materials and cooling capability.

2.4 Compressor Characterization

We assume throughout these calculations that the compressor temperature ratio τ_c is that which gives maximum thrust ($\tau_c = \sqrt{\theta_t/\theta_0}$). Given a polytropic efficiency η_{poly}, we can find the pressure ratio π_c

$$\pi_c = \tau_c^{\frac{\gamma_c\eta_{\text{poly}}}{\gamma_c-1}}. \tag{16}$$

2.5 Effect on Fan Performance of Cooling Air Injected Into Fan

If the cooling flow is injected into the fan duct at station 7 with a total temperature $T_{t_{cf}}$ and total pressure $P_{t_{cf}}$, then an energy balance across the fan nozzle gives $\tau_{cf} = T_{t_8}/T_{t_7}$:

$$c_{p_c}(\alpha\dot{m}+\varepsilon_f\dot{m})\,T_{t_8} = c_{p_c}\alpha\dot{m}T_{t_7} + c_{p_c}\dot{m}\varepsilon_f T_{t_{cf}} \tag{17}$$

$$\tau_{cf} = \frac{\alpha + \left(\dfrac{T_{t_{cf}}}{T_0\theta_0\tau_f}\right)\varepsilon_f}{\alpha+\varepsilon_f}, \tag{18}$$

where $T_{t_{cf}}$ is the total temperature of the cooling air at the point of injection into the fan. We can find the effect of the cooling flow on the bypass total pressure by first assuming a value for the bypass Mach number at the point of cooling flow injection, which can be set by the geometry. With the assumption that the pressure of the cooling flow is equalized with the pressure of the bypass flow ($P_{cf} = P_7$), the isentropic total pressure relations give us the Mach number of the cooling air at the injection point:

$$M_{cf} = \sqrt{\frac{2}{\gamma-1}\left[\left(\frac{P_{t_{cf}}}{P_{t_7}}\right)^{\frac{\gamma-1}{\gamma}}\left(1+\frac{\gamma-1}{2}M_7^2\right)-1\right]} . \tag{19}$$

The isentropic temperature relations, $T = T_t / \left(1 + 0.5\,(\gamma - 1)\,M^2\right)$, and the definition of Mach number for an ideal gas, $u = M\sqrt{\gamma RT}$, give us the temperature and velocity in the two flows to be mixed. Conservation of momentum gives us

$$u_{7'} = \frac{\dot{m}_7 u_7 + \dot{m}_{cf} u_{cf}}{\dot{m}_7 + \dot{m}_{cf}} = \frac{\alpha u_7 + \varepsilon_f u_{cf}}{\alpha + \varepsilon_f}, \tag{20}$$

where the primed quantity indicates a location just after the cooling flow has been injected, but before the nozzle. From the first law of thermodynamics, we can find the temperature of the mixed flow:

$$T_{7'} = \frac{1}{(\alpha + \varepsilon_f)\, c_{p_c}}\left[\alpha c_{p_c} T_7 + \varepsilon_f c_{p_c} T_{cf} + \alpha\frac{u_7^2}{2} + \varepsilon_f\frac{u_{cf}^2}{2} - (\alpha + \varepsilon_f)\frac{u_{7'}^2}{2}\right]. \tag{21}$$

Finally, assuming that the mixing takes place at constant pressure, and that the nozzle expansion between 7' and 8 is isentropic, we can find the pressure ratio $\pi_{cf} = P_{t_8}/P_{t_7}$:

$$\pi_{cf} = \frac{p_7}{p_{t_7}}\left(1 + \frac{\gamma-1}{2}\frac{u'^2_7}{\gamma R T_7}\right)^{\frac{\gamma}{\gamma-1}} . \tag{22}$$

2.6 Turbine Characterization

The analysis of the preceding sections, with the exception of Section 2.5, Effect on Fan Performance of Cooling Air Injected Into Fan, can be applied either to the case where cooling air is injected into the flow at the fan, or into the turbine directly from the blade which it is cooling, as shown in Figure 4, "Cooling Injected Into Turbine Core Flow," noting the different values taken on by ε_f in the two cases. The primary modeling differences resulting from the two ducting schemes are in the treatment of the turbine itself. Without the addition of mass in the turbine, the problem could be approached by analyzing a change in flow and cooling properties at one stage, and approximating the accumulation of this effect through the turbine with integrals.[1] In general, it will be easier to use a more computational approach. That is, we begin at the inlet of the turbine with known (or assumed) conditions and proceed stage by stage, allowing the effects of one stage to cascade into the next. The disadvantage of this approach over that of the integral approximations is that the effects of changing any parameter are more obscure.

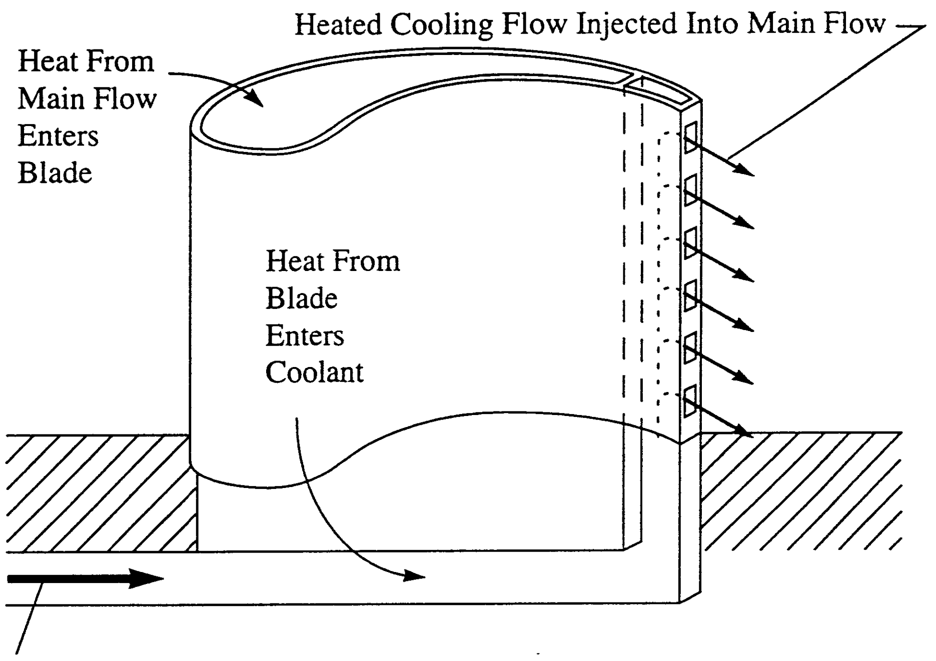
Heated Cooling Flow Injected Into Main Flow
Heat From Main Flow Enters Blade
Heat From Blade Enters Coolant

FIGURE 4. Cooling Injected Into Turbine Core Flow

In addition to the two evaporative cooling schemes, a film cooled turbine is also modeled.

2.6.1 Energy and Mass Flows in the Turbine

An overview of the operation of a turbine can be given by diagrams which track the flow of mass and energy. Figure 5, "Energy Flow Diagram," shows schematically the energy flows for one stage (stage i) of a turbine, which for the purposes of this analysis consists of one stator cascade followed by one rotor cascade. Energy $(\dot{Q}_{s,\,i})$ flows into a stator of a stage with the gas. Some of it is removed through the walls of the blades as they are cooled $(\dot{Q}_{c,\,s,\,i})$. Energy may also be added to the main flow $(\dot{Q}_{a,\,s,\,i})$ if the cooling air is bled into the turbine. This last energy flux would be equal to that of the energy removed by cooling plus that of the cooling air itself, which has been given some energy in the compressor. Similarly, in the rotor we have the energy flux through the walls due to cooling $(\dot{Q}_{c,\,r,\,i})$ and the energy added back to the flow with the cooling air $(\dot{Q}_{a,\,r,\,i})$. In addition, energy is removed from the flow as the flow does work on the rotor blades. This quantity is denoted $\dot{W}_i$.

1. This approach will be shown in an appendix, since at first glance it reveals more about the effect of varying operating conditions on the engine.

Figure 6, "Mass Flow Diagram," shows a similar diagram for mass flows. Mass primarily flows through the stages of the turbine ($\dot{m}_{s,i}$ and $\dot{m}_{r,i}$), but some may be added at each cascade ($\dot{m}_{a,s,i}$, $\dot{m}_{a,r,i}$). Another flow shown is $\dot{m}_{c,i}$ the amount of cooling air required to keep the blade at a given temperature. If the cooling air is added to the main flow in the turbine, $\dot{m}_{a,s,i} = \dot{m}_{c,s,i}$ and $\dot{m}_{a,r,i} = \dot{m}_{c,r,i}$; otherwise $\dot{m}_{a,s,i} = \dot{m}_{a,r,i} = 0$. The method of this analysis will be simply to conserve mass and energy at each cascade.

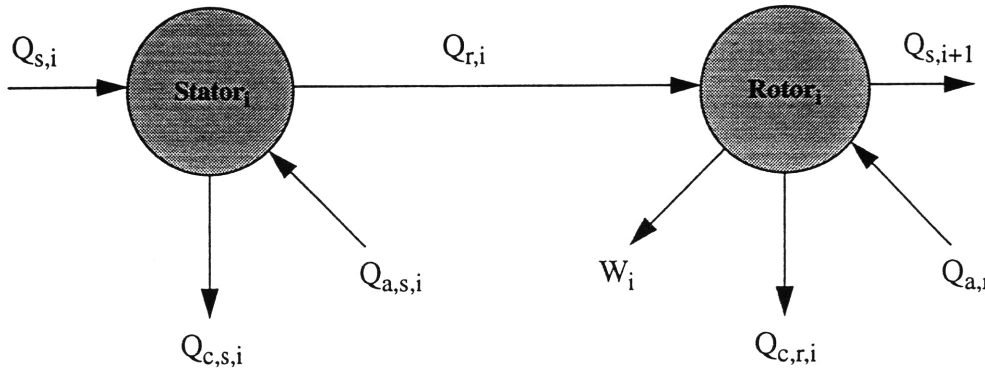

FIGURE 5. Energy Flow Diagram

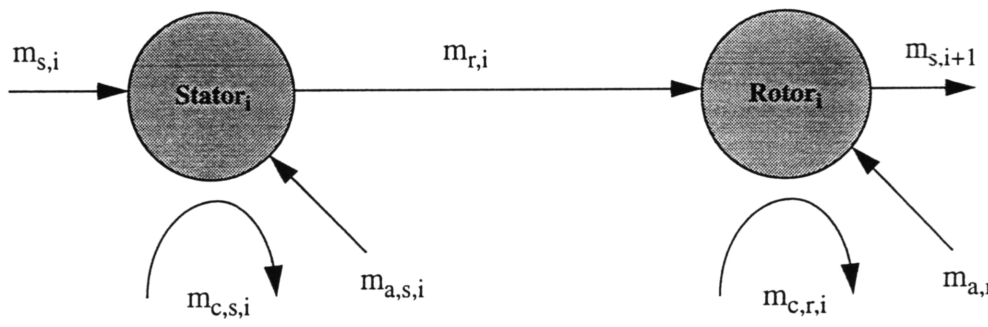

FIGURE 6. Mass Flow Diagram

2.6.2 Film Cooling Model

Film cooling reduces heat transfer from the gas to the wall by flowing a thin film of cool air along the wall as a buffer between it and the hot gas. The degree to which this film achieves its purpose can be characterized by the adiabatic film effectiveness, η_{ad}. This is defined as

$$\eta_{ad} = \frac{T_{rec} - T_{rec,f}}{T_{rec} - T_c}, \tag{23}$$

where T_{rec} is the recovery temperature of the gas in the uncooled case, $T_{rec.f}$ is the effective recovery temperature with the film, and T_c is the temperature of the cooling air.

In order to find out how much cooling air is required to achieve a given η_{ad}, we can refer to Figure 7, "Film Mass Parameter, m, vs. Adiabatic Film Effectiveness," which is adapted from Figure 6.13 in reference 1. The referenced figure is a plot of the adiabatic film effectiveness against a nondimensionalized streamwise distance from the film injection holes, x/t, and is given for various values of the mass parameter m. The parameter x/t is the distance from the holes, x, divided by the thickness, t, of a slot with the same flow area as the row of film injection holes. The mass parameter m is defined as

$$m = \frac{(\rho u)_{c,\,\mathrm{film}}}{(\rho u)_\infty}. \tag{24}$$

We choose a value of x/t of 10 as representing a useful average and plot m vs. η_{ad}, as shown in Figure 7. To these data points we fit the curve shown in the figure. From this plot, we can clearly see that at the high adiabatic film effectiveness required for high gas temperatures, the penalty in required film mass flow increases sharply.

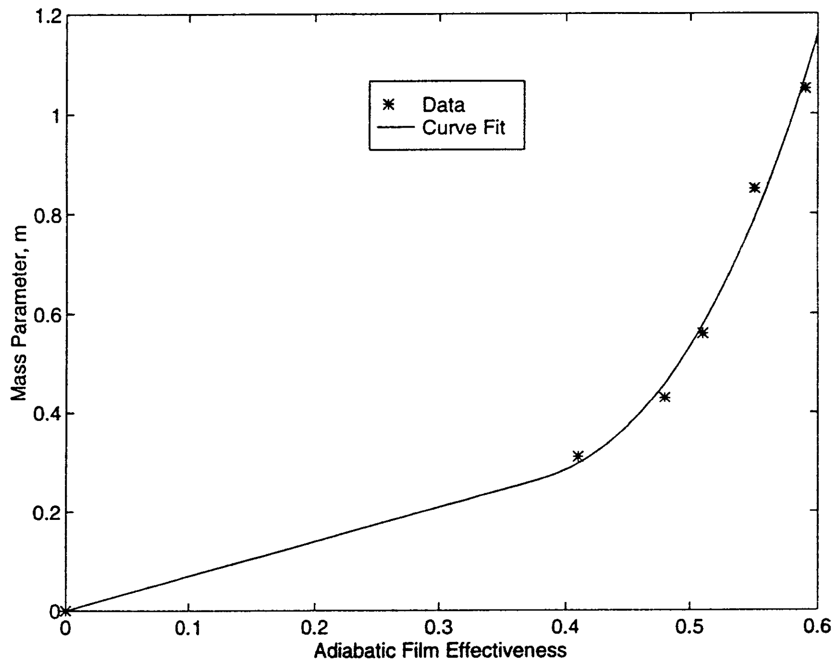

FIGURE 7. Film Mass Parameter, m, vs. Adiabatic Film Effectiveness

If, in the definition of m, $(\rho u)_\infty$ is taken to mean the undisturbed mass flux per unit area in the blade passages, then we find that the amount of cooling air required is

$$\frac{\dot{m}_{c,\,\text{film}}}{\dot{m}_0} = n\frac{mr}{x/t}, \tag{25}$$

where r is the ratio of blade surface area to cross sectional flow area, $A_{\text{blade}}/A_{\text{flow}}$, and n is the number of rows of film cooling injection holes.

For this analysis, we will assume a maximum value of η_{ad} of 0.6. If this maximum value of η_{ad} is insufficient to eliminate heat flux into the walls, this heat flux is absorbed by internal cooling air. The quantity of internal cooling air required ($\dot{m}_{c,\,\text{internal}}$) is calculated as described in Section 2.6.4, Stator Calculations, for the non-film cooled schemes, noting that the recovery temperature should in this case be approximated not by the gas total temperature, but by

$$T_{\text{rec}} \cong T_t - \eta_{\text{ad}}(T_t - T_c)\,. \tag{26}$$

2.6.3 Turbine Stage Temperature Ratio

We assume for the purpose of this section that we know the values of τ_t and of f. Appendix B describes the numerical scheme by which the actual values are arrived at. This value of τ_t does not give the work done by the turbine through $\dot{W} = c_p\dot{m}T_{t_4}(1-\tau_t)$, because this expression does not take the effects of cooling into account. Therefore, in order to balance the turbine work with that of the compressor and fan, we must find the sum of the actual work done at each turbine stage and use this value in the work balance, Equation (66).

We can estimate the number of stages required to achieve the total temperature ratio τ_t from

$$N = \frac{\ln(\tau_t)}{\ln(\tau_s)}\,. \tag{27}$$

Since this is not an integer, we take as the number of stages the closest integer value which yields a reasonable value of the stage temperature ratio τ_s. In this analysis, a reasonable value is taken to be close to 0.89[1].

We have enough information now to begin stepping through the N stages of the turbine and calculating the temperatures, pressures, amount of cooling flow required, and work done at each stage. The following calculations, shown for stage i, are done at each of the N stages.

1. As the turbine inlet temperature is varied, it should really be the blade speed, which is limited by materials, that is held constant.

2.6.4 Stator Calculations

Beginning at the stator, we find the mass flow rate of cooling air required. The heat flux from the main fluid to each blade row is approximately proportional to the difference in the recovery temperature of the fluid and the wall temperature, to the mass flow rate of the main flow, and to the ratio of blade area to flow area $A_{blade}/A_{flow}=r$. This can be seen from

$$\dot{Q}_C = \mathrm{St} r c_{p_t} \dot{m} (T_t - T_w) . \tag{28}$$

If we assume a large hub to tip radius ratio in the turbine cascades, the area ratio r can be approximated by the solidity as follows

$$r_i = \frac{A_{blade_i}}{A_{flow_i}} \approx \left(\frac{2Ncl}{Nsl}\right)_i = 2\left(\frac{c}{s}\right)_i = 2\sigma_i , \tag{29}$$

where c, s, and l are as shown in Figure 8, "Blade Dimensions."

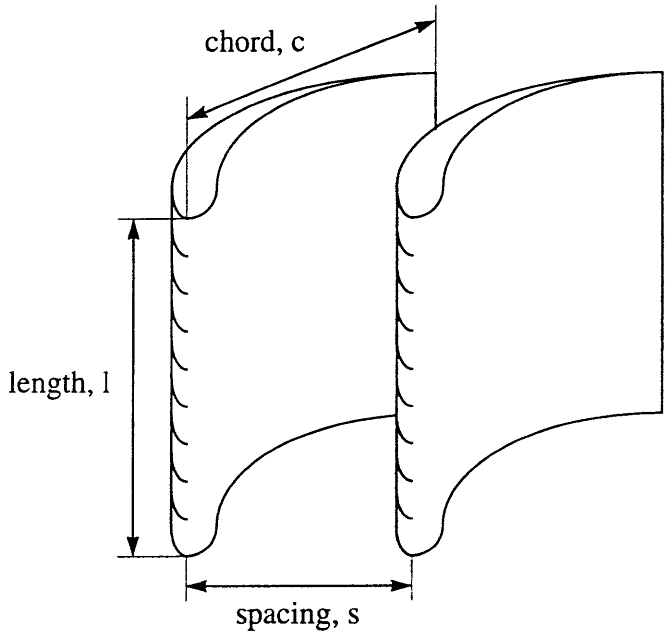

FIGURE 8. Blade Dimensions

In order to find how much coolant flow is needed, we assume that all the energy passing through the walls is eventually absorbed by the cooling air, that the wall is at approximately the same temperature at the inside and outside surfaces, and that as the cooling air leaves the heat exchanger, its temperature is equal to the wall temperature. This last assumption might be refined, since the heat flux to the cooling air would go to zero at the end of the heat exchanger, and unless a large heat transfer area is provided, the wall temperature will in reality not be achieved by the fluid. However, because this analysis is concerned with evaporatively cooled turbines, the heat exchanger for the turbine blade coolant to cooling air interface can be placed in the hub of the turbine, and therefore can be made relatively large. Given these assumptions, then, the amount of heat transferred to the cooling air will be

$$\dot{Q}_C = \dot{m}_c c_{p_c} (T_w - T_c)\ . \tag{30}$$

If we equate this flux with the flux from the main flow to the wall, we can solve for the ratio of cooling air required per unit of core flow.

$$\left(\frac{\dot{m}_c}{\dot{m}}\right)_{s,\,i} = \mathrm{St}r\left(\frac{c_{p_t}}{c_{p_c}}\right)\left(\frac{T_t - T_w}{T_w - T_c}\right). \tag{31}$$

It is emphasized that $\dot{m}$ in the above equation, as well as in all following equations where it appears either with a subscript or within a subscripted fraction, is the core mass flow at that blade row. This corresponds to $\dot{m}_s$ or $\dot{m}_r$ in Figure 6, "Mass Flow Diagram."

2.6.5 Energy and Mass Changes From Stator to Rotor

There are two variables that change and that we must keep track of as we progress through the turbine. These are the mass flow through the turbine and the total temperature, corresponding to the conserved quantities in the diagrams of Figure 6, "Mass Flow Diagram," and Figure 5, "Energy Flow Diagram," respectively. If there is no dumping of cooling air into the turbine, the ratio of mass flow through the rotor to that through the stator will be one. If there is dumping of cooling air in the turbine, this ratio is

$$\left(\frac{\dot{m}_r}{\dot{m}_s}\right)_i = 1 + \left(\frac{\dot{m}_c}{\dot{m}}\right)_{s,\,i}. \tag{32}$$

The total temperature drops by a ratio of τ_s from the inlet to the stator of one stage to the next. To find the temperature at the inlet to the rotor, T_{t_b}, we can refer to Figure 5, "Energy Flow Diagram," and balance the energy flowing in and out of the stator. Without addition of cooling air to the main flow, this results in

$$\frac{T_{t_r}}{T_{t_s}} = 1 - \mathrm{St}r\left(1 - \frac{T_w}{T_{t_s}}\right). \tag{33}$$

If the cooling air is added to the main flow, the temperature ratio becomes

$$\frac{T_{t_r}}{T_{t_s}} = \frac{1 + \left(\frac{c_{p_c}}{c_{p_t}}\right)\left(\frac{\dot{m}_c}{\dot{m}}\right)_{s,\,i}\left(\frac{T_c}{T_{t_s}}\right)}{1 + \left(\frac{\dot{m}_c}{\dot{m}}\right)_{s,\,i}}. \tag{34}$$

2.6.6 Stator Total Pressure Calculations

The changes in total pressure that the fluid undergoes in passing through a cascade can be separated according to location into two distinct groups: changes caused by cooling, frictional losses, and work (for a rotor cascade), and changes due to mixing of the cooling flow with the main flow. The first group of total pressure changes occur before the trailing edge of the blades, and the mixing effect applies after the trailing edge (in Figure 4, "Cooling Injected Into Turbine Core Flow," note that the cooling flow is injected at the trailing edge of the blades). For the configuration in which the cooling flow is injected into the bypass duct, there is no flow mixing, and only the first group applies. For the configuration in which the cooling flow is dumped directly into the turbine, we apply all the total pressure changes in the first group above to find the flow conditions at the trailing edge. Then we separately find the total pressure changes caused by the mixing of the cooling flow with the main flow.

A simplified schematic of the stages involved in the total pressure change across a cascade is given in Figure 9, "Simplified Schematic of Flow Mixing in Stator (or Rotor) Cascade," for the case of coolant injected into the turbine.

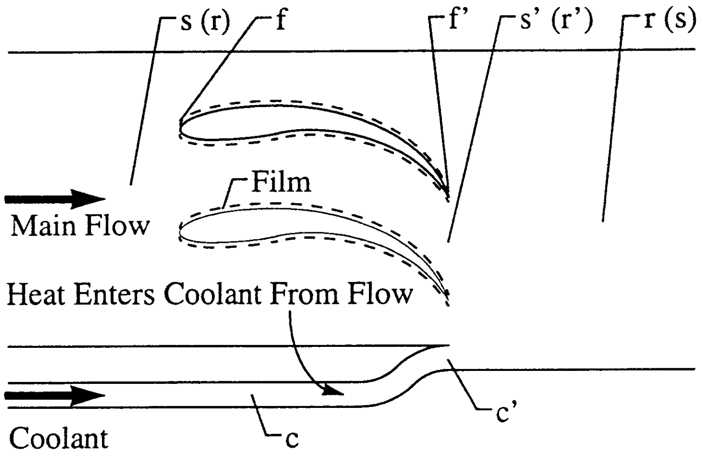

FIGURE 9. Simplified Schematic of Flow Mixing in Stator (or Rotor) Cascade

In this representation, all the cooling flow for the stage is shown entering the main channel at one point, corresponding to the trailing edge of the stator blades. There are three steps involved in the calculation of the total pressure at the inlet to the rotor cascade, labeled r in Figure 9. First the conditions at c' and s' for the cooling and main flows, respectively, are calculated separately. Then, with the assumptions that the mixing of the flows occurs at constant pressure and that momentum is conserved, the total pressure change due to the mixing is calculated.

For the case where the coolant is dumped into the bypass duct, the conditions at r are equal to those at s'.

Finally, for the film cooled turbines, the film air is introduced at f and absorbs heat from the main air as it flows along the blade, but the mixing effect on the total pressure is not considered until f', at the trailing edge of the blades.

The total pressure changes will be described first for the engines cooled by internal coolant flow, then for the film cooled engines.

The total pressure at c' can be estimated from the amount of heat transferred to the cooling flow and from an approximation to the frictional pressure loss in the flow passages between the compressor bleed point and the turbine. Starting from the general relationship

$$ds = c_p \frac{dT_t}{T_t} - R\frac{dP_t}{P_t}, \tag{35}$$

we can find the change in total pressure due to heat addition in the heat exchanger by setting $ds = c_p (dT_t/T)$, giving

$$\frac{dP_t}{P_t} = -\frac{\gamma M^2}{2}\frac{dT_t}{T_t}. \tag{36}$$

For simplicity, we assume a constant Mach number in the heat exchanger. There will also be an entropy rise in the cooling air due to frictional effects. This will be included in a parameter $\Delta p_f/P_{t_c}$, which estimates all of the frictional pressure losses in the cooling flow ducts between compressor and turbine. Combining these components, we have an approximate expression for $P_{t_{c'}}$:

$$P_{t_{c'}} = P_{t_c}\left(1 - \frac{\Delta p_f}{P_{t_c}}\right)\left(1 + \frac{c_{p_t}}{c_{p_c}}\left(\frac{T_w - T_c}{T_c}\right)\right)^{-\frac{\gamma M^2}{2}}. \tag{37}$$

The total temperature at c' is assumed to be approximately the temperature of the blade wall.

At the location s' in Figure 9, the total temperature can be found from an energy balance, which gives

$$T_{t_{s'}} = T_{t_s} - \mathrm{St}r\,(T_{t_s} - T_w)\,. \tag{38}$$

To find the total pressure change across the cascade from point s to point s', we again begin with Equation (35). The entropy change of the fluid is due to two factors: the removal of heat into the blades, and frictional losses. The entropy change due to frictional losses will be characterized by the stage efficiency, η_s, and will be applied at the rotor for the entire stage. The remaining entropy change is given by $ds = Q/T$, and we have again

$$\frac{dP_t}{P_t} = -\frac{\gamma M^2}{2}\frac{dT_t}{T_t}. \tag{39}$$

In the heat exchanger, the Mach number was assumed to be constant, but in the main flow, this assumption is no longer valid. Instead, the simplest model with the correct rough behavior is one where the Mach number in the interblade passages is a linear function of the total temperature. For example, they could both be linear functions of distance along the chord, x,

$$\begin{aligned} M &= M_2 + (M_1 - M_2)\,(1-x) \\ T_t &= T_{t_2} - (T_{t_1} - T_{t_2})\,(1-x) \end{aligned}, \tag{40}$$

or they could both be quadratic functions of x,

$$\begin{aligned} M &= M_2 + (M_1 - M_2)\,(1-x)^2 \\ T_t &= T_{t_2} - (T_{t_1} - T_{t_2})\,(1-x)^2 \end{aligned}. \tag{41}$$

In either case, integrating Equation (39) from s to s' gives the total pressure,

$$\frac{P_{t_{s'}}}{P_{t_s}} = \left(\frac{T_{t_{s'}}}{T_{t_s}}\right)^a \exp\left\{\frac{c}{2}\left[\left(T_{t_{s'}} + \frac{b}{c}\right)^2 - \left(T_{t_s} + \frac{b}{c}\right)^2\right]\right\}, \tag{42}$$

where

$$\begin{aligned} a &= -\frac{\gamma}{2}\left(M_{s'} - T_{t_{s'}}\frac{M_s - M_{s'}}{T_{t_s} - T_{t_{s'}}}\right)^2 \\ b &= -\gamma\left(M_{s'} - T_{t_{s'}}\frac{M_s - M_{s'}}{T_{t_s} - T_{t_{s'}}}\right)\left(\frac{M_s - M_{s'}}{T_{t_s} - T_{t_{s'}}}\right). \\ c &= -\frac{\gamma}{2}\left(\frac{M_s - M_{s'}}{T_{t_s} - T_{t_{s'}}}\right)^2 \end{aligned} \tag{43}$$

We now know the total temperatures and pressures of the two flows at the point of mixing. The cooling flow is assumed to be injected at the same pressure as the main flow, and this pressure is kept constant as the two streams mix. In order to fix this pressure, we assume that the flow coming out of the stator cascade is at a Mach number of one. In the mixing process between the points s' and c', and the point r, we conserve both energy,

$$T_{t_r} = \frac{T_{t_{s'}} + \varepsilon_s T_{t_{c'}}}{1 + \varepsilon_s}, \tag{44}$$

and momentum,

$$u_r = \frac{u_{s'} + \varepsilon_s u_{c'}}{1 + \varepsilon_s}, \tag{45}$$

which gives us the total pressure of the mixed stream

$$P_{t_r} = P_{t_{s'}} \left(\frac{1 + \frac{\gamma - 1}{2} \frac{u_r^2}{\gamma R \left(T_{t_r} - u_r^2 / (2c_p) \right)}}{1 + \frac{\gamma - 1}{2} M_{s'}^2} \right)^{\frac{\gamma}{\gamma - 1}} . \tag{46}$$

In the total pressure analysis of the film cooled turbine, all of the film air is assumed to begin flowing along the outer blade surface at the leading edge. Then, at the trailing edge of the blades, three flows are mixed at constant pressure instead of two: the main flow, the film, and the internal cooling flow.

As before, we need to find the flow conditions for each of the three streams at the point of mixing in order to know the conditions of the mixed flow. To find the conditions of the main flow and of the internal cooling flow, we follow the same procedure as that described above for the case without film cooling. In this case, however, the heat flux to the walls is reduced by the effectiveness of the film. This alters the total temperature at s' from that given in Equation (38)

$$T_{t_{s'}} = T_{t_s} - \frac{c_{p_c}}{c_{p_t}} \frac{\dot{m}_{c,\,\text{film}}}{\dot{m}_s} (T_{t_{f'}} - T_{t_c}) - \frac{c_{p_c}}{c_{p_t}} \frac{\dot{m}_{c,\,\text{internal}}}{\dot{m}_s} (T_{t_{c'}} - T_{t_c}) . \tag{47}$$

The portion of the film where the bulk of the heat from the main flow is absorbed is assumed to be at a constant Mach number. In this case, Equation (39) can be used to find the total pressure change in the film from f to f'.

Conservation of momentum and energy in the mixing process after the trailing edge of the blades gives us the velocity u_r and the total temperature T_{t_r},

$$u_r = \frac{u_{s'} + \frac{\dot{m}_{c,\,\text{internal}}}{\dot{m}_s} u_{c'} + \frac{\dot{m}_f}{\dot{m}_s} u_{f'}}{1 + \frac{\dot{m}_{c,\,\text{internal}}}{\dot{m}_s} + \frac{\dot{m}_f}{\dot{m}_s}} \tag{48}$$

$$T_{t_r} = \frac{T_{t_{s'}} + \frac{\dot{m}_{c,\,\text{internal}}}{\dot{m}_s} T_{t_{c'}} + \frac{\dot{m}_f}{\dot{m}_s} T_{t_{f'}}}{1 + \frac{\dot{m}_{c,\,\text{internal}}}{\dot{m}_s} + \frac{\dot{m}_f}{\dot{m}_s}}, \tag{49}$$

and the total pressure is given by Equation (46).

2.6.7 Rotor Calculations

On page 287 of reference 1, we see that the degree of reaction R can be found from

$$R = 1 - \frac{(1-\tau_s)\left(1 + \frac{\gamma_t - 1}{2} M_r^2\right)}{2(\gamma_t - 1) M_T^2}. \tag{50}$$

In this analysis, we assume that $M_T = 0.5$. Knowing R, we can determine the ratio of stagnation temperatures relative to the rotor blades to that at the inlet to the rotor in stationary coordinates. This is

$$\alpha_T = \frac{T_{t_b}}{T_{t_b}} = 1 + \frac{(4R-3)\left(\frac{\gamma_t - 1}{2} M_T^2\right)}{1 + \frac{\gamma_t - 1}{2} M_r^2}. \tag{51}$$

The cooling mass flow calculation for the rotor is similar to that of the stator, except that the total temperature must be taken relative to the blades.

$$\left(\frac{\dot{m}_c}{\dot{m}}\right)_r = St r \frac{c_{p_t}}{c_{p_c}} \left(\frac{\alpha_t T_{t_r} - T_w}{T_w - T_c}\right). \tag{52}$$

One form of energy transfer that exists in the rotor that doesn't exist in the stator is the work done by the fluid on the blades. Again, we have to differentiate between the cases where coolant is added and where it is not added to the main flow. In the former situation, a balance of energy flow at the rotor results in

$$\left(\frac{\dot{W}}{\dot{m}}\right)_i = c_{p_t}\left[T_{t_r} - \left(1 + \left(\frac{\dot{m}_c}{\dot{m}}\right)_r\right)\tau_s T_{t_s} + \left(\frac{c_{p_c}}{c_{p_t}}\right)\left(\frac{\dot{m}_c}{\dot{m}}\right)_r T_c\right]. \tag{53}$$

In the case of no mass addition, the result is

$$\left(\frac{\dot{W}}{\dot{m}}\right)_i = c_{p_t}[T_{t_r} - \tau_s T_{t_s} - Str(\alpha_T T_{t_r} - T_w)]. \tag{54}$$

In Section 2.6.10, Cooling Flow Fraction and Work Per Unit Total Core Mass Flow, after we have all the contributions from each individual rotor stage, these contributions will be combined to arrive at the total work output of the rotor. This is the quantity that will be used in the turbine-compressor-fan work balance in order to determine whether the initial estimate of τ_t was correct.

2.6.8 Energy and Mass Changes From Rotor to Stator

As we did in Section 2.6.5, Energy and Mass Changes From Stator to Rotor, we apply conservation of mass and energy to keep track of the mass flow rate and the total temperature in the turbine. Again, if the cooling air is not injected into the turbine flow, the mass flow will be constant, and $\dot{m}_{s,i+1}/\dot{m}_{s,i} = 1$. With cooling air injection, this ratio is

$$\frac{\dot{m}_{s,i+1}}{\dot{m}_{s,i}} = 1 + \left(\frac{\dot{m}_c}{\dot{m}}\right)_{s,i} + \left(\frac{\dot{m}_c}{\dot{m}}\right)_{r,i}\left(\frac{\dot{m}_r}{\dot{m}_s}\right)_i. \tag{55}$$

Since by assumption, the stage temperature ratio is τ_s, we have the total temperature at the next stage as

$$T_{t_{s,i+1}} = \tau_s T_{t_{s,i}}. \tag{56}$$

2.6.9 Rotor Total Pressure Calculations

The derivation of the expressions for the total pressure drop across the rotor are similar to those for the stator. As with the stator, changes in total pressure are divided into two groups: changes caused by cooling, frictional losses, and work, and changes due to mixing of the cooling flow with the main flow. Referring to the simplified schematic in Figure 9, "Simplified Schematic of Flow Mixing in Stator (or Rotor) Cascade," we assume that all the changes in the first group occur before the trailing edge of the blades (between r and r', c and c', and f and f'), and the mixing total pressure changes occur after the trailing edge (between points r', c', and f' and point s).

There are two primary differences in the total pressure expressions for the rotor and the stator. First, in addition to the entropy change due to cooling, and from the cooling flow addition itself, we include at the rotor cascade the entropy rise due to viscous effects across the entire stage. Second, as noted in Section 2.6.7, Rotor Calculations, the heat flux into the walls and thus into the cooling air is proportional to the blade relative total temperature of the flow.

In order to estimate the entropy change in the main flow due to viscous effects, we note that an uncooled turbine stage that does work at the same rate ($\dot{W}$) would have an incremental temperature ratio of

$$\frac{T_{t_w} + dT_{t_w}}{T_{t_w}} = 1 - \frac{d\dot{W}}{\dot{m} c_{p_t} T_{t_w}}. \tag{57}$$

We use the subscript w to emphasize that this is a hypothetical, uncooled case. At a polytropic efficiency of η_{poly}, the corresponding incremental total pressure ratio would be

$$\frac{P_{t_w} + dP_{t_w}}{P_{t_w}} = \left(1 - \frac{d\dot{W}}{\dot{m} c_{p_t} T_{t_w}}\right)^{\frac{\gamma}{(\gamma - 1)\eta_{\text{poly}}}}. \tag{58}$$

Taking the limit of small changes in total temperature, and using Equation (35), we find that the incremental change in entropy corresponding to dT_t for an uncooled turbine would be

$$\frac{ds_w}{c_{p_t}} = \left(1 - \frac{1}{\eta_{\text{poly}}}\right)\frac{dT_{t_w}}{T_{t_w}}. \tag{59}$$

Now we take a turbine that does no work, but which is cooled, and see that the incremental entropy change is given by

$$\frac{ds_c}{c_{p_t}} = \frac{dQ}{T_c} = \left(1 + \frac{\gamma - 1}{2}M^2\right)\frac{dT_{t_c}}{T_{t_c}}, \tag{60}$$

where the subscript c emphasizes that this turbine does no work, and where, as in Section 2.6.6, Stator Total Pressure Calculations, the Mach number is assumed to be a linear function of the total temperature. Finally, we think of each incremental total temperature change in our cooled, work producing turbine as being due in part to cooling and in part to producing work, or $dT_t = dT_{t_c} + dT_{t_w}$. That is,

$$\begin{aligned} dT_{t_c} &= \frac{Q_r}{\dot{m}_r c_{p_t} (T_{t_r} - T_{t_{r'}})} dT_t \\ dT_{t_w} &= \frac{W}{\dot{m}_r c_{p_t} (T_{t_r} - T_{t_{r'}})} dT_t \end{aligned}, \tag{61}$$

when both Q_r and W are taken positive as energy is removed from the flow.

If we further assume that an incremental entropy change in the cooled, work producing turbine is the sum of the entropy changes due separately to cooling and to viscous effects, as given in Equation (60) and Equation (59),we can find the incremental total pressure change using Equation (35). This expression can then be integrated from T_{t_r} to $T_{t_{r'}}$, yielding the total pressure ratio

$$\frac{P_{t_{r'}}}{P_{t_r}} = \left(\frac{T_{t_{r'}}}{T_{t_r}}\right)^{\frac{\gamma}{\gamma-1}\left(1-\frac{1}{c_{p_t}\left(T_{t_r}-T_{t_{r'}}\right)}\left(\frac{1-\eta_{\text{poly}}}{\eta_{\text{poly}}}\frac{\dot{W}}{\dot{m}_r}+\left(1-\frac{\gamma-1}{\gamma}a\right)\frac{Q}{\dot{m}_r}\right)\right)} \cdot \exp\left(\frac{\frac{Q}{\dot{m}_r}}{c_{p_t}(T_{t_r}-T_{t_{r'}})}\frac{c}{2}\left(\left(T_{t_{r'}}+\frac{b}{c}\right)^2-\left(T_{t_r}+\frac{b}{c}\right)^2\right)\right), \tag{62}$$

where a, b, and c are as given in Equation (43), with the subscript s for stator replaced by an r for rotor.

As was done for the stator in Section 2.6.6, Stator Total Pressure Calculations, the total pressure after the main and cooling streams mix is found by conserving both energy and momentum.

2.6.10 Cooling Flow Fraction and Work Per Unit Total Core Mass Flow

Since we have ratios of cooling air flow rates to local turbine flow rates, and ratios of flow rates across all the cascades, we can find ε, the overall cooling flow fraction required,

$$\varepsilon = \sum_{j=1}^{N}\left[\left(\frac{\dot{m}_c}{\dot{m}}\right)_{s,j}\prod_{i=1}^{N-1}\left(\frac{\dot{m}_r}{\dot{m}_s}\right)_i\left(\frac{\dot{m}_{s+1}}{\dot{m}_r}\right)_i+\left(\frac{\dot{m}_c}{\dot{m}}\right)_{r,j}\left(\frac{\dot{m}_r}{\dot{m}_s}\right)_1\prod_{i=2}^{N}\left(\frac{\dot{m}_s}{\dot{m}_{r-1}}\right)_i\left(\frac{\dot{m}_r}{\dot{m}_s}\right)_i\right]. \tag{63}$$

In Section 2.6.7, Rotor Calculations, we calculated the work done by each individual turbine stage as a function of the core mass flow at that stage. In order to balance the turbine work output against the compressor and fan work input, we accumulate all the individual stage contributions to the turbine work into a single value, representing the total turbine work per unit total core mass flow:

$$\frac{\dot{W}}{\dot{m}} = \sum_{i=1}^{N}\left(\frac{\dot{W}}{\dot{m}}\right)_i\left(\frac{\dot{m}_i}{\dot{m}}\right) = \sum_{i=1}^{N}\left(\frac{\dot{W}}{\dot{m}}\right)_i\left(\sum_{j=1}^{i-1}\frac{\dot{m}_{c,j}}{\dot{m}_4}+1\right)(1+f-\varepsilon). \tag{64}$$

2.7 Turbine-Compressor-Fan Work Balance

In its most basic form, the work balance states that the energy that the flow transmits to the main shaft(s) in the turbine is equal to the shaft work done on the flow in the compressor plus that in the fan, or $\dot{W}_t = \dot{W}_c + \dot{W}_f$. In this analysis, we can model the compressor work as proportional to the mass flow rate and the total temperature drop across the compressor. Since cooling air is taken out of the compressor flow at a point where the total temperature is equal to $\tau_c{}'T_{t_2}$, the compressor work becomes

$$\dot{W}_c = \dot{m} c_{p_c} T_{t_2} (\tau'_c - 1) + \dot{m} (1 - \varepsilon) c_{p_c} T_{t_2} (\tau_c - \tau'_c) \ . \tag{65}$$

The work done by the fan is similarly found.

The work done by the flow in the turbine, however, cannot be found from an expression such as Equation (65). This is because in the turbine, in addition to the temperature drop as a result of the work exerted by the fluid on the blades, the temperature changes because of the effects of blade cooling. In addition, the mass flow rate can change at each cascade due to the addition of cooling air to the flow. The amount of cooling air needed is in turn dependent on the temperature and the flow rate. A more detailed examination of the turbine is required in order to find the work done (see Section 2.6, Turbine Characterization). For now, the work balance can be written as

$$\dot{W}_t = \dot{m} c_{p_c} T_{t_2} [(\tau'_c - 1) + (1 - \varepsilon) (\tau_c - \tau'_c) + \alpha (\tau_f - 1)] \ . \tag{66}$$

3.0 Regenerated Ground Based Engine

The thermal efficiency of a gas generator can be improved by the addition of a regenerator, which transfers heat from the flow exiting the turbine to the flow exiting the compressor (see Figure 10, "Ground Based Engine With Regeneration"). The penalties involved, however, include a greatly increased engine mass due in part to the size of the heat exchanger, which makes such a powerplant unsuitable for use on an airplane. In addition, the regenerated gas turbine is most efficient at low pressure ratios. This means that the work extracted per kilogram of working fluid is also low, so that such an engine must be larger than a non-regenerated engine of equivalent power. For ground-based gas turbines, though, weight is not a prime consideration, and the high efficiencies possible with regeneration make this type of engine attractive. A regenerated gas turbine can approach the thermal efficiency of a Carnot cycle operating between the minimum and maximum temperatures of the engine.

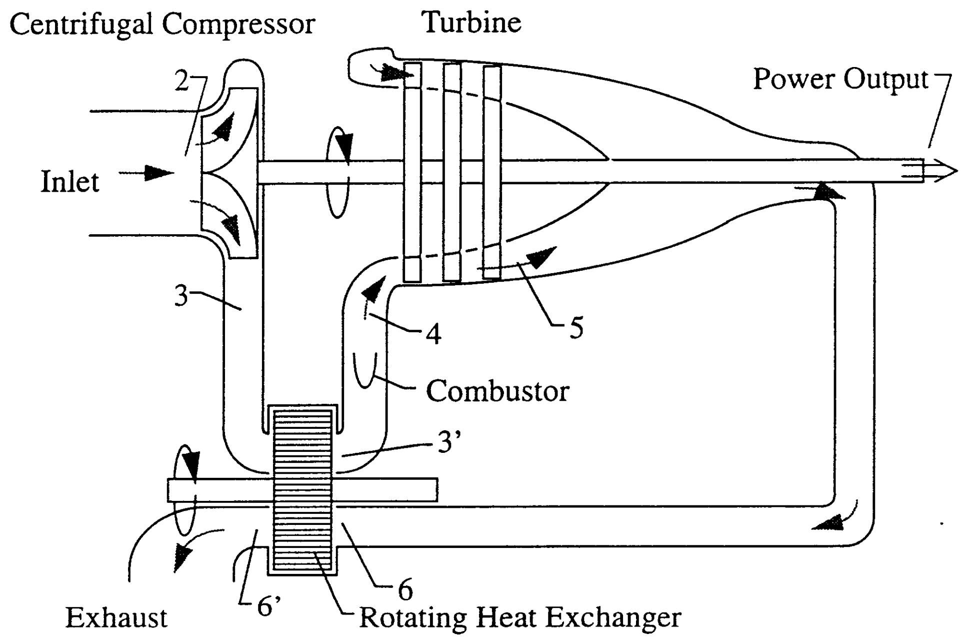

FIGURE 10. Ground Based Engine With Regeneration

Two equivalent measures of the overall efficiency of the device are the specific fuel consumption s and the thermal efficiency η_{thermal}. The specific fuel consumption is defined as the fuel mass flow rate per unit power output:

$$s = \frac{\dot{m}f}{\dot{W}_{\text{net}}}, \tag{67}$$

where $\dot{W}_{\text{net}}$ is the net power extracted from the engine. The thermal efficiency is defined as the power output per unit fuel energy input, or

$$\eta_{\text{thermal}} = \frac{\dot{W}_{\text{net}}}{hf\dot{m}} = \frac{1}{hs} . \qquad (68)$$

For a ground-based regenerated gas turbine, the useful work output is the turbine work in excess of what is required to drive the compressor,

$$\dot{W}_{\text{net}} = \dot{W}_t - \dot{W}_c . \qquad (69)$$

If we call P the nondimensional power output of the engine, then

$$P = \frac{\dot{W}_{\text{net}}}{\dot{m}c_{p_c}T_0} = \frac{\dot{W}_t}{\dot{m}c_{p_c}T_0} - (\tau_c - 1) , \qquad (70)$$

where the fact that the system is stationary ($M_0 = 0$) has been used. To evaluate P, we first assume a value for the compressor pressure ratio, which gives us the total temperature ratio from

$$\tau_c = \pi_c^{\frac{\gamma_c - 1}{\gamma_c \eta_{\text{poly}}}} . \qquad (71)$$

We then use the procedure outlined in Section 2.6 on page 13 to find the work output of the turbine per unit mass flow.

In order to close this system of equations, we need to set the condition that the gas is exhausted with total pressure equal to atmospheric pressure, so that all of the energy possible is extracted by the turbine and the regenerator. This can be done by tracing the pressure through the engine and assuming that $p'_{t_6} = p_0$, or

$$\pi_c \pi_b \pi_t \pi_r = 1 , \qquad (72)$$

where $\pi_r = (p'_{t_3}/p_{t_3})(p'_{t_6}/p_{t_6})$ is the pressure drop through both ducts of the heat exchanger. This pressure drop can be evaluated using Equation 3.6 from Kerrebrock,

$$\pi_r = 1 - \alpha_r M_r^2 \left(\frac{\varepsilon_r}{1 - \varepsilon_r} \right) , \qquad (73)$$

where α_r is a constant of order unity, M_r is the Mach number of the flow in the heat exchanger, and ε_r is the regenerator effectiveness, which is defined as

$$\varepsilon_r = \frac{T'_{t_3} - T_{t_3}}{T_{t_6} - T_{t_3}} . \qquad (74)$$

It remains now to find the fuel-air ratio f. This can be found by balancing the energy across the combustor, noting that the temperature at the inlet to the combustor has been raised in the regenerator from T_{t_3} to T'_{t_3}. The result is

$$f = \frac{c_{p_t} T_0 \theta_t - c_{p_c} \left(\frac{T'_{t_3}}{T_0} \right) T_0}{\eta_b h - c_{p_t} T_0 \theta_t} . \tag{75}$$

We can use the fact that the device is stationary to find T'_{t_3}/T_0 in the above expression, yielding

$$\frac{T'_{t_3}}{T_0} = \tau_c + \varepsilon_r \left(\theta_t \tau_t - \tau_c \right) . \tag{76}$$

4.0 Results of Turbofan Analysis and Conclusions

The model described in Section 2.0, Turbofan Aircraft Engine, was applied to a family of engines operating at sea level static, or takeoff, conditions, where the highest demands are made on the cooling system and the turbine materials. At each operating condition, results were found for engines with no cooling, with cooling flow injected into the turbine, and with cooling flow injected into the fan nozzle.

In one set of calculations, the turbine inlet temperature was varied while the turbine wall temperature was held constant, and in the other set of calculations, the turbine inlet temperature was held at the stoichiometric temperature for the fuel and the wall temperature was decreased. In both series, the engines are assumed to be such that the exhaust velocities of the core and fan flows are equalized, in order to maximize propulsive efficiency[1]. In addition, as the turbine inlet temperature is increased, and more energy is available in the flow at the turbine inlet, the bypass ratio is increased so that more energy is removed from the turbine flow and the core exhaust velocity is held constant. In one set of calculations, this jet velocity was chosen to be that of an engine with a turbine inlet temperature of 1650K, a turbine wall temperature of 1250K, and a bypass ratio of 7, a typical configuration for current high bypass engines. Another set of simulations modeled a low bypass engine and a third set was based on a high bypass engine operating at cruise conditions. Finally, in all calculations the cooling flow was bled off of the compressor at the point which gives a total pressure of the cooling flow as it is being injected, either into the fan duct or into the turbine, equal to a constant multiple of the static pressure of the main flow at that point. This constant factor is typically taken as 1.11.

4.1 Effects of Increasing Turbine Inlet Temperature for a High Bypass Engine

The first set of results were obtained to determine the effect of elevating the turbine inlet temperature to that of stoichiometric burning of the fuel. Specifically, the wall temperature in the turbine was kept constant at 1250K, approximately the maximum temperature allowable with current material technology, and the amount of cooling air $(\dot{m}\varepsilon)$ was allowed to increase as the turbine inlet temperature increased from 1250K to 2500K. The intention was to determine whether the cost of such large scale cooling required at stoichiometric turbine inlet temperature was enough to cancel the efficiency and power benefits of increasing turbine inlet temperature. These plots include a line representing engines cooled using film cooling. This plot is not intended to accurately represent the current state of engine technology, but to enable a comparison of the internally cooled and the film cooled engines under the same (simplified) conditions.

In this model, it is assumed that if the exhaust in either jet does not reach sonic conditions, then it is ideally expanded to atmospheric pressure. When this is the case, from Equation (4) and Equation (5) we see that the thrust is dependent primarily upon the

1. Even though the engine isn't moving in the SLS case!

exhaust velocities. If we hold these velocities constant and subsonic as explained above, then the sum of the fan and core thrusts given by Equation (4) and Equation (5) is

$$\frac{F_8}{\dot{m}a_0} + \frac{F_6}{\dot{m}a_0} \approx \frac{u_{6,\,8}}{a_0}(1+\alpha) \ , \tag{77}$$

so that the total thrust nondimensionalized by the total mass flow, given by Equation (6), is approximately constant. This can be seen in Figure 11, "Thrust vs. Turbine Inlet Temperature," on page 35.

As Figure 16, "Bypass Ratio vs. Turbine Inlet Temperature," shows, the bypass ratio of the cooled engines increases from about 3 to approximately 13 as the turbine inlet temperature increases, in order to accommodate the extra energy in the turbine flow. In the uncooled engine, more energy is available to do work on the blades, so a larger bypass ratio (of about 16) is required. If the bypass ratio is seen as a measure of the core flow energy available to do useful work, then we can see that both of the internal cooling schemes reduce this available energy, and that this loss is greater for the scheme in which the cooling flow is injected into the turbine. The film cooled engines, because of the very large fraction of cooling air required, achieve a maximum bypass ratio of less than 5 at a turbine inlet temperature of approximately 1600 K.

Since the nondimensionalized thrust is constant, the actual thrust increases with increasing bypass ratio. The specific impulse, which is thrust per unit fuel weight flow rate, therefore increases with turbine inlet temperature, as shown in Figure 12, "Specific Impulse vs. Turbine Inlet Temperature." This figure also shows that there is a penalty involved in cooling the turbine. However, for the engines with internally cooled turbines under these conditions, this penalty is small and the two different ducting schemes result in almost identical specific impulses. From a comparison of the bypass ratios, we could infer that the scheme in which cooling air was injected into the bypass duct produces more thrust per unit core airflow than does the turbine injection scheme. Balancing this effect in the calculation of the specific impulse, however, is the fact that the bypass injection scheme uses more fuel per unit core airflow, as will be seen below. In the plot of the specific impulse, we see that these two effects nearly cancel each other out. The specific impulse of the film cooled engines is lower than that of any of the other engines, reaching a peak of 7200 sec at 1600K.

The fraction of the core flow which must be bled off to cool the turbine is significant at high turbine inlet temperatures, as is shown in Figure 13, "Cooling Flow Fraction vs. Turbine Inlet Temperature." With both ducting schemes, the cooling flow fraction rises rapidly with turbine inlet temperature until at stoichiometric temperatures approximately 17% of the core flow must be used for cooling when it is injected into the turbine, and approximately 9% must be used for cooling when it is injected into the bypass duct. The difference in required cooling mass flow between the two ducting schemes is due to the fact that different temperature cooling air is required in each case. For the engines with the cooling air diverted into the bypass duct, the low pressure at the point of injection allows us to take the cooling air from further upstream in the compressor, where the air is at a lower temperature. For the engines with turbine coolant injection, the high pressures in the turbine require higher pressure, and therefore higher temperature, cooling air. Here is where we see the most dramatic effect of using film cooling in the turbine in place of internal cooling. The fraction of compressor air needed in order to cool the turbine rises rapidly with increasing turbine inlet temperature, reaching approximately 27% at a turbine inlet temperature of 1650 K.

Since the fraction of core flow removed for cooling is different for the two cooling schemes, the amount of fuel needed to raise the temperature of the remaining air to the prescribed turbine inlet temperature is also different. This is because for a given combustor exit temperature, $\dot{m}_f/(\dot{m} - \dot{m}\varepsilon)$ (fuel mass flow per unit combustor airflow) is fixed. This effect can be seen in Figure 14, "Fuel-Air Ratio vs. Turbine Inlet Temperature," on page 36, which plots fuel mass flow per unit core mass flow $(\dot{m}_f/\dot{m})$ at the inlet of the compressor for various turbine inlet temperatures. In the uncooled engine, all of this core mass flow $\dot{m}$ passes through the combustor (ε is zero), so more fuel is needed to achieve a given temperature than in the cooled cases. Likewise, a greater fraction of $\dot{m}$ passes through the combustor in the cooled engines with bypass injection than in the cases with turbine injection, so the fuel-air ratio is greater in these engines. The difference in the fuel requirements increases as the turbine inlet temperature, and therefore the cooling flow fraction, increases.

If there is no flow added to the bypass duct, then holding the fan temperature ratio constant as the turbine inlet temperature increases results in constant exhaust velocity as can be seen from Equation (9), Equation (11), and Equation (13). This is reflected in the results shown in Figure 15, "Fan Temperature Ratio vs. Turbine Inlet Temperature." When the cooling air is added to the bypass duct, however, increases in the bypass flow temperature due to the cooling air must be balanced by a drop in the fan temperature ratio, in order to keep the exhaust velocity constant.

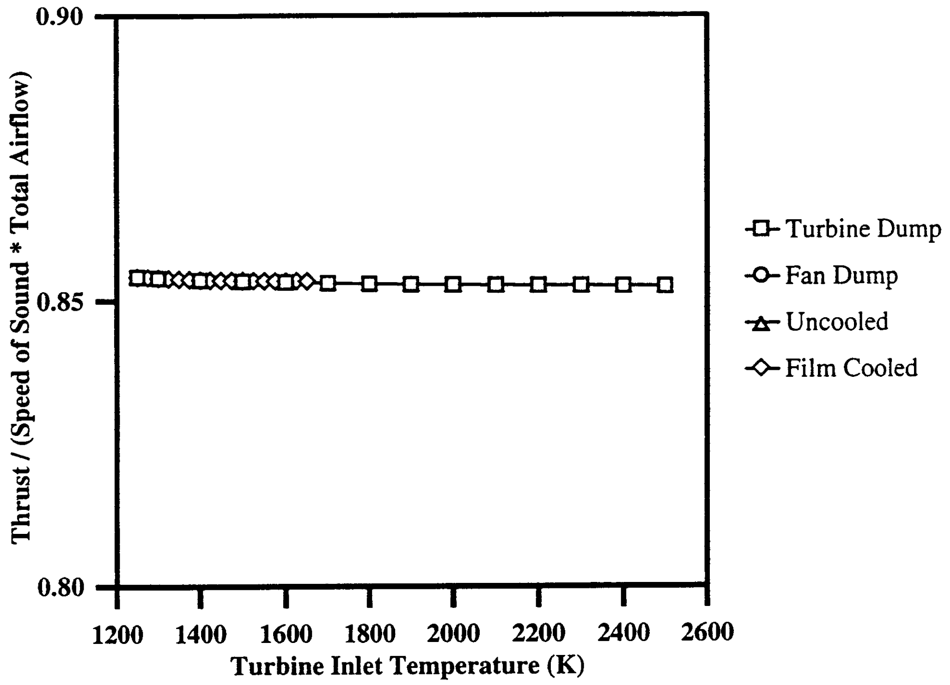

FIGURE 11. Thrust vs. Turbine Inlet Temperature

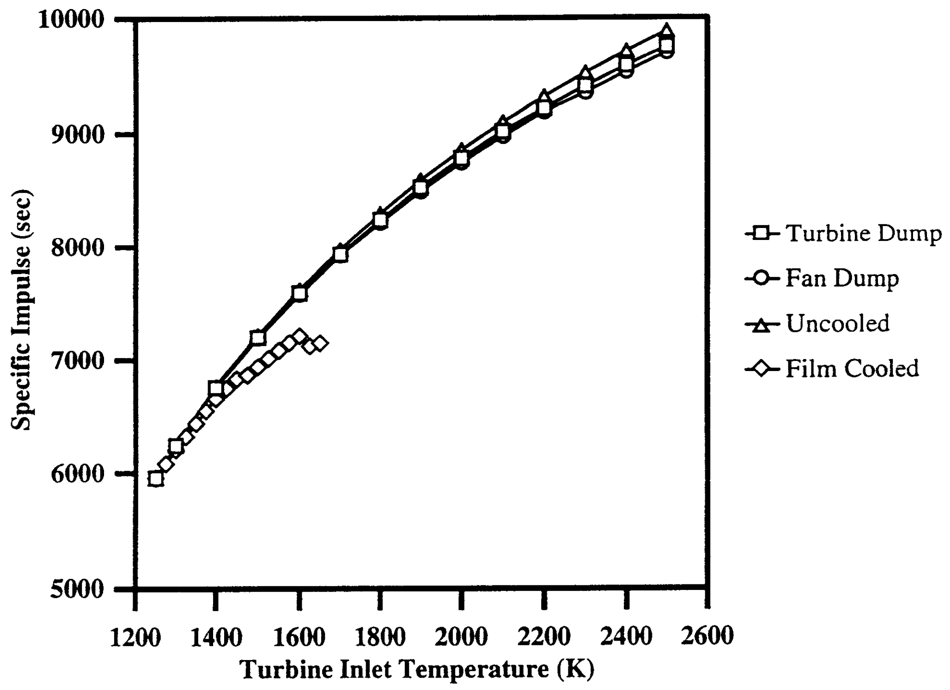

FIGURE 12. Specific Impulse vs. Turbine Inlet Temperature

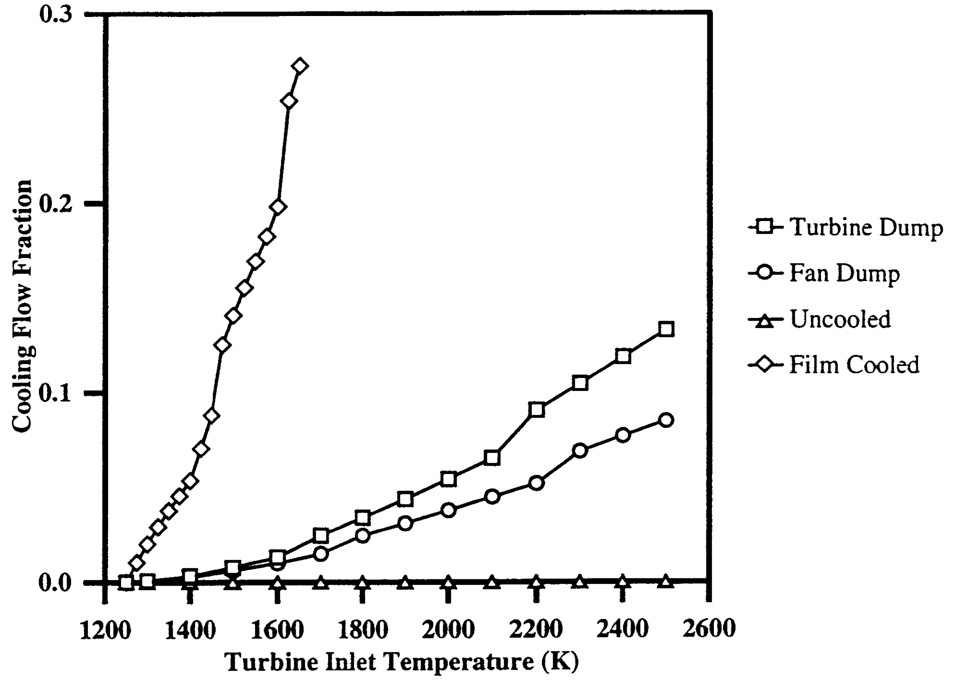

FIGURE 13. Cooling Flow Fraction vs. Turbine Inlet Temperature

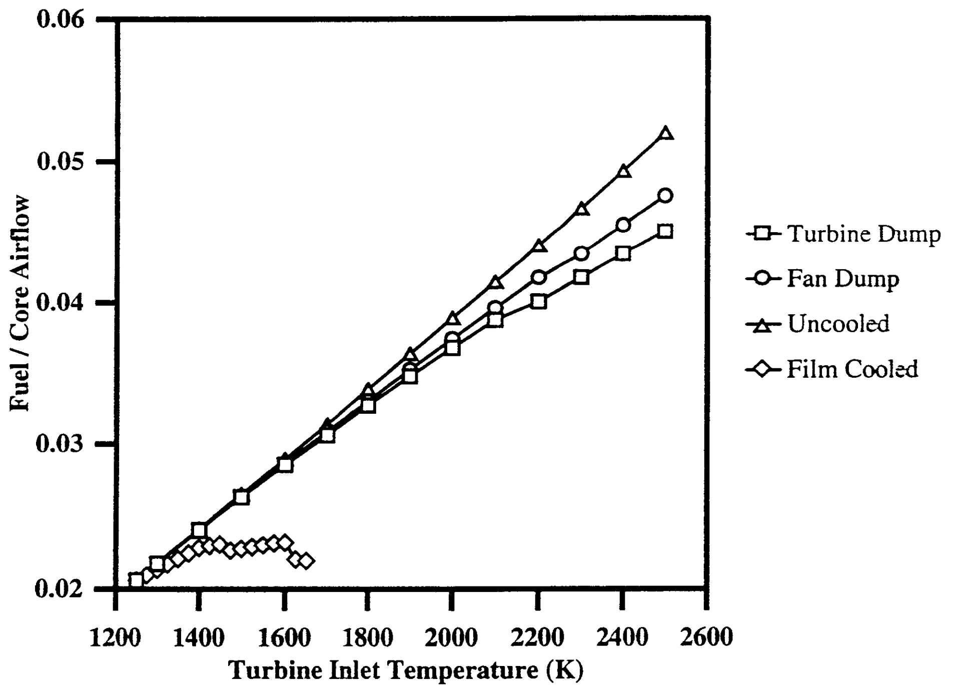

FIGURE 14. Fuel-Air Ratio vs. Turbine Inlet Temperature

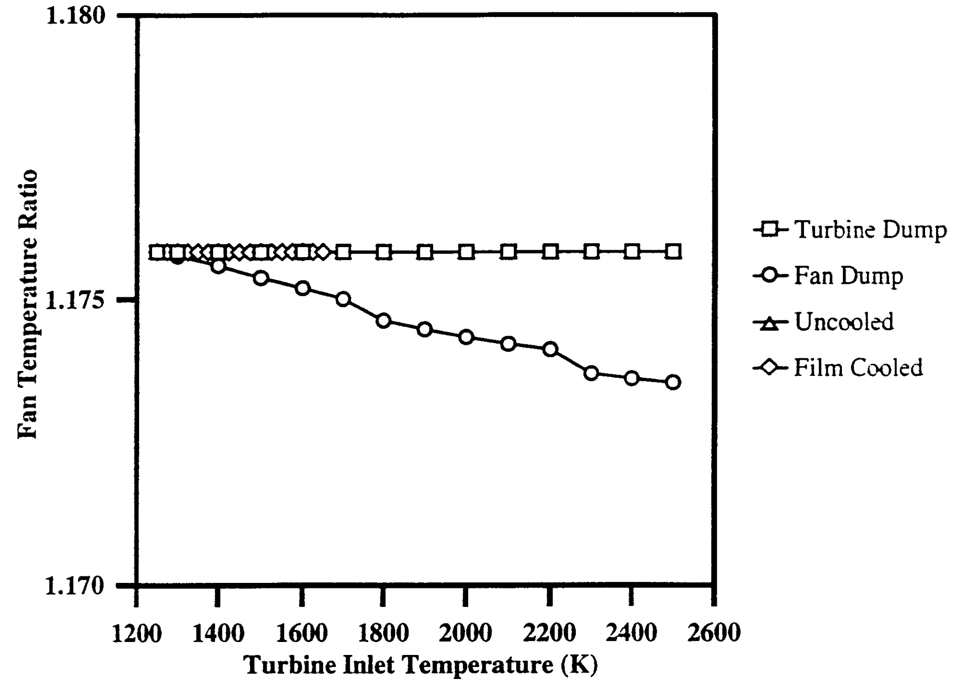

FIGURE 15. Fan Temperature Ratio vs. Turbine Inlet Temperature

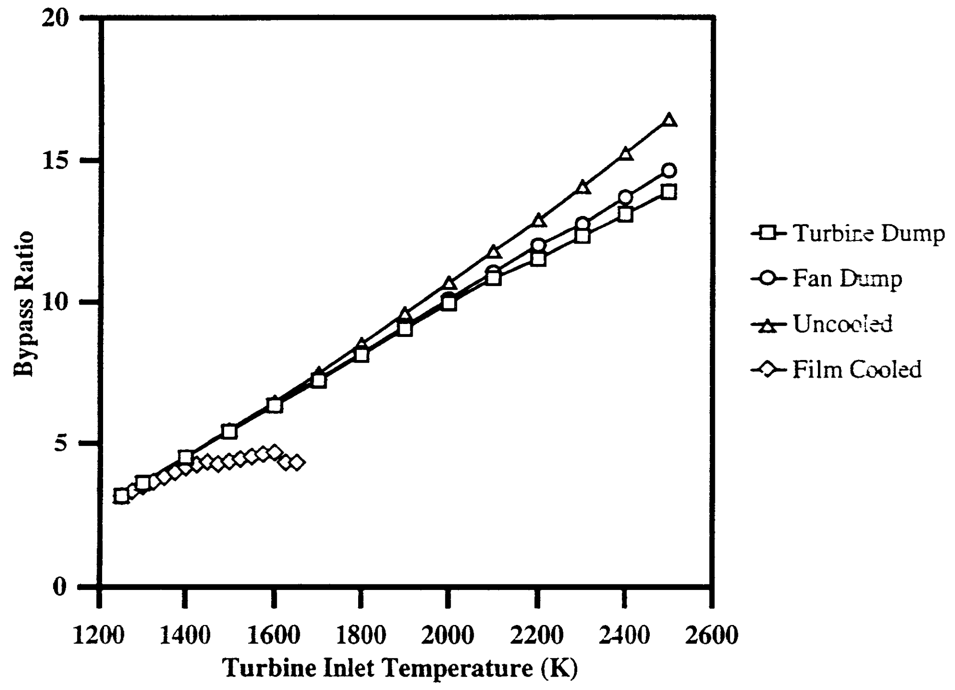

FIGURE 16. Bypass Ratio vs. Turbine Inlet Temperature

4.2 Effects of Decreasing Wall Temperature For a High Bypass Engine

In Section 4.1, Effects of Increasing Turbine Inlet Temperature for a High Bypass Engine, it was established that considerable improvement in engine performance can be achieved by raising the turbine inlet temperature to that of stoichiometric burning of the fuel. In the following series of graphs, results are given of a simulation in which the turbine inlet temperature was kept constant at 2500K while the wall temperature was lowered from 1250K to 900K. This was done to explore another potential of evaporative turbine blade cooling. This is the promise of cheaper and lighter blade materials made possible by lower material temperatures.

As in the case where the turbine inlet temperature was varied, detailed in Section 4.1, the condition of fixed exit velocities results in constant thrust. The specific impulse drops for both ducting schemes with falling turbine wall temperature, as shown in Figure 17, "Specific Impulse vs. Turbine Wall Temperature." This effect is much smaller for the case of bypass cooling air injection than for that of turbine injection.

Figure 18, "Cooling Flow Fraction vs. Turbine Wall Temperature," on page 39, shows the cooling flow fraction required as the wall temperature is lowered. We can see here the same trends as in Figure 13, "Cooling Flow Fraction vs. Turbine Inlet Temperature," except that the difference between the two ducting schemes is still more pronounced. At a wall temperature of 1000 K, almost 45% of the core air must be used for cooling if it is to be injected into the turbine compared to about 15% if it is to be diverted to the bypass duct.

As mentioned in Section 4.1, Effects of Increasing Turbine Inlet Temperature for a High Bypass Engine, under the given assumptions the bypass ratio can be seen as a measure of the energy in the core flow that can be used to do useful work. With this in mind we see from Figure 19, "Bypass Ratio vs. Turbine Wall Temperature," that the penalties involved in lowering the turbine wall temperature are much greater if the cooling air is added to the turbine airflow than if it is added to the bypass airflow. At a turbine inlet temperature of 2500 K, the bypass ratio drops from approximately 13 to less than 8 for a 250 K drop in wall temperature in the case of turbine injection. For the same temperature drop, engines with bypass cooling air injection go from a bypass ratio of about 14.5 to one of about 13.5.

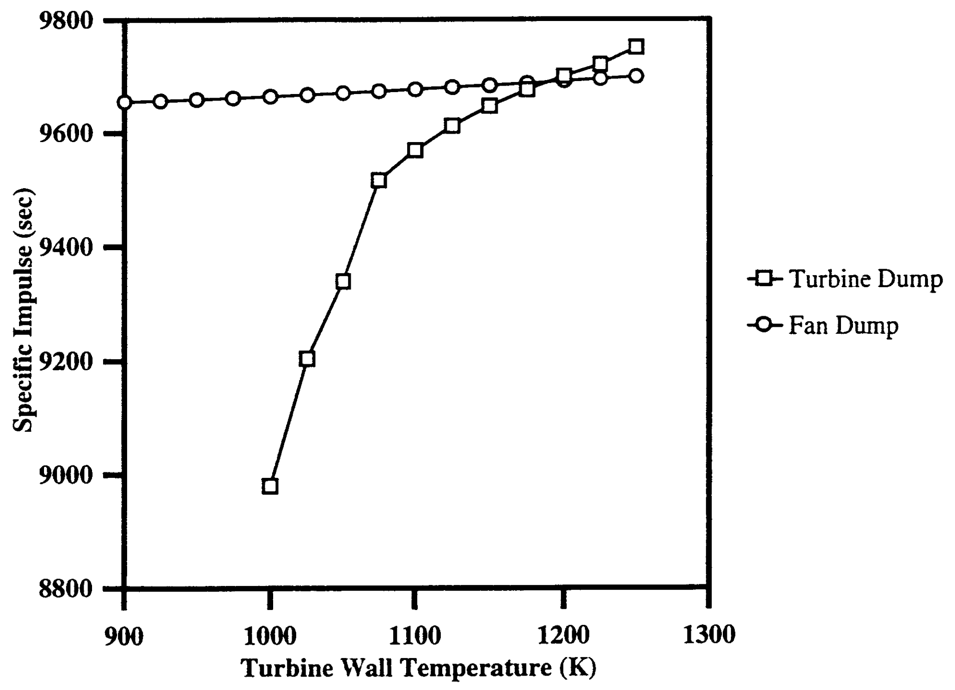

FIGURE 17. Specific Impulse vs. Turbine Wall Temperature

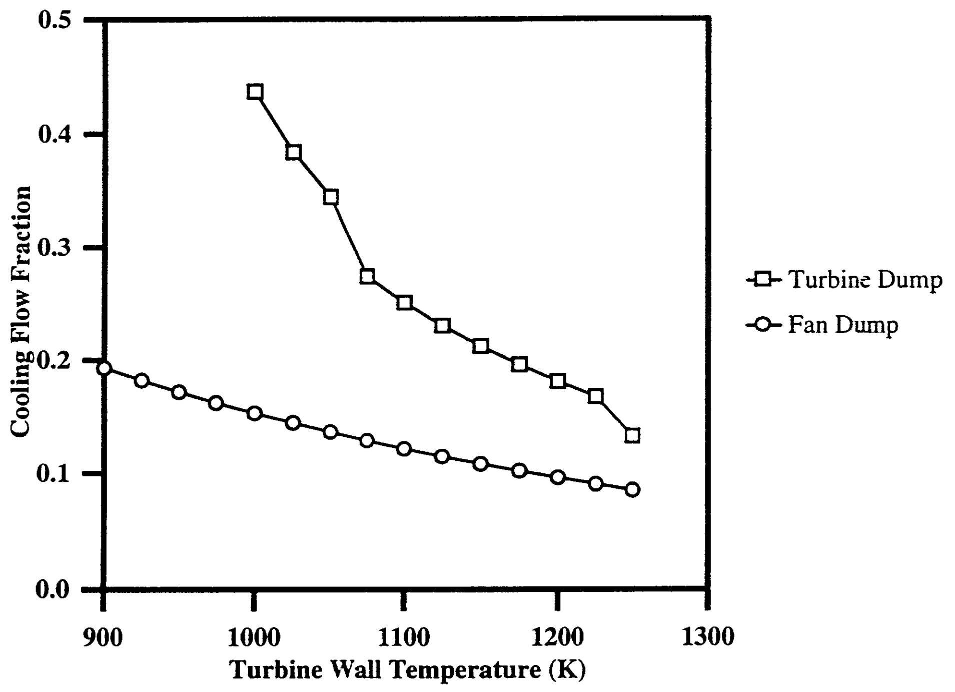

FIGURE 18. Cooling Flow Fraction vs. Turbine Wall Temperature

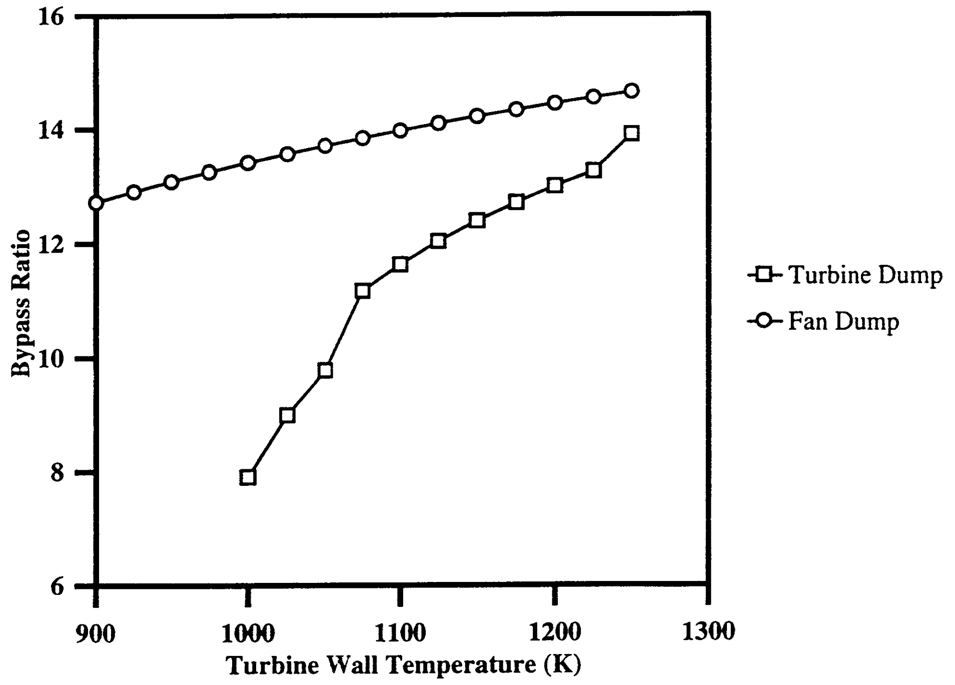

FIGURE 19. Bypass Ratio vs. Turbine Wall Temperature

4.3 Effects of Increasing Turbine Inlet Temperature for a Low Bypass Engine

In addition to finding the effects of cooling on a high bypass turbofan engine, a series of cases were run to find the effects of cooling on low bypass engines. In this section, results will be shown for a family of engines in which the turbine wall temperature was held constant while the turbine inlet temperature was increased. To simulate a low bypass engine, the exhaust velocities for the core and bypass jets were chosen to be such that an engine with a turbine inlet temperature of 1600 K has a bypass ratio of approximately 0.5, and then held constant while T_{t_4} was increased.

Figure 20, "Thrust vs. Turbine Inlet Temperature for a Low Bypass Engine," plots the nondimensionalized thrust of Equation (6). From Equation (4) and Equation (5), we see that if the pressures at the exit planes of the two jets are different from atmospheric pressure, the total nondimensionalized thrust will no longer be constant as it was for the high bypass engines. As mentioned previously, this model assumes that once the exhaust pressures are sufficient to reach sonic conditions the exhaust nozzles are choked. Because the lower bypass engines have higher exhaust velocities, the exhaust nozzles are indeed choked, and the effect of the added pressure terms in Equation (4) and Equation (5) can be seen in Figure 20. At low turbine inlet temperatures, both the core and the bypass nozzles are choked. As the temperature in the core jet increases, however, the speed of sound in the core exhaust increases and the core exhaust Mach number drops below one while the bypass duct remains choked. The temperature at which the core exhaust becomes unchoked is indicated in the plot by the minima in the thrust curves.

The specific impulse is shown in Figure 21, "Specific Impulse vs. Turbine Inlet Temperature for a Low Bypass Engine." The specific impulse for the engines in which the cooling air is injected into the turbine is lower at the same turbine inlet temperature than that of the uncooled engines, as it was for the high bypass engines. For those with the cooling air injected into the bypass duct, however, the specific impulse is higher than an uncooled engine under the same operating conditions.

There are several key factors that explain this result. The first is that for the cooled engine with bypass coolant injection, heat is removed from the turbine without a loss of pressure. The second key factor is that the core exhaust nozzle is unchoked at the higher turbine inlet temperatures (precisely where the specific impulse of the cooled engine is higher). The fact that the exhaust velocity and the turbine inlet pressure are the same in both the cooled and uncooled engines, and that the core exhaust flow is expanded to the same atmospheric pressure, means that the turbine pressure ratio must also be approximately the same for both engines (the temperature of the exhaust has a smaller effect on the final velocity than does the pressure). Work per unit turbine flow is therefore only slightly affected, because the only pressure drop mechanism is work production (the slight effect is due to the different temperatures at which work is done).

The final key factor in explaining the increased specific impulse for the cooled engine with turbine coolant injection over that of the uncooled engines is that the bypass exhaust nozzle is choked in both cases. The condition of fixed exit velocities translates into fixing the speed of sound, since the nozzle is choked, which means fixing the temperature. The addition of cooling air to the much cooler air in the bypass duct accounts for some of this required temperature rise. The fan temperature ratio for the cooled engine can therefore be lower than that of the uncooled engine, and the work that is extracted from the turbine can go into increasing the bypass ratio instead of the turbine temperature ratio. It is true that the pressure at the bypass exhaust is lower in the cooled engine, but since the nozzle is underexpanded, the excess pressure in the uncooled engine is not used very efficiently. The lower bypass exhaust pressure in the cooled engine means that the nozzle is closer to an ideal nozzle than that of an uncooled engine.

In the core duct what is important is the pressure and not the temperature, and in the bypass duct what is important is the temperature and not the pressure. What this ducting scheme does is transfer heat without changing pressures - exactly what is called for under these conditions. Some of the loss of work from the fact that less air goes through the turbine (we saw that the work per unit turbine air was the same) is made up by adding heat to the bypass flow, and it is done without making up any of the saved fuel consumption. The result is an increase in specific impulse.

The amount of cooling air required for these engines is shown in Figure 22, "Cooling Flow Fraction vs. Turbine Inlet Temperature for a Low Bypass Engine." As in previous cases, the smaller amount of cooling air required to cool the engines with bypass coolant injection is due to the lower temperature of the cooling air used.

Finally, Figure 23, "Bypass Ratio vs. Turbine Inlet Temperature for a Low Bypass Engine," shows the effect of cooling on the bypass ratios. As mentioned in the discussion above on the specific impulse of this series of tests, the bypass ratio for the engines in which the cooling air is added to the bypass duct is significantly higher than that of the engines with turbine coolant injection, approaching that of the uncooled engines.

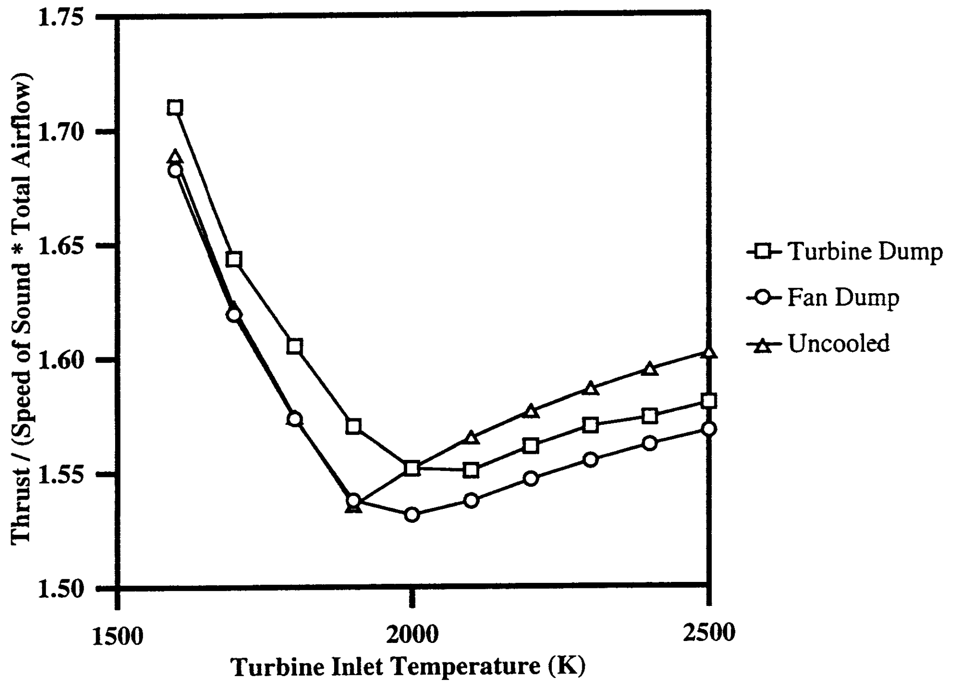

FIGURE 20. Thrust vs. Turbine Inlet Temperature for a Low Bypass Engine

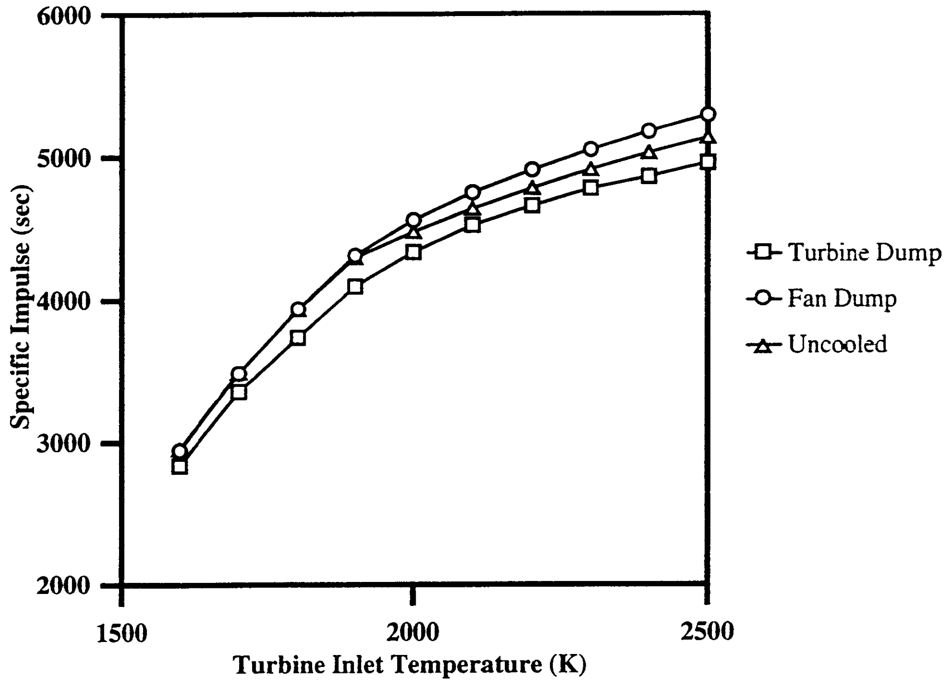

FIGURE 21. Specific Impulse vs. Turbine Inlet Temperature for a Low Bypass Engine

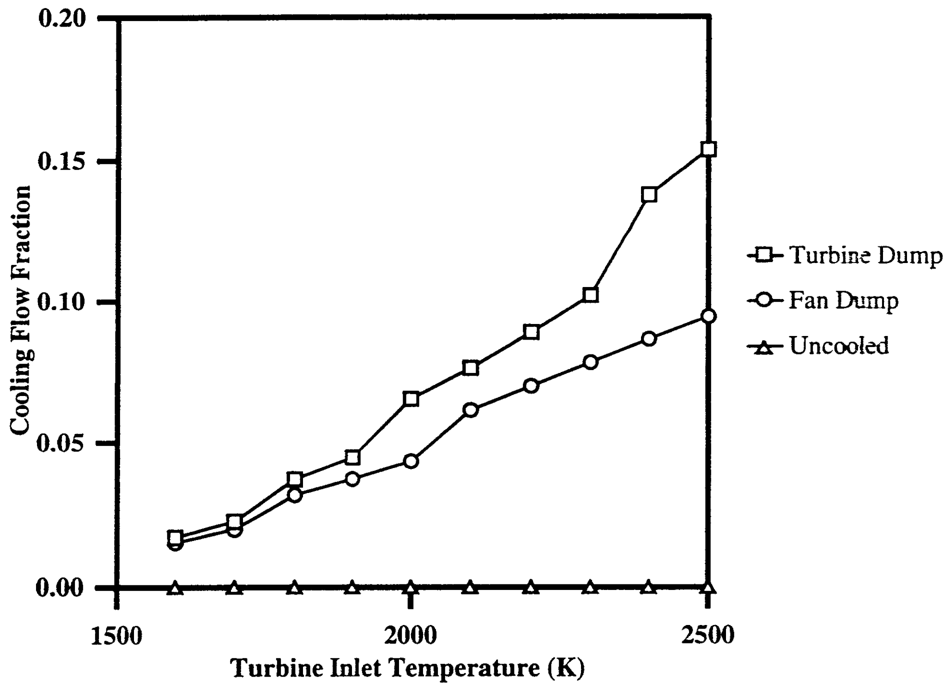

FIGURE 22. Cooling Flow Fraction vs. Turbine Inlet Temperature for a Low Bypass Engine

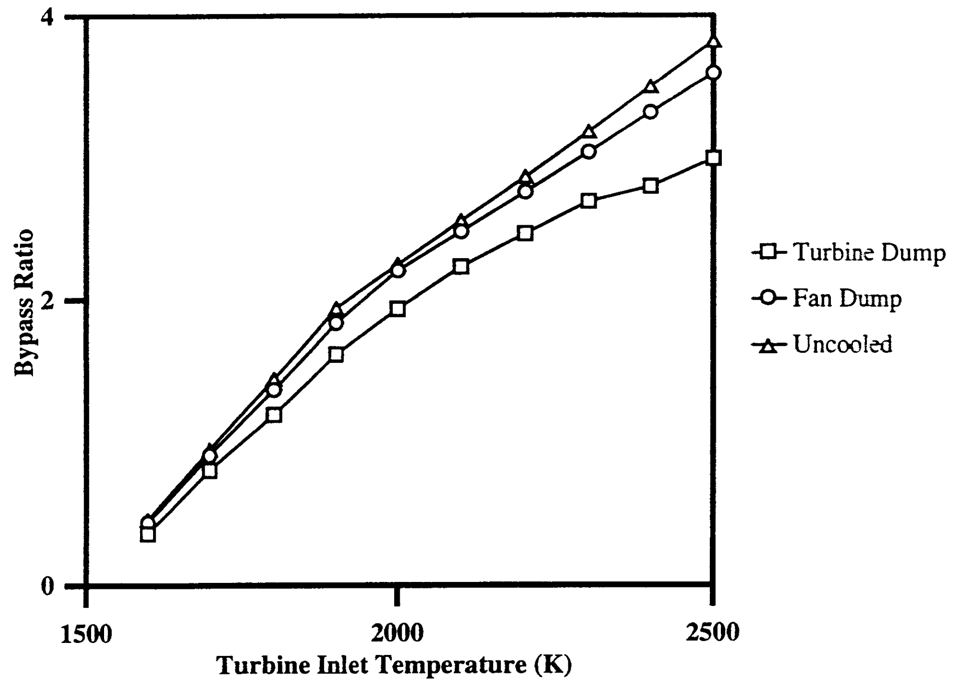

FIGURE 23. Bypass Ratio vs. Turbine Inlet Temperature for a Low Bypass Engine

4.4 Effects of Decreasing Wall Temperature for a Low Bypass Engine

In this series of cases, the turbine inlet temperature was held constant at 2500 K while the turbine wall temperature was decreased from 1250 K. The four parameters shown in the plots of this section are the nondimensionalized thrust (Figure 24), the specific impulse (Figure 25), the cooling flow fraction (Figure 26) and the bypass ratio (Figure 27).

The wall temperatures for the engines in which the cooling air is dumped into the turbine only go from 1250 K to 1050 K. Below this wall temperature, some of the turbine stages require negative work, in essence behaving as compressor stages, so these data points were omitted from the plots. One of the assumptions of the model is that the stage temperature ratio remains constant through the turbine. In the very aggressively cooled turbines, especially in the first stage, so much heat is taken from the flow that extra energy has to be added to the flow to achieve the stage temperature ratio required in order to satisfy the other conditions on the engine.

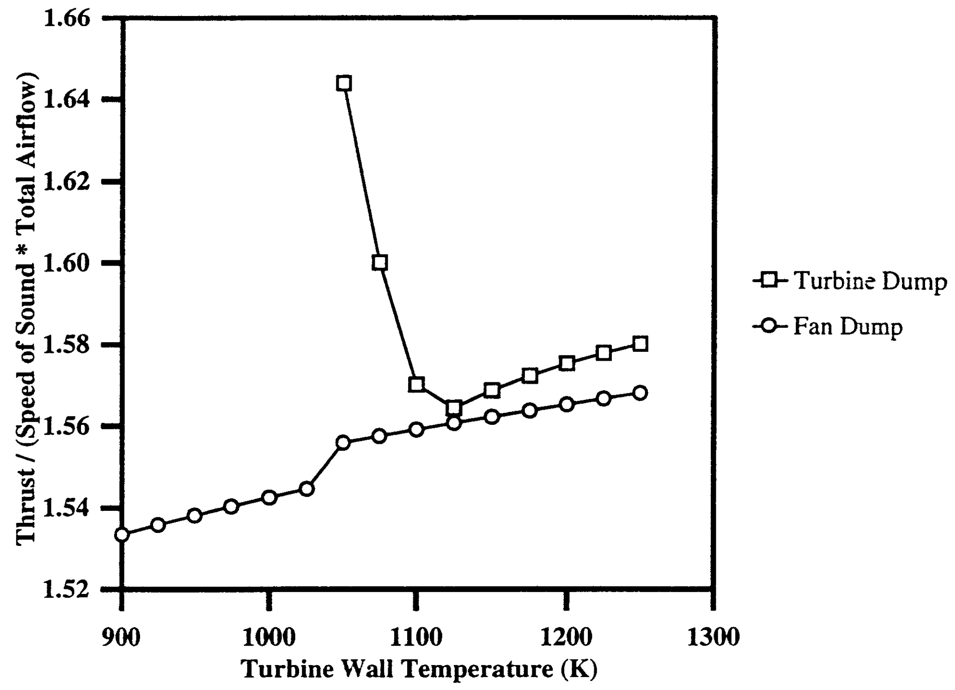

FIGURE 24. Thrust vs. Turbine Wall Temperature for a Low Bypass Engine

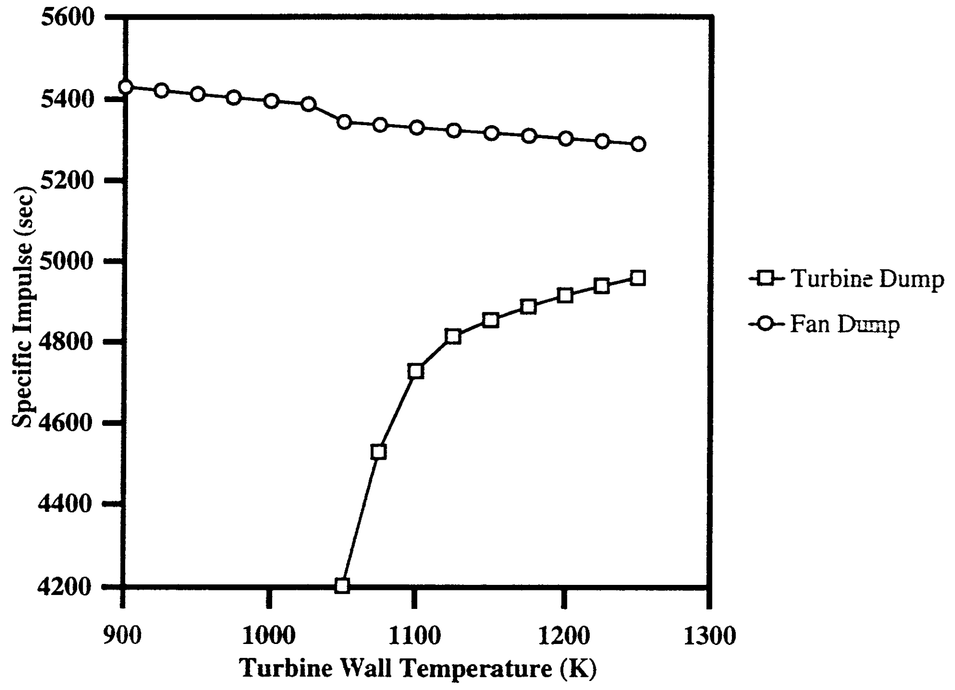

FIGURE 25. Specific Impulse vs. Turbine Wall Temperature for a Low Bypass Engine

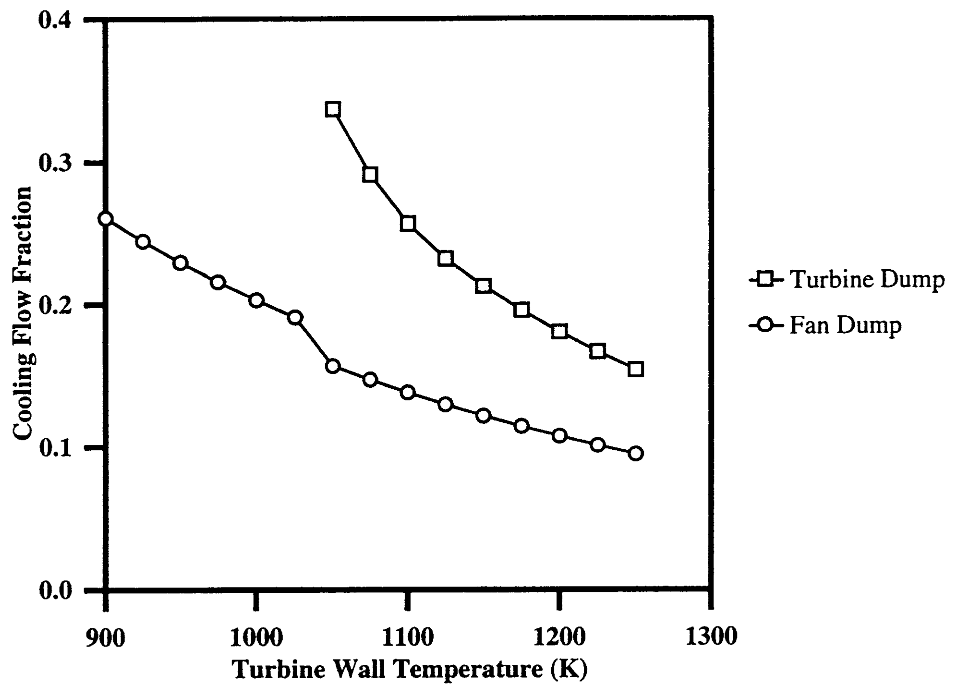

FIGURE 26. Cooling Flow Fraction vs. Wall Temperature for a Low Bypass Engine

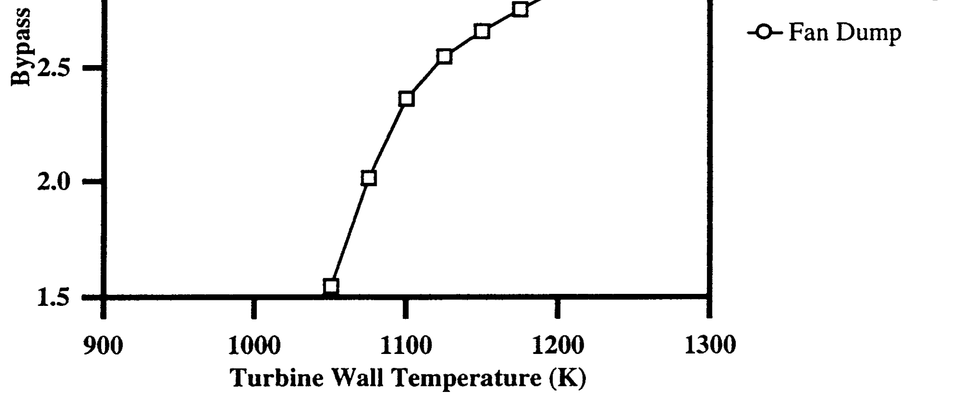

FIGURE 27. Bypass Ratio vs. Turbine Inlet Temperature for a Low Bypass Engine

4.5 Effects of Increasing Turbine Inlet Temperature for a High Bypass Engine at Cruise Conditions

This section gives the results of raising the turbine inlet temperature in engines operating at approximately 30,000 ft., with an ambient temperature of 220 K and an ambient pressure of 26,500 Pa, flying at a Mach number of 0.8. The results are presented without further discussion since they are very similar in nature to those at takeoff conditions.

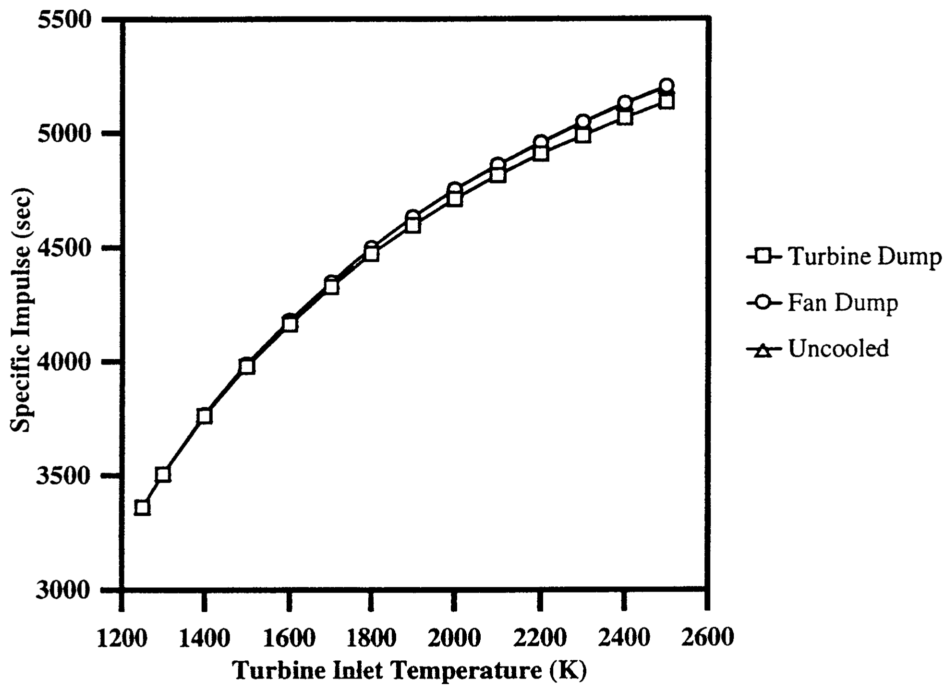

FIGURE 28. Specific Impulse vs. Turbine Inlet Temperature at Cruise Conditions

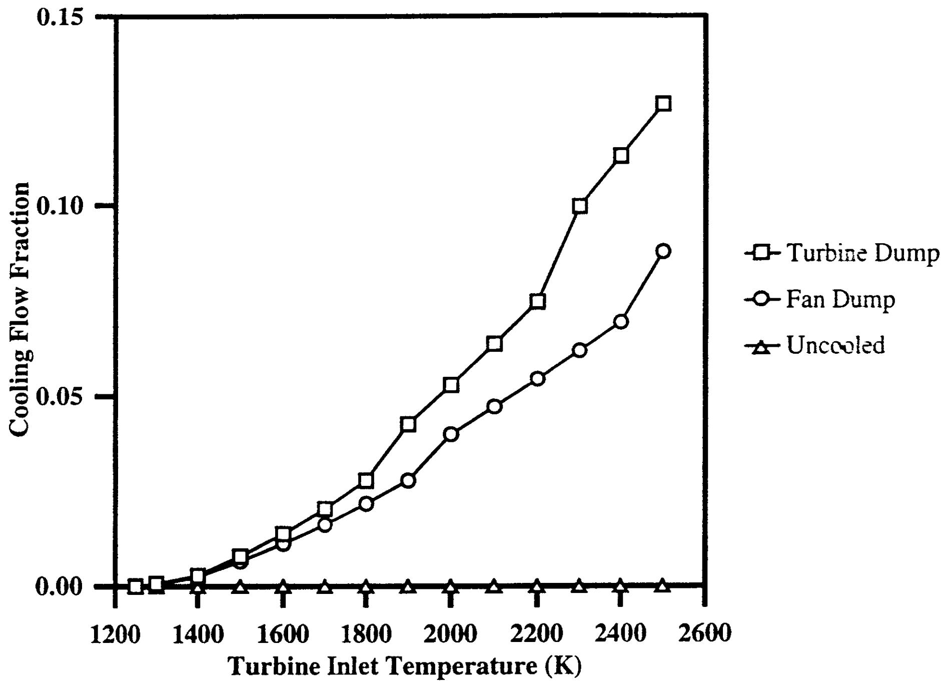

FIGURE 29. Cooling Flow Fraction vs. Turbine Inlet Temperature at Cruise Conditions

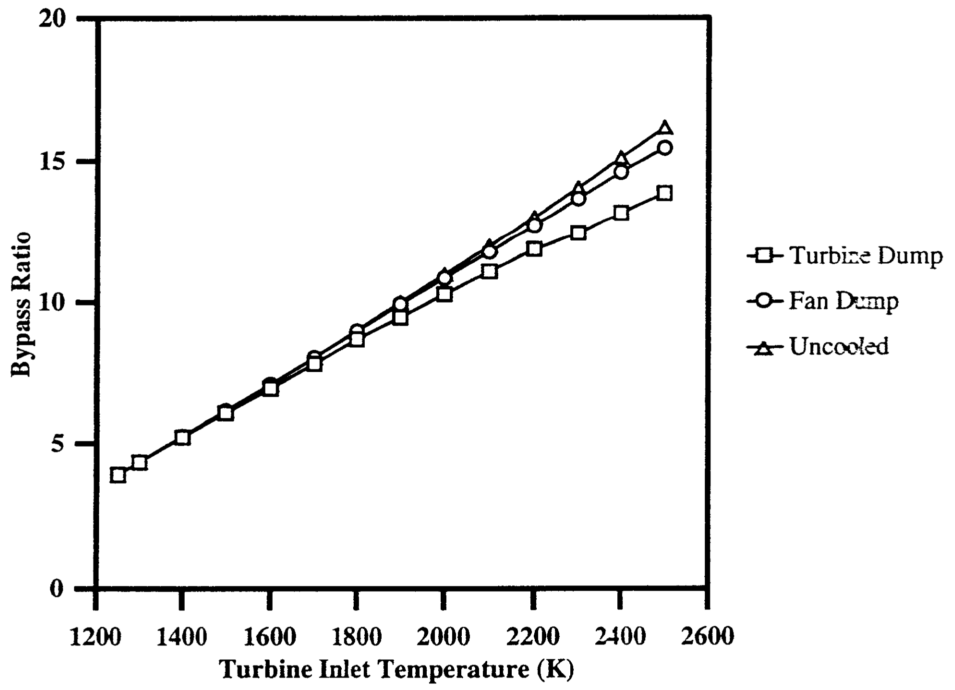

FIGURE 30. Bypass Ratio vs. Turbine Inlet Temperature at Cruise Conditions

4.6 Effects of Decreasing Wall Temperature for a High Bypass Engine at Cruise Conditions

This section gives the results of decreasing the turbine wall temperature in engines operating at cruise conditions. As in the previous section, the results are presented without discussion.

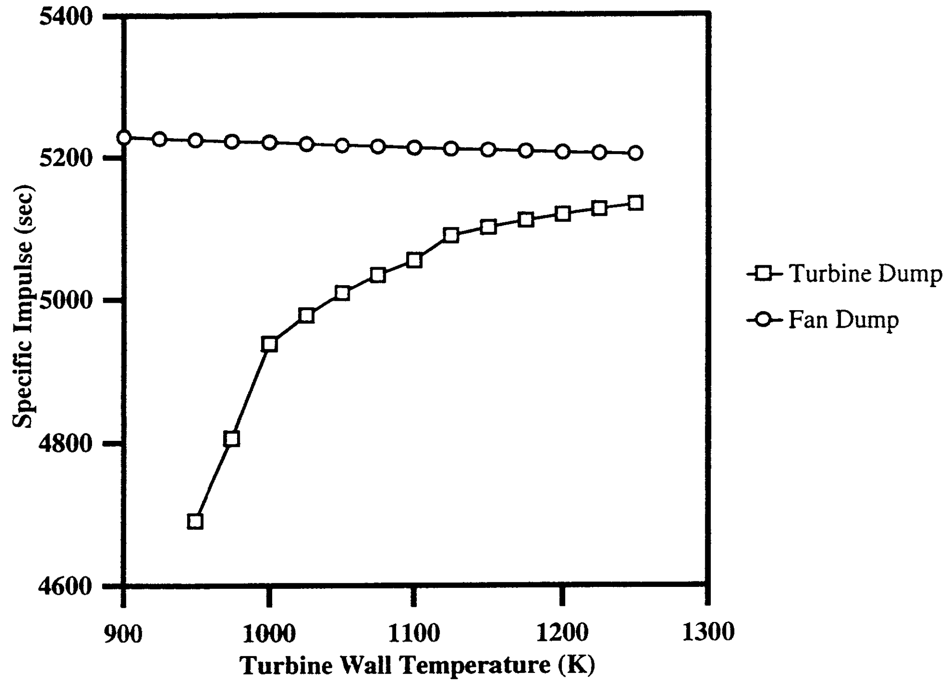

FIGURE 31. Specific Impulse vs. Turbine Wall Temperature at Cruise Conditions

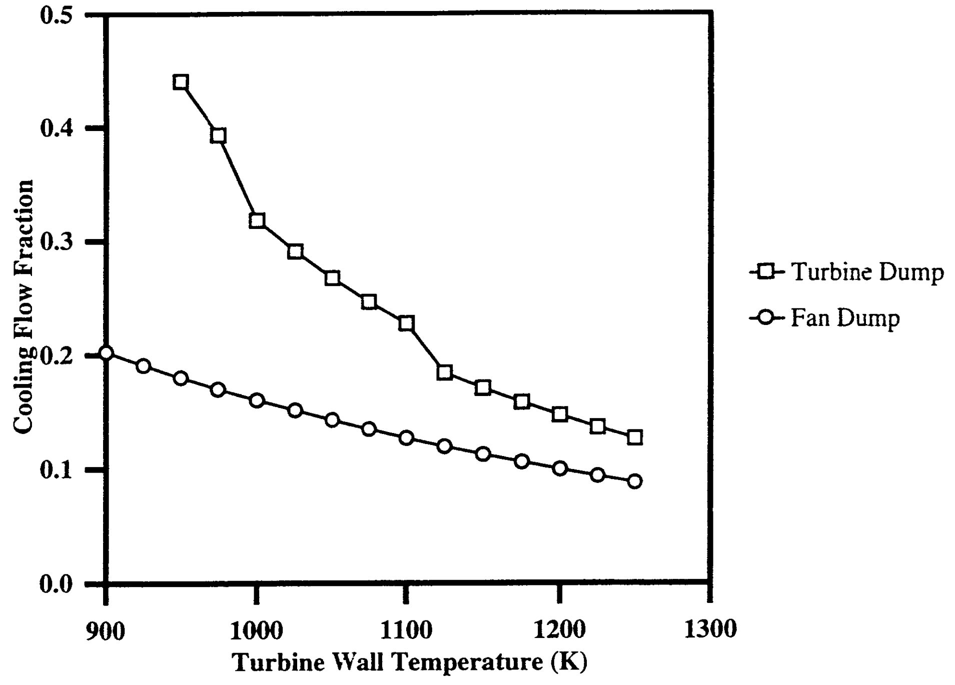

FIGURE 32. Cooling Flow Fraction vs. Turbine Wall Temperature at Cruise Conditions

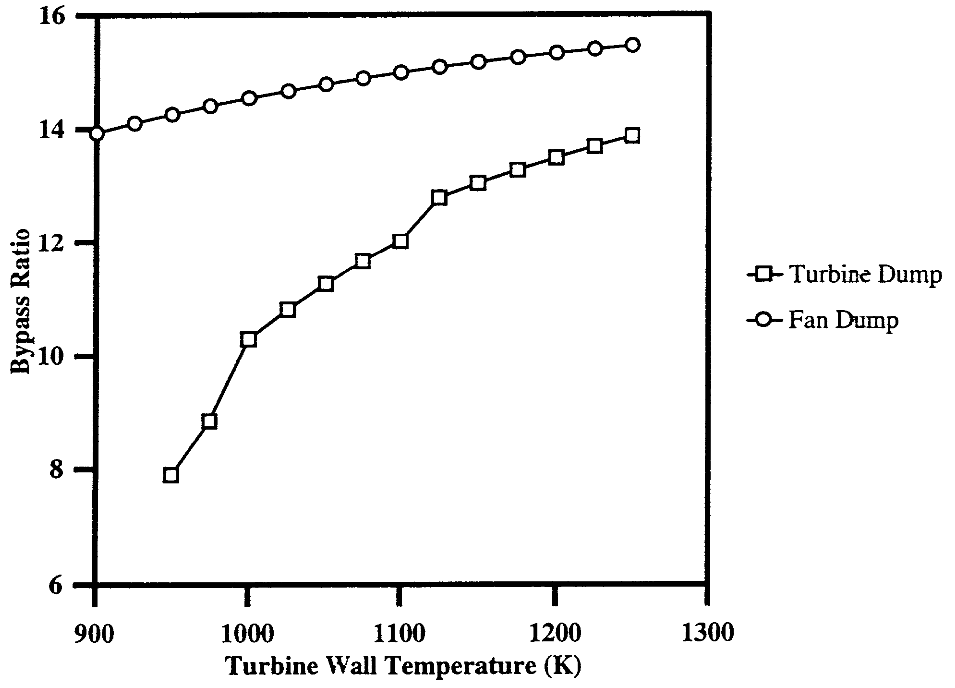

FIGURE 33. Bypass Ratio vs. Turbine Wall Temperature at Cruise Conditions

4.7 Conclusions From Turbofan Calculations

With the given conditions on the exit velocities, it is clear that increasing the turbine inlet temperature past 1650 K significantly improves the specific impulse (by approximately 25% at T_{t_4}=2500 K for the evaporatively cooled high bypass engine at takeoff) at a constant thrust level. The need for cooling changes the operation of the engines, most notably in the ducting for the cooling and in the size of the bypass ratio, but the resulting specific impulse is not notably affected. For the film cooled engines, however, the very large fraction of core mass flow that must be diverted to cool the turbine, even at relatively low turbine inlet temperatures, make this type of cooling unfeasible for cooling at very high temperatures. Again it must be noted that these calculations are meant to provide a comparison between film cooling and internal cooling in a similar engine model, not necessarily to accurately quantify the performance of a real film cooled engine.

Although the benefits of moving to higher turbine inlet temperatures is clear, given the technology to cool the engine at these temperatures, there are drawbacks. The most notable drawback of such high turbine inlet temperatures is the amount of cooling air required, and the engine layout redesign that may be necessary to accommodate it. This is especially true of the cooling scheme in which the cooling air is sent back to the bypass duct and added to the flow there. In general, this was the cooling scheme which performed the best with the set of assumptions made.

Another result of these calculations is that, at least for the engines with bypass coolant injection, there is little performance penalty in dropping the turbine wall temperature so that a wider range of materials can be used in designing the turbine. Once again, the catch is that up to 20% of the core flow (for T_{t_4}=2500 K and T_w=900 K) must be brought from the compressor to the turbine and then back to the bypass duct.

5.0 Results of Regenerated Gas Turbine Analysis and Conclusions

This section describes the results of the regenerated gas turbine analysis given in Section 3.0, Regenerated Ground Based Engine. As described in that section, this analysis was tailored towards stationary ground-based engines. For each set of operating conditions, the value for the regenerator effectiveness was chosen such that the thermal efficiency of the engine was maximized. Finally, the compressor pressure ratio was held constant at 10.

5.1 Effects of Increasing Turbine Inlet Temperature

In this section, results are given which show the effects of increasing the turbine inlet temperature from 1300 K to 2500 K. For the cooled engines, the wall temperature was held constant at 1250 K.

The maximum thermal efficiency of the uncooled regenerated engine increases by almost 50% when the turbine inlet temperature is increased to 2500 K, as shown by Figure 34, "Maximum Thermal Efficiency vs. Turbine Inlet Temperature." When the turbine walls are cooled to a constant temperature, there is a penalty involved in thermal efficiency which increases with turbine inlet temperature to 4.2% at 2500 K.

Figure 35, "Regenerator Effectiveness at Maximum Efficiency vs. Turbine Inlet Temperature," shows the effectiveness required to achieve the thermal efficiencies given in Figure 34. The larger the quantity of heat transferred from the turbine exhaust to the compressor exhaust, however, the larger the associated pressure loss. For this reason, the regenerator effectiveness (an indicator of this quantity of heat) which produces the highest thermal efficiency is less than one. In both the cooled and the uncooled engines, it ranges from about 0.46 at a turbine inlet temperature of 1300 K to approximately 0.7 at T_{t_4} equal to 2500 K.

The nondimensionalized power output of the engine (P) rises steadily with turbine inlet temperature, with no difference between the cooled and the uncooled cases (Figure 36, "Power Output at Maximum Efficiency vs. Turbine Inlet Temperature"). The difference in the thermal efficiencies between the cooled and the uncooled engines is therefore attributable to the fact that the fuel-air ratio is lower for the uncooled engines than it is for the cooled engines. The fuel-air ratio is shown in the plot in Figure 37, "Fuel-Air Ratio at Maximum Efficiency vs. Turbine Inlet Temperature."

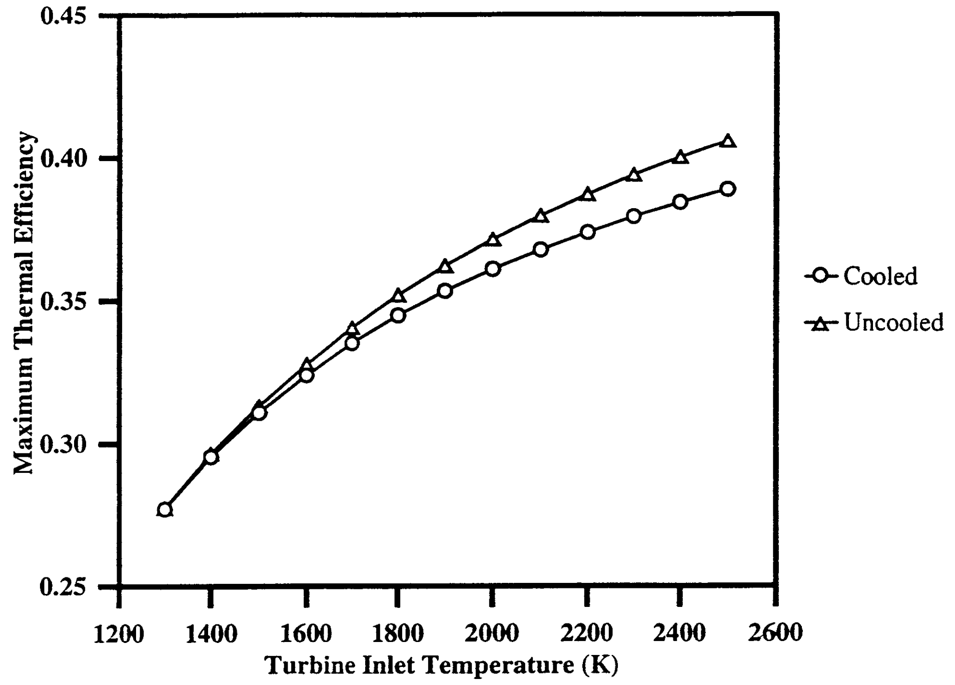

FIGURE 34. Maximum Thermal Efficiency vs. Turbine Inlet Temperature

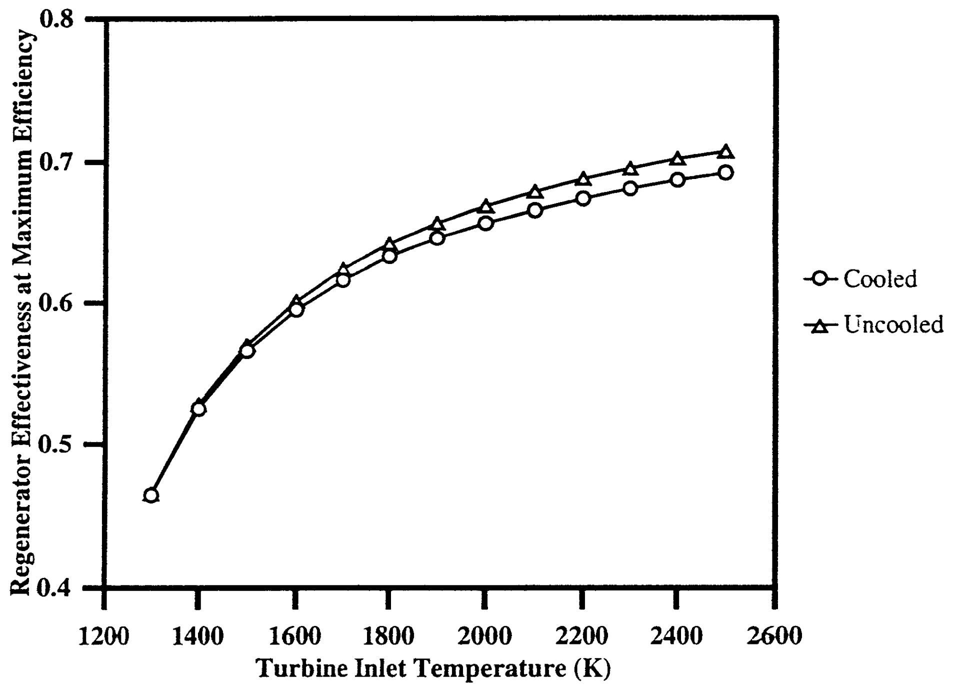

FIGURE 35. Regenerator Effectiveness at Maximum Efficiency vs. Turbine Inlet Temperature

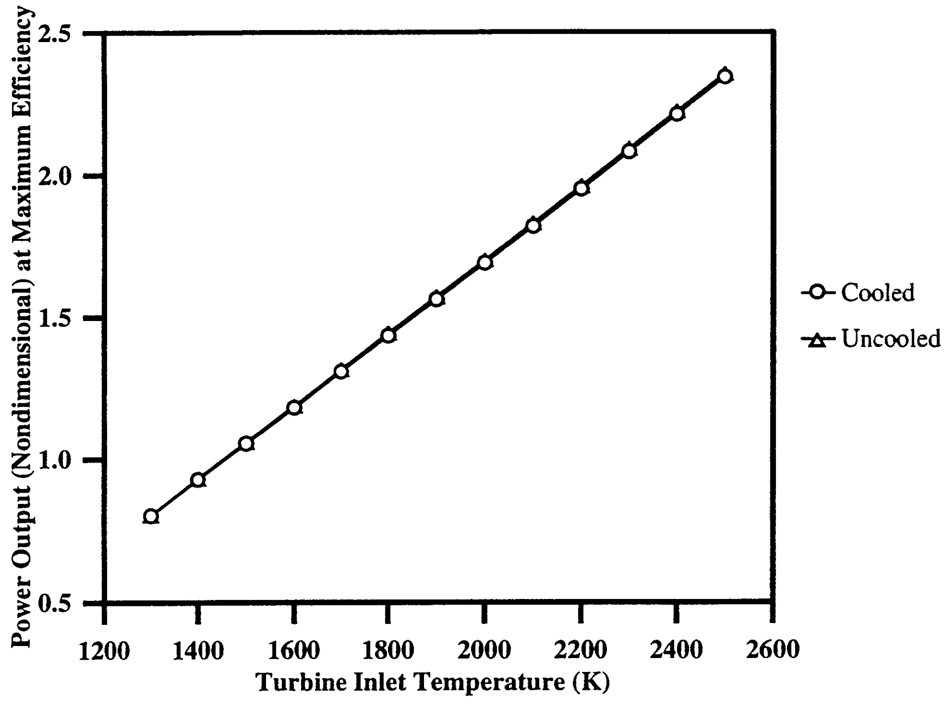

FIGURE 36. Power Output at Maximum Efficiency vs. Turbine Inlet Temperature

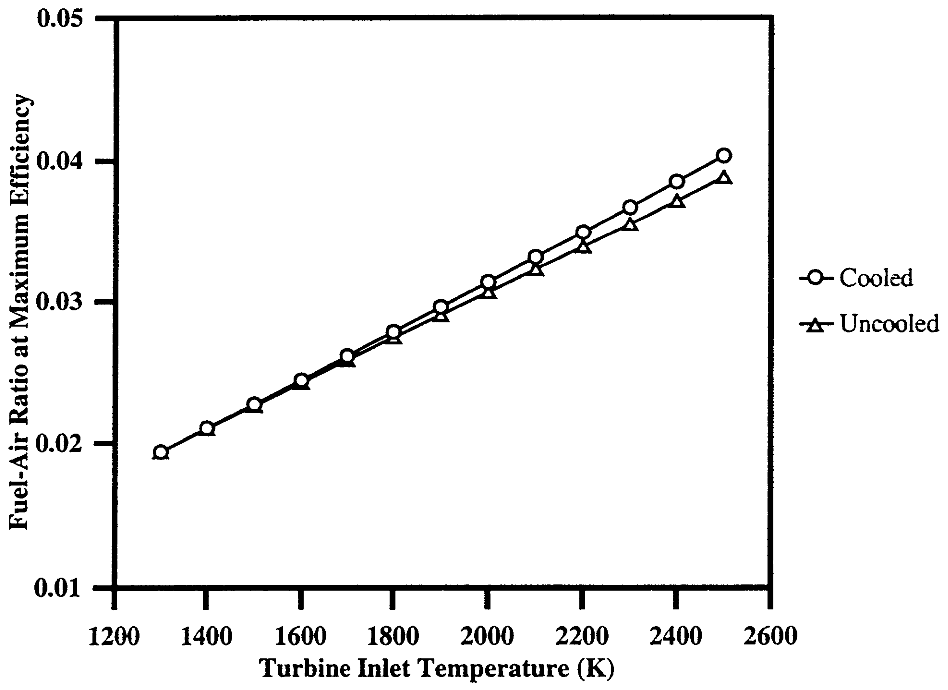

FIGURE 37. Fuel-Air Ratio at Maximum Efficiency vs. Turbine Inlet Temperature

5.2 Effects of Decreasing Wall Temperature

The effects of decreasing the wall temperature at a constant turbine inlet temperature are not as dramatic for the regenerated engines as they are for the aircraft engines. This reflects the fact that in a ground based engine, the cooling fluid does not need to be taken from the core engine flow, but can be on an entirely separate loop. Thus the only penalties involved in cooling the ground based engines are those resulting from the removal of energy from the turbine flow.

The effects of decreasing the turbine wall temperature can be seen in Figure 38, "Maximum Thermal Efficiency vs. Turbine Wall Temperature," Figure 39, "Regenerator Effectiveness at Maximum Efficiency vs. Turbine Wall Temperature," Figure 40, "Power Output at Maximum Efficiency vs. Turbine Wall Temperature," and Figure 41, "Fuel-Air Ratio at Maximum Efficiency vs. Turbine Wall Temperature."

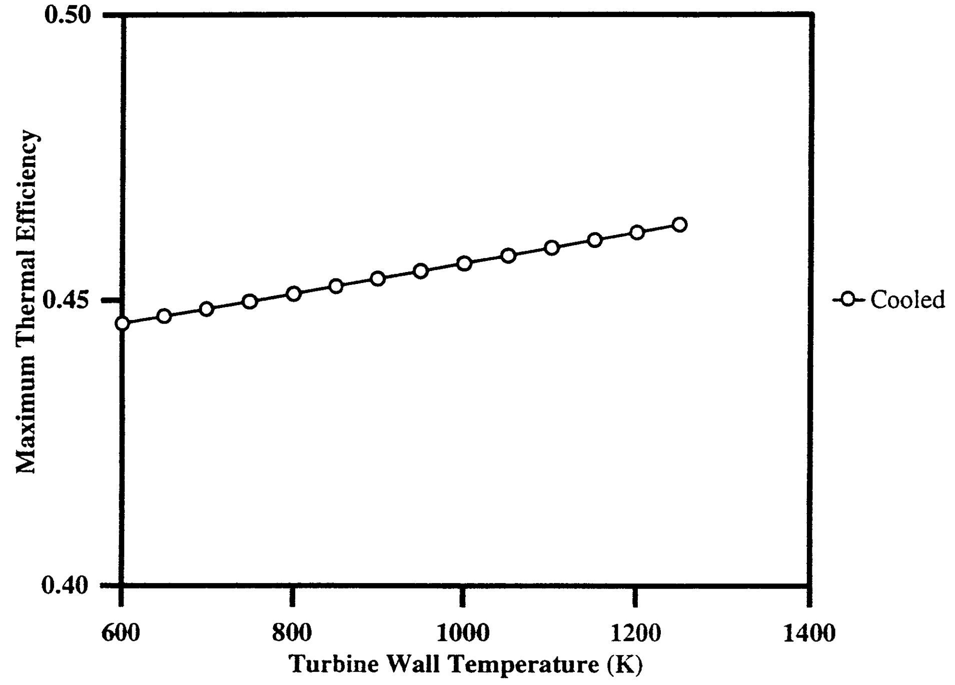

FIGURE 38. Maximum Thermal Efficiency vs. Turbine Wall Temperature

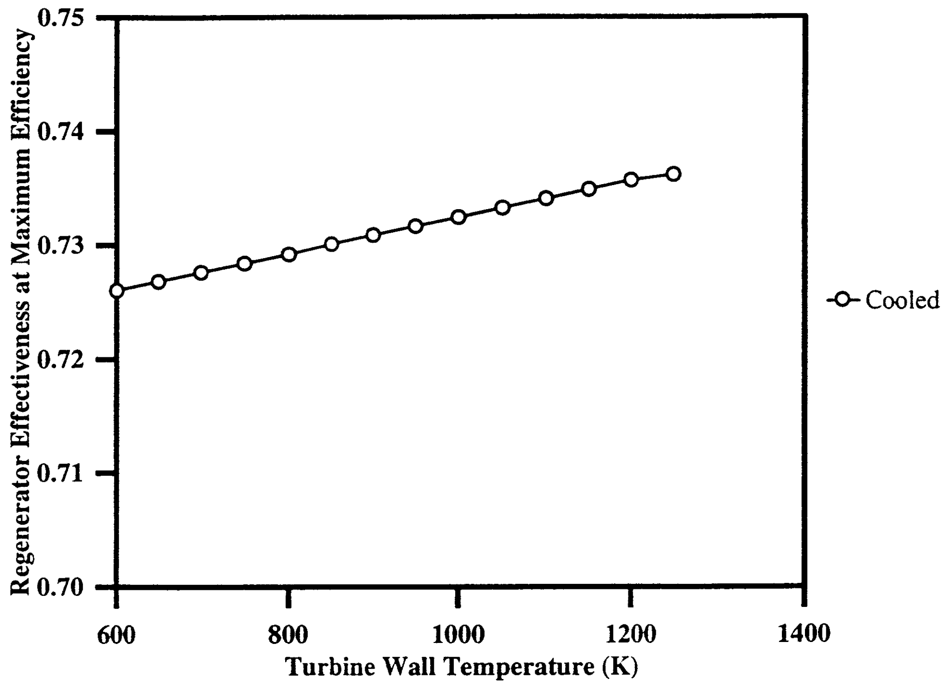

FIGURE 39. Regenerator Effectiveness at Maximum Efficiency vs. Turbine Wall Temperature

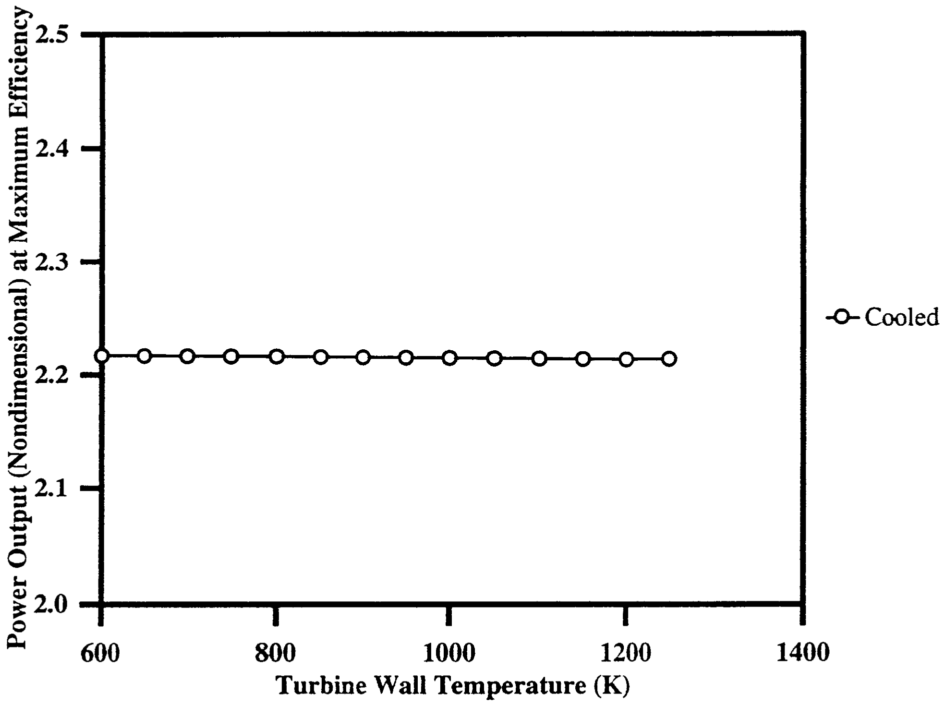

FIGURE 40. Power Output at Maximum Efficiency vs. Turbine Wall Temperature

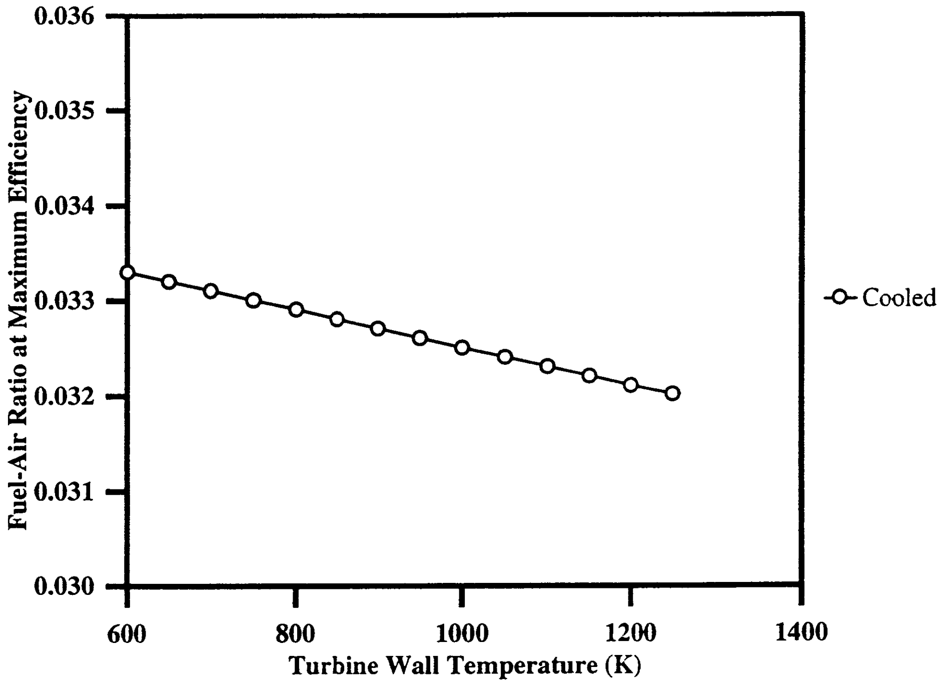

FIGURE 41. Fuel-Air Ratio at Maximum Efficiency vs. Turbine Wall Temperature

5.3 Conclusions From Regenerated Engine Calculations

The benefits of evaporative cooling are more clear for the ground-based engines than for the aircraft engines. Since it has been assumed that these engines are stationary, the cooling system is separate from the gas cycle, and the problems of taking cooling air out of the main flow path or of where to reintroduce it don't exist. For example, the main impediment to dropping the turbine wall temperature for the aircraft engines, which was the increasingly large quantity of cooling air required, is not a problem for the ground based engines. Therefore, decreasing the turbine wall temperature from 1250 K to as low as 600K entails a penalty in maximum thermal efficiency of less than 5%. On the other hand, the rise in thermal efficiency due to increasing the turbine inlet temperature from 1650K to 2500 K, which is made possible by the evaporative cooling, is almost 19%.

List of References

1. J. L. Kerrebrock, *Aircraft Engines and Gas Turbines*. MIT Press, 1992.

Appendix A: Alternative Formulation of Cooling Flow Fraction

Section 2.6, Turbine Characterization, described a model of the turbine which allows us to calculate the fraction ε of the core airflow that must be taken from the compressor to cool the turbine, the amount of work done by the turbine, and the pressure drop across the turbine. With that model, however, it would be unreasonable to express these values in a few equations. This appendix will describe a simpler model which will bring us closer to the goal of obtaining results in closed form. Such a model can make clearer the effect of each parameter on operating conditions.

This appendix will be divided into two sections. The first section will develop expressions for the cooling flow fraction which involve summations, and the second will describe an approximation to these summations using integrals.

A.1 Cooling Flow Requirements

This section will develop analytical expressions for the amount of cooling air required to maintain the turbine walls at a given temperature. Of the two cooling schemes considered in this report, the simpler to express analytically is the case in which the cooling air accepts heat from the turbine and is then taken out to the bypass duct and injected into the flow at that point. This case will be analyzed first. The second option, which is injecting the cooling air directly into the turbine flow, will then be analyzed.

A.1.1 Cooling Air Injected Into Bypass Flow

We will use as a starting point Equation (31), which describes the cooling flow required for one stator cascade as a fraction of the main flow at that cascade, and Equation (52), which describes the same thing for the rotor cascade. We denote the local cooling flow fraction for the stator by ε'_s and that for the rotor by ε'_r. That is,

$$\varepsilon'_s = \left(\frac{\dot{m}_c}{\dot{m}}\right)_s = \mathrm{St}r\left(\frac{c_{p_t}}{c_{p_c}}\right)\left(\frac{T_{t_s} - T_w}{T_w - T_c}\right), \tag{78}$$

$$\varepsilon'_r = \left(\frac{\dot{m}_c}{\dot{m}}\right)_r = \mathrm{St}r\left(\frac{c_{p_t}}{c_{p_c}}\right)\left(\frac{\alpha_t T_{t_r} - T_w}{T_w - T_c}\right). \tag{79}$$

In the case where no cooling air is added to the turbine flow, these equations can be combined to yield a single expression for the ratio of cooling flow for the nth stage to the turbine flow (equal to $\dot{m}_4$), which we will call ε'_n:

$$\varepsilon'_n = \frac{\dot{m}_{c,n}}{\dot{m}_4} = \frac{\mathrm{Str}}{T_w - T_c c_{p_c}} c_{p_t} ((1 + \alpha_T - \alpha_T \mathrm{Str}) T_{t_s} - (2 - \alpha_T \mathrm{Str}) T_w) . \tag{80}$$

As in Section 2.6, Turbine Characterization, this formulation assumes that the coolant exiting the blade is at T_w, which cannot be exactly true in reality.

To find the total cooling flow required as a fraction of $\dot{m}_4$, which we will denote ε', we can sum ε'_n over all of the N stages:

$$\varepsilon' = \sum_{n=1}^{N} \varepsilon'_n . \tag{81}$$

However, to find ε, the total cooling flow as a fraction of total core flow, we need to multiply ε' by $\dot{m}_4/\dot{m} = 1 + f - \varepsilon$, giving us

$$\begin{aligned} \varepsilon &= \frac{(1+f)\varepsilon'}{1+\varepsilon'} \\ &= \frac{(1+f) \sum_{n=1}^{N} \varepsilon'_n .}{1 + \sum_{n=1}^{N} \varepsilon'_n} \end{aligned} \tag{82}$$

We could use ε' as an approximation to ε. From Equation (82) we see that if both f and ε' are small compared to 1, Equation (82) reduces to $\varepsilon = \varepsilon'$. Likewise, if they are approximately equal, Equation (82) again reduces to $\varepsilon = \varepsilon'$. This simplification may therefore be made for cases in which the turbine requires only moderately cooling.

We can estimate the number of stages required by noting that $\tau_s^N = \tau_t$, or

$$N = \frac{\ln \tau_t}{\ln \tau_s} , \tag{83}$$

where τ_s is the total temperature ratio across one turbine stage. The total temperature in stationary coordinates at the beginning of any given stage is approximately

$$T_{t_s} = T_{t_4} \tau_s^{n-1} . \tag{84}$$

Using Equation (85), the summations of Equation (81) and Equation (82) can now be written

$$\sum_{n=1}^{N} \varepsilon'_n = \frac{\mathrm{Str}}{T_w - T_c c_{p_c}} c_{p_t} \left((1 + \alpha_T - \alpha_T \mathrm{Str}) T_{t_4} \sum_{n=1}^{N} \tau_s^{n-1} - (2 - \alpha_T \mathrm{Str}) N T_w \right) . \tag{85}$$

A.1.2 Cooling Air Injected Into Turbine

We begin by noting that the argument that led to Equation (78) and Equation (79) for the local cooling flow fraction at each cascade applies equally for both cooling schemes. In considering the cooling flow required as a fraction of the total mass flow, however, we now have to take into account the fact that the local turbine mass flow changes with each stage, depending on the amount of cooling required by the previous stages. As a fraction of the turbine inlet mass flow, the cooling required at each stage is

$$\varepsilon''_{s,n} = \varepsilon'_{s,n}(1+\varepsilon'_{r,n-1})\,(1+\varepsilon'_{s,n-1})\ldots(1+\varepsilon'_{r,1})\,(1+\varepsilon'_{s,1})\,, \tag{86}$$

$$\varepsilon''_{r,n} = \varepsilon'_{r,n}(1+\varepsilon'_{s,n})\,(1+\varepsilon'_{r,n-1})\ldots(1+\varepsilon'_{r,1})\,(1+\varepsilon'_{s,1})\,. \tag{87}$$

The total cooling flow as a fraction of the turbine inlet mass flow would therefore be a summation of Equation (86) and Equation (87) over all N stages, resulting in

$$\begin{aligned}\varepsilon'' &= \sum_{n=1}^{N}(\varepsilon''_{s,n}+\varepsilon''_{r,n})\\ &= \varepsilon'_{s,1}+\varepsilon'_{r,1}(1+\varepsilon'_{s,1})+\ldots+\varepsilon'_{r,N}(1+\varepsilon'_{s,N})\,(1+\varepsilon'_{r,N-1})\ldots(1+\varepsilon'_{s,1})\,.\\ &= \prod_{n=1}^{N}(1+\varepsilon'_{s,n})\,(1+\varepsilon'_{r,n})-1\end{aligned} \tag{88}$$

We can convert this product into a summation by taking the natural log, so that

$$\varepsilon'' = \exp\left[\sum_{n=1}^{N}\ln(1+\varepsilon'_{s,n})+\sum_{n=1}^{N}\ln(1+\varepsilon'_{r,n})\right]-1\,. \tag{89}$$

As in the previous section, to find the cooling flow as a fraction of the total flow, the above equation must be multiplied by $\dot{m}_4/\dot{m} = 1+f-\varepsilon$, giving

$$\varepsilon = \frac{(1+f)\,\varepsilon''}{1+\varepsilon''}\,. \tag{90}$$

A.2 Integral Approximation of Summations

We can obtain continuous expressions for ε by approximating the summation in Equation (85) and in Equation (89) with integrals, and then evaluating these integrals. The equations thus derived can then be used to find approximate expressions for various quantities, including the variation of required cooling flow fraction with turbine inlet temperature, turbine wall temperature, or cooling flow temperature, for example.

A.2.1 Cooling Air Injected Into Bypass Flow

The summation in Equation (85) looks something like the area in the rectangles of Figure 42, "Continuous Approximation to Discrete Summation," and can be approximated by the integral of the equation for the curved line, $y = \tau_s^{n-1}$, in the figure between the limits of 1/2 and N+1/2. This approximation should be close[1], especially as τ_s approaches 1:

$$\int_{1/2}^{(N+1/2)} \tau_s^{n-1} dn = \int_{1/2}^{(N+1/2)} e^{(n-1)\ln\tau_s} = \frac{1}{\ln\tau_s} e^{(n-1)\ln\tau_s}\Bigg|_{1/2}^{N+1/2}$$
$$= \frac{1}{\ln\tau_s}\left(\tau_s^{N-\frac{1}{2}} - \frac{1}{\sqrt{\tau_s}}\right) = \frac{\tau_t - 1}{\sqrt{\tau_s}\ln\tau_s} \quad . \tag{91}$$

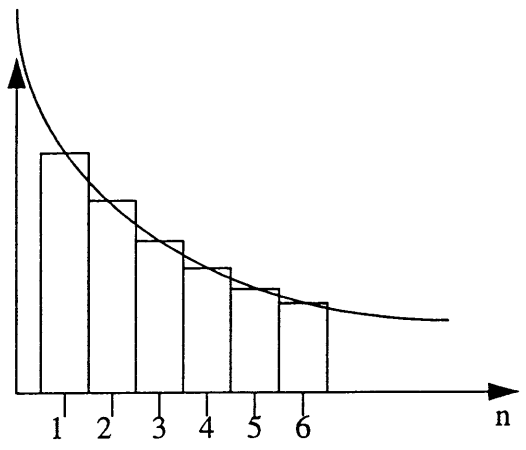

FIGURE 42. Continuous Approximation to Discrete Summation

Substituting this expression back into Equation (85), and using Equation (83) for N, we get

$$\sum_{n=1}^{N} \varepsilon'_n = \frac{Str}{T_w - T_c c_{p_c}} c_{p_t} \left[\frac{(1+\alpha_T - \alpha_T Str)\, T_{t_4} (\tau_t - 1)}{\sqrt{\tau_s}\ln\tau_s} - T_w \frac{\ln\tau_t}{\ln\tau_s}\right] . \tag{92}$$

A.2.2 Cooling Air Injected Into Turbine

In order to approximate Equation (89) with an integral expression, we must first simplify it a little further. The natural logarithms can be expressed as a series:

1. To check the accuracy of the approximation, we can evaluate both the integral and the summation for τ_s =0.9 and N=3, and find that the summation equals 2.71 while the integral is equal to 2.711.

$$\ln(1+\varepsilon') = \varepsilon' - \frac{1}{2}\varepsilon'^2 + \frac{1}{3}\varepsilon'^3 - \frac{1}{4}\varepsilon'^4 + \dots \tag{93}$$

We can estimate the maximum value for ε' and keep as many terms in this series as desired to achieve a given accuracy. For the purposes of this appendix, which are illustrative only, it will be sufficient to keep just the first term. For actual calculations of very heavily cooled turbines, where the maximum cooling flow fraction for a stage might be on the order of 0.1, the first three terms might be kept.

The first summation in Equation (89) then becomes

$$\begin{aligned} \sum_{n=1}^{N} \ln(1+\varepsilon'_{s,n}) &\approx \sum_{n=1}^{N} \varepsilon'_{s,n} \\ &= a\sum_{n=1}^{N} \tau_s^{n-1} + Nb \end{aligned}, \tag{94}$$

where

$$\begin{aligned} a &= \xi T_{t_4} \\ b &= -\xi T_w \\ \xi &= \mathrm{St}r\frac{c_{p_t}}{c_{p_c}}\frac{1}{T_w - T_c} \end{aligned} . \tag{95}$$

In order to evaluate the second summation in Equation (89), we need to express ε'_r in terms of n:

$$\varepsilon'_r = \frac{c\tau_s^{n-1}}{d+g\tau_s^{n-1}} + \frac{h}{d+g\tau_s^{n-1}} + k, \tag{96}$$

where

$$\begin{aligned} c &= \xi\alpha_T\left(1+\xi\frac{c_{p_c}}{c_{p_t}}T_c\right)T_{t_4} \\ d &= 1-\xi T_w \\ g &= \xi T_{t_4} \\ h &= -\xi^2\alpha_T\frac{c_{p_c}}{c_{p_t}}T_wT_c \\ k &= -\xi T_w \end{aligned} \tag{97}$$

Then, using the same approximation for the logarithm as in Equation (94), this summation becomes

$$\begin{aligned} \sum_{n=1}^{N}\ln(1+\varepsilon'_{r,n}) &\approx \sum_{n=1}^{N}\varepsilon'_{r,n} \\ &= c\sum_{n=1}^{N}\frac{\tau_s^{n-1}}{d+g\tau_s^{n-1}}+h\sum_{n=1}^{N}\frac{1}{d+g\tau_s^{n-1}}+Nk \end{aligned} \tag{98}$$

Equation (94) and Equation (98) can now be substituted into Equation (89) to yield an expression for the total cooling flow as a fraction of the turbine inlet mass flow:

$$\begin{aligned} \varepsilon'' &= \frac{\dot m_c}{\dot m_4} \\ &= \exp\left[a\sum_{n=1}^{N}\tau_s^{n-1}+Nb+c\sum_{n=1}^{N}\frac{\tau_s^{n-1}}{d+g\tau_s^{n-1}}+h\sum_{n=1}^{N}\frac{1}{d+g\tau_s^{n-1}}+Nk\right]-1 \end{aligned} \tag{99}$$

The summations in Equation (99) can be approximated by integrals in the same manner as in the previous section, yielding as a final result

$$\varepsilon'' = \frac{\dot{m}_c}{\dot{m}_4}$$

$$= \exp\left[a\frac{\tau_t - 1}{\sqrt{\tau_s}\ln\tau_s} + \frac{c}{g\ln\tau_s}\ln\left(\frac{d + \frac{g\tau_t}{\sqrt{\tau_s}}}{d + \frac{g}{\sqrt{\tau_s}}}\right) + \frac{h}{d\ln\tau_s}\ln\left(\frac{d\tau_t + \frac{g\tau_t}{\sqrt{\tau_s}}}{d + \frac{g\tau_t}{\sqrt{\tau_s}}}\right) + N(b + k)\right] - 1 \quad (100)$$

As for the case in which the cooling air is added to the bypass flow, the total cooling flow fraction ε can be found from ε'' by using

$$\varepsilon = \frac{(1 + f)\,\varepsilon''}{1 + \varepsilon''}. \quad (101)$$

Appendix B: Computer Code Description

This appendix will describe the Fortran program which was written in order to obtain numerical results from the analysis described in this report. The code itself will be given in Appendix C.

B.1 Functional Outline of Program

The following is a general outline of the program. At the end of each line, in italics, is the program subunit in which the described action takes place.

```
1.  Set up output file, open input file (main)
2.  loop on input conditions (main)
3.      Read input conditions (main)
4.      Calculate some values directly from the input conditions (main)
5.      Loop on N while $\tau_s - \tau_{s,\,\mathrm{target}}$ not at minimum (SolveConditions)
6.          Loop until the equations are solved (SolveGivenN)
7.              Evaluate equations (EvalModel, Fcall, Ucall)
8.              Approximate the jacobian of the system (SolveGivenN)
9.              Find new guess for solution (Gauss)
10.         End loop on equations (found solution) (SolveGivenN)
11.     End loop on N (SolveConditions)
12.     Calculate final values: $F/(\dot{m}a_0)$, $I$, $\varepsilon_{\mathrm{turbine}}$ (Ftotcall)
13.     Print results to individual file and to table file (main)
14. End loop on input conditions (main)
15. End (main)
```

B.2 Description of Program Subunits

B.2.1 main

The function of main is to read in the inputs for each case, calculate a few simple parameters from the inputs, call the subroutine `Conditions` to do the detailed calculations, and print the results to a file. There is one loop which repeats once for every set of input conditions.

1. Set up output file, open input file

   In total, two files are opened by the program, one for output, and one for input. The output file is where all the results are printed in tabular format, with values separated by commas so that they can be imported into the spreadsheet program Xess. The top line is the column headings, and each succeeding line contains the results for one set of input conditions. Cycle prompts the user for the name of this file.

The input file is named `Cycle.input`, and can have any number of sets of input conditions. Each set is preceded by a logical value indicating whether another set follows in the input file. When this logical value is false, no more lines are read from the input file.

2. Loop on input conditions

   This is simply a loop which repeats for every set of input conditions. As seen from the outline in Appendix Section B.1, all of the important steps are taken from within this loop.

3. Read input conditions

   Each set of input conditions in the file Cycle.input must be in the following format:

Variable Name	Variable Type
CaseFollows	logical
fname	character*20
Tw	real
Tc	real
T0	real
utarget_a0	real
St	real
r	real
delPf_Ptc	real
etapoly	real
P0	real
M0	real
Tt4	real
TurbineDump	logical
FanDump	logical
FilmCooled	logical
ChokedNoz	logical
Uncooled	logical

4. Calculate some values directly from input conditions

   Before beginning the iterations which are required to find the operating point of the engine, some values can be calculated directly from the input values. These are

$$\theta_t = \frac{T_{t_4}}{T_0}$$

$$\theta_0 = 1 + \frac{\gamma_c - 1}{2} M_0^2$$

$$\tau_c = \frac{\sqrt{\theta_t}}{\theta_0}$$

This is the value of τ_c which maximized work for an ideal, uncooled cycle.

$$\pi_c = \tau_c^{\frac{\gamma_c \eta_{\text{poly}}}{\gamma_c - 1}}$$

$$T_{t_2} = \theta_0 T_0$$

$$p_{t_0} = p_0 \theta_0^{\frac{\gamma_c}{\gamma_c - 1}}$$

$$p_{t_4} = p_{t_0} \pi_d \pi_c \pi_b$$

In addition, the conditions of the cooling flow taken at the compressor bleed point are calculated. This is done by estimating the flow pressure at the injection point and stepping back to see what the bleed pressure must be to overcome losses in the piping and arrive at the injection point at the same pressure as the flow.

5.-12. These steps are taken care of from the subroutine Conditions

13. Print results to table

    A line of the table output file is written with the results of the calculations at this set of operating conditions.

14. End loop on input conditions

15. End

    Close up all the files and end program.

B.2.2 SolveConditions

SolveConditions is the subroutine called by main to do the iterative calculations involved in finding the engine's operating point. The portion of this task performed within this subroutine consists in finding the correct number of stages for the turbine. In order to find the operating conditions of an engine configured with a certain number of turbine stages, this routine calls SolveGivenN.

The reason that the number of stages must be found separately from the remainder of the operating conditions has to do with the solution method. A system of nonlinear equations is solved using Newton's method. Allowing N to be determined within this procedure would introduce large discontinuities in the "hyper-surfaces" where a transition occurs from an engine with n turbine stages to one with n+1 or n-1 stages. This would make the system very difficult to solve. Instead, a certain number of stages is assumed, and the rest of the conditions are found. Then the temperature drop per stage which results (τ_s) is compared to a target value, and N is adjusted accordingly.

Once the correct operating point has been found, SolveConditions calls Ftotcall to calculate some of the final output values such as the fan, core, and total thrust, the specific impulse, and an estimate of the efficiency of the turbine and cooling scheme.

5. Loop on N while $\tau_s - \tau_{s,\,\text{target}}$ not at minimum

   The number of turbine stages N is adjusted until the stage temperature ratio τ_s is as close as possible to a given target value, $\tau_{s,\,\text{target}}$. This process consists of the following steps:

   a. Take initial guess of N=5 and solve for operating conditions.
   b. If the resulting τ_s is less than $\tau_{s,\,\text{target}}$, increase N and solve, otherwise decrease N and solve.
   c. keep adjusting N until $|\tau_s - \tau_{s,\,\text{target}}|$ increases, or $\tau_s - \tau_{s,\,\text{target}}$ changes sign.

6.-10. These steps are taken care of from the subroutine SolveGivenN

11. End loop on N

12. Calculate final values: $F/(\dot{m}a_0)$, I, $\varepsilon_{\text{turbine}}$

    Once all of the details of turbine operation are settled, the operating point is characterized by finding values for the total thrust, specific impulse, and a rough measure of turbine efficiency which includes the effects of mixing the cooling air back into the flow.

    The equations used in this step can be found in Section 2.1, Thrust and Specific Impulse.

B.2.3 SolveGivenN

Depending on the input conditions, there may be up to five coupled nonlinear equations that need to be solved simultaneously, each in the from $f_n(\bar{x}) = 0$. The subroutine SolveGivenN evaluates each of these functions f_n at the current guess for the solution, $\bar{x}$, by calling EvalModel. Then it finds an approximations to the Jacobian matrix, J, where the i,j element is $\partial f_i/\partial x_j$. It then calls the subroutine Gauss to solve the equation $J\bar{s} = -\bar{f}$, where $\bar{s}$ is the step to take in the solution vector $\bar{x}$, and $\bar{f}$ is the vector of the f_n at the current guess for $\bar{x}$.

6. Loop until the equations are solved.

   The equations to be solved are:

   a. Compressor-fan-turbine work balance:
      $f_1 = c_1\,(\text{turbine work} - \text{compressor and fan work}) = 0$

   b. Core exit velocity equal to a fixed target velocity: $f_2 = u_6 / u_{\text{target}} - 1 = 0$

   c. Bypass exit velocity equal to a fixed target velocity:
      $f_3 = u_8 / u_{\text{target}} - 1 = 0$

   d. Combustor energy balance:
      $f_4 = c_4\,(\varepsilon \text{ required to cool turb.} - \varepsilon \text{ given by comb. ener. bal.}) = 0$

   e. Pressure at cooling flow injection point equal to flow pressure at that point:
      $f_5 = c_5\,(P_{\text{at injection}} - P_{\text{cooling}}) = 0$

   where the c_n are constants. The unknowns are:

   a. τ_t

   b. τ_f

   c. α

   d. f

   e. $\pi_{c'}$

   An initial guess is taken for the unknowns, the surfaces f_n are approximated by planes at that point, and the intersection of these planes becomes the next guess. This continues until the absolute values of all of the functions f_n are below a certain tolerance.

7. Evaluate equations

   Values for the unknowns are fixed in the iteration either by the initial guess or by the most recent linear approximation of the functions f_n. The subroutine EvalModel is then called to evaluate the vector $\vec{f}$. This subroutine in turn calls Fcall, which follows the procedure given in Section 2.6, Turbine Characterization, to find the turbine operating conditions, and Ucall, which uses the results of Fcall to find the core and bypass exit velocities as explained in Section 2.2, Exit Temperatures and Velocities.

8. Approximate the Jacobian of the system

To find an approximation to the Jacobian matrix, J, each of the variables is individually incremented by dx_i, and EvalModel is called. Then the i,j element, $\partial f_i/\partial x_j$, is approximately $[f_i(\bar{x}+dx_j) - f_i(\bar{x})]/dx_j$.

9. Find new guess for solution

   The next guess for $\bar{x}$ is $\bar{x}+\bar{s}$, where $\bar{s}$ is the required step at the current guess. $\bar{s}$ is found from the equation $J\bar{s} = -\bar{f}$. The subroutine Gauss solves this equation by using gaussian elimination with partial pivoting

10. End loop on equations (found solution)

B.2.4 EvalModel

EvalModel simply calls Fcall and Ucall, then evaluates the functions f_n described above and returns.

B.2.5 Fcall

Fcall follows the procedure given in Section 2.6, Turbine Characterization, to find the operating conditions for the turbine, stepping once through the equations given there for each of the N turbine stages.

B.2.6 Ucall

Once the turbine operating conditions are known, including the required quantity of cooling air and the pressure drop across the turbine, the exit velocities in the core and the bypass ducts may be calculated. Ucall does this in two main steps:

a. Calculate π_{cf}, τ_{cf}

   For the cases where the cooling air is injected into the bypass duct, this section calculates the effect on the total pressure and temperature of the bypass flow. The equations for this step can be found in Section 2.5, Effect on Fan Performance of Cooling Air Injected Into Fan.

b. Calculate exit velocities and temperatures

   Finally, Ucall calculates the velocity of the exit jets for use in the α and τ_f loops in Conditions. The equations for this step can be found in Section 2.2, Exit Temperatures and Velocities.

B.2.7 Ftotcall

This routine is called by SolveConditions once the values of N and of all the unknowns in $\bar{x}$ have been settled on. It calculates final output values for thrust and specific impulse, as given in Section 2.1, Thrust and Specific Impulse.

Appendix C: Computer Code

C.1 Main Program File: A4.f

```
program cycle

      implicit none

      include 'C7.inc'

      character*20 fname,fnamebig, ch,chlast
      logical CaseFollows, firstloop
      double precision dum,e
      parameter (e=2.718281828)

      firstloop = .true.
      ch = 'ok'
      chlast = 'ok'


c-- open input file and read logical variable CaseFollows
      open (7,file='Cycle.input',status='OLD')
      read (7,*) CaseFollows

c-- open big table form output file (name ends in .csv for xess input)
      write (6,'("What is the big output file name? Remember to add ",
     *          "extension .csv for Xess comma separated values.")')
      read (5,*) fnamebig
      open (9,file=fnamebig,status='UNKNOWN')
      write (9,'(70(a20,","))')
     *     'FileName','Tw','Tc','Tcinp','T0','utarget_a0','St','r',
     *     'delPf_Ptc','etapoly','P0','M0','Tt4','TurbineDump',
     *     'FanDump','FilmCooled','ChokedNoz','Uncooled','tht','th0',
     *     'tauc','pic','Tt2','Pt0','Pt4','taucp','picp','Ptc',
     *     'taut','f','eps','W','pit','N','taus','tauf','alpha','pif',
     *     'picf','PatInj','PtcatInj','M8','taucf','T8_T0','u8_a0',
     *     'P8_P0','F8_ma0','M6','T6_T0','u6_a0','P6_P0','F6_ma0',
     *     'Ftot_mtota','Isp','TurbEff','FilmEff'


c-- Set up initial guesses
      tautInit = 0.5
      fInit = 0.03
      taufInit = 1.2
      alphaInit = 10.0
      picpInit = 10.0

c-- Set up dx's
      dtaut = tautInit/10000.
      df = fInit/10000.
      dtauf = taufInit/10000.
      dalpha = alphaInit/10000.
      dpicp = picpInit/10000.

c-- Start loop on operating points
      do while (CaseFollows)

c-- Read input values
         read (7,*) fname
         write (6,'("Working on input for file ",a15,"....")') fname
         read (7,*) Tw
         read (7,*) Tc
         read (7,*) T0
         read (7,*) utarget_a0
         read (7,*) St
         read (7,*) r
         read (7,*) delPf_Ptc
         read (7,*) etapoly
         read (7,*) P0
         read (7,*) M0
         read (7,*) Tt4
         read (7,*) TurbineDump
         read (7,*) FanDump
         read (7,*) FilmCooled
         read (7,*) ChokedNoz
         read (7,*) Uncooled
         read (7,'(/l5)') CaseFollows
         if (CaseFollows) then
            write (6,'("case to follow...")')
         else
            write (6,'("last case!!")')
         end if

         chlast = ch
         if (TurbineDump) then
            ch = 'td'
         else if (FanDump) then
            ch = 'fd'
         else if (Uncooled) then
            ch = 'uc'
         else if (FilmCooled) then
            ch = 'fc'
         end if


c-- Calculate some values directly from the inputs
         N = 5
         tht = Tt4/T0
         th0 = 1. + (gac-1.)/2.*M0*M0
         tauc = sqrt(tht)/th0
         pic = tauc**((gac*etapoly)/(gac-1.))
         Tt2 = Th0*T0
         Pt0 = Th0**(gac/(gac-1.))*P0
         Pt4 = Pt0*pid*pic*pib
         Tcinp = Tc
         if (Tc.gt.10.) then
            if (Tc .gt. Tt2*tauc) then
               write (6,'("Tc as given = ",g15.8," is too large")') Tc
               write (6,'("Will use value of Tt3 = ",g15.8)') Tt2*tauc
               Tc = Tt2*tauc
               taucp = tauc
            else
               taucp = Tc/Tt2
            end if
            picp = taucp**((gac*etapoly)/(gac-1.))
            Ptc = Pt0*pid*picp
            picp = Ptc/Pt0/pid
            taucp = picp**((gac-1.)/gac/etapoly)
            Tc = Tt2*taucp
         else
            if (Uncooled) then
               picp = 0.0
               Ptc = 0.0
               taucp = 0.0
               Tc = 0.0
            else
               FiveEQFlag = .true.
            end if
         end if


c-- Take the initial guesses only the first time around
            if (ch.ne.chlast) then
               taut = tautInit
               if (Uncooled) then
c                   f = (tht - th0*tauc)/(h*etab/cpav/T0 - tht)
                  f = (cpt*T0*tht - cpc*T0*th0*tauc) /
     *                      (h*etab - cpt*T0*tht)
               else
                  f = fInit
               end if
               tauf = taufInit
               alpha = alphaInit
               if (FiveEQFlag) then
                  if (FilmCooled) then
                     picp = pic * pib / Pmargin *
     *                    (1.+(gat-1.)/2.*.5*.5)**
     *                              (-gat/(gat-1.))/(1.-delpf_Ptc)
                  else
                  picp = pic
               end if
            end if
         end if
c         if (Uncooled) f=(tht - th0*tauc)/(h*etab/cpav/T0 - tht)
         if (Uncooled) f = (cpt*T0*tht - cpc*T0*th0*tauc) /
     *                    (h*etab - cpt*T0*tht)

c-- Calculate conditions at each operating point
         call SolveConditions

c-- Write conditions to output file
         if (TurbineDump) then
            dum = Ptcp
         else if (FilmCooled) then
            dum = Ptc*(1.-delPf_Ptc)
         else if (FanDump) then
            dum = Ptcp*(1.-delPf_Ptc)
            Patinj = P7
         else
            dum = 0.0
         end if
         write (9,'(1(a20,","), 12(g20.10,","), 5(120,","),
     *          15(g20.10,","), 1(i20,","), 22(g20.10,","))')
     *        fname, Tw, Tc, Tcinp, T0, utarget_a0, St, r,
     *        delPf_Ptc, etapoly, P0, M0, Tt4, TurbineDump,
     *        FanDump, FilmCooled, ChokedNoz, Uncooled, tht, th0,
     *        tauc, pic,
     *        Tt2, Pt0, Pt4, taucp, picp, Ptc,
     *        taut, f, eps, W, pit, N, taus, tauf, alpha, pif,
     *        picf, Patinj, dum, M8, taucf, T8_T0, u8_a0, P8_P0, F8_ma0,
     *        M6, T6_T0, u6_a0, P6_P0, F6_ma0, Ftot_mtota, Isp,
     *        etat,etaad


c-- End loop on operating points
         firstloop = .false.
      end do
      close (7)
      close (9)


c-- Finish up program
      end



c#########################################################################
c#########################################################################
c#########################################################################
```

```
      subroutine SolveConditions

      implicit none

      include 'C7.inc'

      double precision xlastvec(5)
      integer Nlo, Nhi, step, k, Nlast
      logical loopflag, doneloop,error, approached
      double precision taustarget, tauslast
      parameter (taustarget=0.89)


      Nlo = -1
      Nhi = -1
      step = 1
      loopflag = .true.
      doneloop = .false.
      approached = .false.


      do while (loopflag)
         tauslast = taus
         do k=1,5
            xlastvec(k) = xvec(k)
         end do
         write (6,'("N:",i2," ######## Calling SolveGivenN ########")')
     *         N
         call SolveGivenN
         if ((taus-taustarget).lt.0.0) then
            Nlo = N
         else
            Nhi = N
         end if
c     --- if this is at least the second time through the loop
         if (doneloop) then
c     ---    if all taus so far under target or all over target
            if ((Nlo.lt.0.0 .or. Nhi.lt.0.0).and..not.NegWorkFlag) then
c     ---       if we're getting farther away
               if (abs(taus-taustarget).gt.abs(tauslast-taustarget)
     *                                                                then
                  N = Nlast
                  do k=1,5
                     xvec(k) = xlastvec(k)
                  end do
                  taus = tauslast
                  if (approached) then
                     call EvalModel(error)
                     loopflag = .false.
                  else
                     step=-1
                     if ((taus-taustarget).lt.0.0) then
                        Nlo = Nlast
                     else
                        Nhi = Nlast
                     end if
                  end if
               else
                  approached = .true.
               end if
            else
               loopflag = .false.
c    ---       if (we're getting farther away AND last taus ok) OR
c    ---       (this taus not ok)
               if (((abs(taus-taustarget).gt.abs(tauslast-taustarget))
     *              .and. (tauslast.gt.0.87))
     *              .or. taus.lt.0.87 .or. NegWorkFlag) then
                  N = Nlast
                  do k=1,5
                     xvec(k) = xlastvec(k)
                  end do
                  call EvalModel(error)
               end if
            end if
         end if
         if (loopflag) then
            Nlast = N
            if ((taus-taustarget).lt.0.0) then
               N = N + step
            else
               N = N - step
            end if
            if (N.lt.1) then
               write (6,'(/"*** ERROR: N<1 ***")')
               stop
            end if
         end if
         doneloop = .true.
      end do

      call Ftotcall
      return
      end


c#######################################################################
```

```
c#########################################################################
c#########################################################################


      subroutine SolveGivenN

      implicit none

      include 'C7.inc'

      integer nn, k, kk, i, j, nm1, loopcount
      double precision f0(5), Jac(5,5), s(5), xmax(5), xmin(5),
     *     dxvect(5), dxmin(5),
     *     dxmax(5), f1(5), fn1(5)
      double precision Tol
      logical loopflag, loop2flag, error, dontprint
      double precision factor


      dontprint = .false.
      factor = 1.0

c-- Set the overall tolerance
      Tol = 1.e-5

c-- Set the limits on each of the variables
      xmax(1)=1.0
      xmax(2)=2.0
      xmax(3)=30.0
      xmax(4)=0.2
      xmax(5)=50.0
      xmin(1)=0.1
      xmin(2)=1.0
      xmin(3)=0.01
      xmin(4)=0.001
      xmin(5)=1.0

c-- Set the limits on the dx vector
      dxmin(1) = 5.e-7
      dxmin(2) = 1.2e-6
      dxmin(3) = 5.e-6
      dxmin(4) = 3.e-8
      dxmin(5) = 1.e-5
      dxmax(1) = .00005
      dxmax(2) = .00012
      dxmax(3) = .0001
      dxmax(4) = .000003
      dxmax(5) = .001
```

```
c-- Set the number of Equations to solve
      if (Uncooled) then
         nn = 3
      else if (FiveEQFlag) then
         nn = 4
      end if
      nm1 = nn-1


c-- Set the dxvect that will actually be used
      do kk=1,nn
         k=kk
         if (k.eq.4) k=5
         dxvect(k)=dxvec(k)
      end do


c-- Evaluate the model at the initial guess
 5    error = .true.
      do while (error)
         error = .false.
         call EvalModel(error)
         if (error) then
               picp = picp + 0.1*(pic-picp)
         end if
      end do

c-- Print results to screen (get rid of this later)
      if (dontprint) then
         write (6,'("::::::: adjusting picp - ",g16.5)') picp
      else
         write (6,'(5(g16.5))') xvec(1),xvec(2),xvec(3),xvec(5)
         write (6,'(4x5(g16.5))')fvec(1),fvec(2),fvec(3),fvec(4)
      end if

c-- Begin the iteration
c----------------------
      loopcount = 0
      loopflag = .true.
      do while (loopflag)
         loopcount = loopcount + 1

c-- Evaluate the Jacobian
         do k=1,nn
            f0(k) = fvec(k)
         end do
         if (loopcount.gt.2) then
            do kk=1,nn
               k=kk
               if (k.eq.4) k=5
            dxvect(k) = s(k)/1000.
            if (dxvect(k).gt.dxmax(k)) dxvect(k) = dxmax(k)
            if (dxvect(k).lt.dxmin(k)) dxvect(k) = dxmin(k)
         end do
      end if
      do kk=1,nn
         k=kk
         if (k.eq.4) k=5

         xvec(k) = xvec(k) + dxvect(k)
         error = .false.
         call EvalModel(error)
         if (error) then
            picp = picp + 0.1*(pic-picp)
            write (6,'("uh-oh !!")')
            xvec(k) = xvec(k) - dxvect(k)
            dontprint = .true.
            goto 5
         end if
         do j=1,nn
            f1(j) = fvec(j)
         end do

         xvec(k) = xvec(k) - 2.0*dxvect(k)
         error = .false.
         call EvalModel(error)
         if (error) then
            picp = picp + 0.1*(pic-picp)
            write (6,'("uh-oh !!")')
            xvec(k) = xvec(k) + dxvect(k)
            dontprint = .true.
            goto 5
         end if
         do j=1,nn
            fn1(j) = fvec(j)
         end do

         do j=1,nn
            Jac(j,kk) = (f1(j)-fn1(j))/2./dxvect(k)
         end do
         xvec(k) = xvec(k) + dxvect(k)
      end do
c-- Put the minus here because we need it for solving the equation
c-- and because we won't be using this fvec anymore anyway
      do k=1,nn
         fvec(k) = -f0(k)
      end do
      dontprint = .false.
```

```
c-- Find the step to take
         call gauss(Jac,nn,fvec,s)
         s(5) = s(4)

c-- Check that the step keeps us within bounds
         loop2flag = .true.
         do while (loop2flag)
            loop2flag = .false.
            do kk=1,nn
               i=kk
               if (i.eq.4) i=5
               if ((xvec(i)+s(i)).gt.xmax(i) .or.
     *             (xvec(i)+s(i)).lt.xmin(i)     ) loop2flag = .true.
            end do
c--         This line keeps alpha from running wild
            if (abs(s(3)).gt.0.5) loop2flag = .true.
c--         This line keeps picp from running wild
            if (abs(s(5)).gt.1.0) loop2flag = .true.
            if (loop2flag) then
               do kk=1,nn
                  i=kk
                  if (i.eq.4) i=5
                  s(i) = 0.9*s(i)
               end do
            end if
         end do

c-- Find the next x
         do kk=1,nn
            i=kk
            if (i.eq.4) i=5
            xvec(i) = xvec(i) + s(i)
         end do

c-- Evaluate f
         error = .true.
         do while (error)
            error = .false.
            call EvalModel(error)
            if (error) then
               do kk=1,nn
                  i=kk
                  if (i.eq.4) i=5
                  xvec(i) = xvec(i) - s(i)
                  s(i) = 0.9*s(i)
                  xvec(i) = xvec(i) + s(i)
               end do
            end if
         end do
```

```
c-- Print results to screen (get rid of this later)
          write (6,'(5(g16.5))') xvec(1),xvec(2),xvec(3),xvec(5)
          write (6,'(4x5(g16.5))')fvec(1),fvec(2),fvec(3),fvec(4)

c-- Check Tolerance
          loopflag = .false.
          do i=1,nn
             if (abs(fvec(i)).gt.Tol) loopflag = .true.
          end do

          if (loopcount.gt.34 .and. NegWorkFlag) loopflag = .false.

       end do


       return
       end



c#########################################################################
c#########################################################################
c#########################################################################



       subroutine gauss(A,n,b,c)

       double precision A(5,5), b(5), c(5)
       integer n, k(5)
       double precision dum

c-- Set up the index vector k
       do i=1,n
          k(i) = i
       end do

c-- Triangularize the system
       do i=1,n
c--       Find the pivot
          ihold = k(i)
          itemp = i
          do ii=i+1,n
             if (abs(A(k(ii),i)).gt.abs(A(k(i),i))) then
                k(i) = k(ii)
                itemp = ii
             end if
          end do
          k(itemp) = ihold
c--       Eliminate below pivot
```

```
         do ii=i+1,n
            dum = A(k(ii),i)/A(k(i),i)
            do j=i,n
               A(k(ii),j) = A(k(ii),j) - dum*A(k(i),j)
            end do
            b(k(ii)) = b(k(ii)) - dum*b(k(i))
         end do
      end do

c-- Solve by backsubstitution
      do i=n,1,-1
         do j=i+1,n
            b(k(i)) = b(k(i)) - A(k(i),j)*b(k(j))
         end do
         b(k(i)) = b(k(i))/A(k(i),i)
      end do

c-- put the solution back in order
      do i=1,n
         c(i) = b(k(i))
      end do

      return
      end


c#########################################################################
c#########################################################################
c#########################################################################


      subroutine EvalModel(error)

      implicit none

      include 'C7.inc'

      double precision Efunction, Wfunction
      external Efunction, Wfunction
      logical error



      pif = tauf**((gac*etapoly)/(gac-1.))
      if (FiveEQFlag) then
         Ptc = Pt0*pid*picp
         taucp = picp**((gac-1.)/gac/etapoly)
         Tc = Tt2*taucp
         Ptcp = Ptc*(1.-delPf_Ptc)*(Tw/Tc)**(-gac*0.1*0.1/2.)
      end if
      call  Fcall(error)
      if (error) return
c      f = (1.-eps)*(tht-th0*tauc)/(h*etab/cpav/T0 - tht)
      f = (1.-eps)*(cpt*T0*tht - cpc*T0*th0*tauc) /
     *    (h*etab - cpt*T0*tht)
      call Ucall

c-- Balance the turbine work with the compressor and fan work
      fvec(1) = (W-Wfunction())/1.e6

c-- Match the target output velocities
      fvec(2) = u6_a0/utarget_a0 - 1.0
      fvec(3) = u8_a0/utarget_a0 - 1.0

c-- Match the turbine eps with the burner energy balance eps
c      if (.not.Uncooled) fvec(4) = (eps-Efunction())/0.2

c-- Match the pressure at the coolant injection with the coolant press.
      if (FiveEQFlag .and. TurbineDump) then
         fvec(4) = (Patinj-(Ptcp*Pmargin))/1.e5
      else if (FiveEQFlag .and. FilmCooled) then
         fvec(4) = (Patinj-(Pt0*pid*picp*(1.-delPf_Ptc)*Pmargin))/1.e5
      else if (FiveEQFlag .and. FanDump) then
         fvec(4) = (p7-((Ptcp*(1.-delpf_Ptc))*Pmargin))/1.e5
      end if

      return
      end


c#########################################################################
c#########################################################################
c#########################################################################


      subroutine Ftotcall

      implicit none

      include 'C7.inc'



c-- Calculate P8/P0
      P8_P0 = Pt0/P0*pid*pif*picf*(1.+(gac-1.)/2.*M8*M8)**
     *                                              (-gac/(gac-1.))

c-- Calculate F8/ma0
      if (TurbineDump .or. FilmCooled .or. Uncooled) then
```

```
         F8_ma0 = alpha*u8_a0 - alpha*M0 +
     *            alpha/gac*T8_T0/u8_a0*(1.-1./P8_P0)
      else
         F8_ma0 = (alpha + eps)*u8_a0 - alpha*M0 +
     *            (alpha+eps)/gac*T8_T0/u8_a0*(1.-1./P8_P0)
      end if


c-- Calculate P6/P0
      P6_P0 = Pt0/P0*pid*pic*pib*pit*(1.+(gat-1.)/2.*M6*M6)**
     *                                          (-gat/(gat-1.))

c-- Calculate F6/ma0
      if (TurbineDump .or. FilmCooled .or. Uncooled) then
         F6_ma0 = (1.+f)*u6_a0 - M0 +
     *            (1.+f)/gat*T6_T0/u6_a0*(1.-1./P6_P0)
      else
         F6_ma0 = (1.+f-eps)*u6_a0 - M0 +
     *            (1.+f-eps)/gat*T6_T0/u6_a0*(1.-1./P6_P0)
      end if


c-- Calculate Ftot/mtota
      Ftot_mtota = (F6_ma0 + F8_ma0)/(1.+alpha)


c-- Calculate Isp
      Isp = sqrt(gac*Rgasc*T0)*(1.+alpha)*Ftot_mtota/g/f


c-- Calculate efficiency
      if (Uncooled) then
         etat = (1.-taut)/(1.-pit**((gat-1.)/gat))
      else if (TurbineDump .or. FilmCooled) then
         etat = ((1.-eps+f)*Tt4+eps*Tc-(1.+f)*Tt4*taut)/
     *        ((1.-eps+f)*Tt4*(1.-pit**((gat-1.)/gat)) +
     *         eps*Tc*(1.-(Pt4*pit/Ptc)**((gat-1.)/gat)))
      else
         etat = ((1.-eps+f)*Tt4+eps*Tc+alpha*Tt2*tauf-
     *           (alpha+eps)*Tt2*tauf*taucf-(1.-eps+f)*Tt4*taut)/
     *        ((1.-eps+f)*Tt4*(1.-pit**((gat-1.)/gat))+
     *         eps*Tc*(1.-(Pt0*pid*pif*picf/Ptc)**((gac-1.)/gac))+
     *         alpha*Tt2*tauf*(1.-picf**((gac-1.)/gac)))
      end if


c-- Return
      return
      end
```

```
c##########################################################################
c##########################################################################
c##########################################################################


      subroutine Ucall

      implicit none

      include 'C7.inc'

      double precision M7, Mcf, T7, Tcf, u7, ucf, u7p, T7p, Ptcf,Ttcf,
     *     Pt7, Tt7
      double precision dum
      parameter (M7=0.3)



c-- Calculate picf and taucf
      if (FanDump) then
         Ttcf = Tw
         Ptcf = Ptcp*(1.0-delPf_Ptc)
         Tt7 = Tt2*tauf
         Pt7 = pt0*pid*pif
         P7 = Pt7/(1.0+(gac-1.0)/2.0*M7*M7)**(gac/(gac-1.0))
         T7 = Tt7/(1.0+(gac-1.0)/2.0*M7*M7)
         dum = (Ptcf/Pt7)**((gac-1.0)/gac)
         dum = dum*(1.0+(gac-1.0)/2.0*M7*M7)
c from here is added a test (6 lines) (seems to work, though)
         dum = 2.0/(gac-1.0)*(dum-1.0)
         if (dum.lt.0.0) then
            Mcf = -sqrt(-dum)
         else
            Mcf = sqrt(dum)
         end if
         Tcf = Ttcf/(1.0+(gac-1.0)/2.0*Mcf*Mcf)
         u7 = M7*sqrt(gac*Rgasc*T7)
         if (.not.(u7.lt.1. .or. u7.gt.0.)) then
            write (6,'("NaN error #U2#")')
            stop
         end if
         ucf = Mcf*sqrt(gac*Rgasc*Tcf)
         if (.not.(ucf.lt.1. .or. ucf.gt.0.)) then
            write (6,'("NaN error #U3#")')
            stop
         end if
         u7p = (alpha*u7 + eps*ucf)/(alpha + eps)
         dum = alpha*cpc*T7 + eps*cpc*Tcf + alpha*u7*u7/2.0
```

```
            dum = dum + eps*ucf*ucf/2.0 - (alpha+eps)*u7p*u7p/2.0
            T7p = dum/(alpha+eps)/cpc
            picf = P7/Pt7*(1.0+(gac-1.0)/2.0*u7p*u7p/gac/Rgasc/T7p)**
     *                                                  (gac/(gac-1.0))
            taucf = (alpha + Ttcf*eps/T0/th0/tauf)/(alpha + eps)
         else
            picf = 1.
            taucf = 1.
         end if



c-- Find u8/a0
c-
c-- Calculate M8
         dum = 2./(gac-1.)*((Pt0/P0*pid*pif*picf)**((gac-1.)/gac)-1.)
         if (dum.lt.0.0) then
            M8 = -sqrt(-dum)
         else
            M8 = sqrt(dum)
         end if
         if (ChokedNoz .and. M8.gt.1.) M8=1.
c-
c-
c-- Calculate T8/T0
         T8_T0 = th0*tauf*taucf/(1.+(gac-1.)/2.*M8*M8)
c-
c-
c-- Calculate u8/a0
         u8_a0 = M8*sqrt(T8_T0)




c-- Find u6/a0
c-
c-- Calculate M6
         dum = 2./(gat-1.)*((Pt0/P0*pid*pib*pic*pit)**((gat-1.)/gat)-1.)
         if (dum.lt.0.0) then
            M6 = -sqrt(-dum)
         else
            M6 = sqrt(dum)
         end if
         if (ChokedNoz .and. M6.gt.1.) M6=1.
c-
c-
c-- Calculate T6/T0
         T6_T0 = tht*taut/(1.+(gat-1.)/.2*M6*M6)
c-
c-
```

```
c-- Calculate u6/a0
      u6_a0 = M6*sqrt(gat/gac*Rgast/Rgasc*T6_T0)



      return
      end




c#########################################################################
c#########################################################################
c#########################################################################



      function Efunction()

      implicit none

      include 'C7.inc'

      double precision Efunction



c-- Comes from the fuel-air ratio equation

      if (Uncooled) then
         Efunction = 0.0
      else
c         Efunction = 1.0 - f*(h*etab/cpav/T0 - tht)/(tht-th0*tauc)
         Efunction = 1.0 - f*(h*etab - cpt*T0*tht) /
     *                     (cpt*T0*tht - cpc*T0*th0*tauc)
      end if


      return
      end



c#########################################################################
c#########################################################################
c#########################################################################
```

```
      function Wfunction()

      implicit none

      include 'C7.inc'

      double precision Wfunction

      double precision Efunction
      external Efunction



c-- Comes from the work balance

      if (Uncooled) then
         Wfunction = cpc*Tt2*(tauc - 1. + alpha*(tauf - 1.))
      else
         Wfunction = cpc*Tt2*(taucp-1.+(1.-Efunction())*(tauc-taucp) +
     *                         alpha*(tauf-1.))
      end if

      return
      end



c#########################################################################
c#########################################################################
c#########################################################################



      subroutine Fcall(error)

      implicit none

      include 'C7.inc'

      logical notanum
      external notanum

      logical error
      double precision RR
      double precision summc_m1, sumWi_mdot
      double precision alphaT, Tta(25), Ttb(25), msiplus1_msi(0:25),
     *     ms_m1(0:25),
     *     mcs_ms(25), mr_ms(25), Pta(25), Ptb(25),
     *     mr_m1(25), mcr_mr(25),W_m1(25), dsw_cpr, dsc_cps, dsc_cpr,
     *     Wi_summcj(25),Wi_mdot(25)
      double precision Ttsp, Ptsp, Tsp, Psp, usp, Msp,
     *     Ttfp, Ptfp, Tfp, Pfp, ufp, Mfp,
     *     Ttf,  Ptf,  Tf,  Pf,  uf,  Mf,
     *     Tts,  Pts,  Ts,  Ps,  us,  Ms,
     *     Ttc,             Pc,  uc,  Mc,
     *     Ttcp,       Tcp, Pcp, ucp, Mcp,
     *     Ttr,  Ptr,  Tr,  Pr,  ur,  Mr,
     *     Ttrp, Ptrp, Trp, Prp, urp, Mrp
      double precision a, b, c,
     *     dadM, dbdM, dcdM, db_cdM, dedM, dTdM, dPdM, dFuncdM, Func
      double precision mcint, mcfilm, m, x_t, nholerows
      double precision Q_m, ex, uT, MT, Tcmoving, Ttfprel, Mfpstep
      integer j

      integer i
      double precision dum,dum2,dum3
      parameter (x_t=10.0, nholerows=2.0)
      parameter (Msp=1.0, Mrp=0.5)
      parameter (MT=0.5)

      Ms = 0.5
      Ttc = Tc
      Ttf = Ttc
      Ptf = Ptc*(1. - delPf_Ptc)
      NegWorkFlag = .false.

c-- Calculate taus
      taus = taut**(1./real(N))


c-- Open loop on stages (4-9)
      Tta(1) = Tt4
      Pta(1) = Pt4
      msiplus1_msi(0) = 1.
      ms_m1(0) = 1.
      summc_m1 = 0.
      do i=1,N

c--     Do stator calculations
         ms_m1(i) = msiplus1_msi(i-1)*ms_m1(i-1)
         if (Uncooled) then
            mcs_ms(i) = 0.
         else
            if (Tta(i).gt.Tw) then
               if (.not.FilmCooled) then
                  mcs_ms(i) = St*r*cpt/cpc*(Tta(i)-Tw)/(Tw-Tc)
               else
```

```
                  etaad = (Tta(i)-Tw)/(Tta(i)-Tc)
                  if (etaad.gt.0.6) then
                     etaad = 0.6
                     mcint = St*r*cpt/cpc*
     *                       (Tta(i)-Tw-etaad*(Tta(i)-Tc)) /
     *                       (Tw-Tc)
                  else
                     mcint = 0.0
                  end if
                  if (etaad.lt.0.37821) then
                     m = 0.6976*etaad
                  else
                     m = 25.0646*etaad**3 - 18.9596*etaad**2 +
     *                     4.2830*etaad    +  0.0002
                  end if
                  mcfilm = m*r*nholerows/x_t
                  mcs_ms(i) = mcint + mcfilm
               end if
            else
               mcs_ms(i) = 0.0
            end if
         end if
         summc_m1 = summc_m1 + mcs_ms(i)*ms_m1(i)

c--     Do stator -> rotor calculations
         if (TurbineDump .or. FilmCooled) then
            mr_ms(i) = 1. + mcs_ms(i)
            Ttb(i) = Tta(i)*(1.+mcs_ms(i)*cpc/cpt*Tc/Tta(i)) /
     *                       (1.+mcs_ms(i))
         else
            mr_ms(i) = 1.
            if (Uncooled) then
               Ttb(i) = Tta(i)
            else
               if (Tta(i).gt.Tw) then
                  Ttb(i) = Tta(i)*(1.-St*r*(1.-Tw/Tta(i)))
               else
                  Ttb(i) = Tta(i)
               end if
            end if
         end if

c--     Do stator Pt calculations
         if (Uncooled .or. Tta(i).le.Tw) then
            Ptb(i) = Pta(i)
            if (i.eq.1 .and. .not.Uncooled) Patinj=Pta(i)/1.8324
         else
            if (FilmCooled) then
               Ttfp = Tta(i) - etaad*(Tta(i)-Ttc)
               if (mcint.le.1.e-14) then
               Ttsp = Tta(i) - cpc/cpt*mcfilm*(Ttfp-Ttf)
            else
               Ttsp = Tta(i) - cpc/cpt*(mcfilm*(Ttfp-Ttf) +
     *                                 mcint*(Tw-Ttc))
            end if
         else
            Ttsp = Tta(i) - cpc/cpt*mcs_ms(i)*(Tw-Ttc)
         end if
         dum = 1.0
         do j=1,30
            dum2 = (Tta(i) + real(j)/30.0*(Ttsp-Tta(i)))  /
     *             (Tta(i) + real(j-1)/30.0*(Ttsp-Tta(i)))
            dum2 = dum2**(-gat/2.0 *
     *               ((Ms + real(j)/30.0*(Msp-Ms)) +
     *                (Ms + real(j-1)/30.0*(Msp-Ms)))**2/4.0)
            dum = dum*dum2
         end do
         Ptsp = Pta(i)*dum
         Psp = Ptsp*(1.+(gat-1.)/2.*Msp**2)**(-gat/(gat-1.))

         if (FanDump) then
            Ptb(i) = Ptsp
         else
            if (i.eq.1) then
               if (TurbineDump) then
                  Patinj = Psp
               else
                  Patinj = Pta(i) /
     *                     (1.+(gat-1.)/2.*Ms*Ms)**(gat/(gat-1.))
               end if
            end if
            usp = Msp*sqrt(gat*Rgast*Ttsp/(1.+(gat-1.)/2.*Msp**2))
            if (notanum(usp)) then
               write (6,'("usp=NaN, enter dummy number to cont.")')
               read (5,*) dum
            end if
            if (FilmCooled) then
               Ps = Pta(i)*(1.+(gat-1.)/2.*Ms**2)**(-gat/(gat-1.))
               if (Ps.gt.(Patinj+Patinj*1.e-10)) then
                  write (6,'("@@@@@@@@  Patinj too small,",
     *                 " changing it at i=",i3)') i
                  Patinj = Ps
               end if
               if (Ptf.lt.Ps) then
                  write (6,'("ptf.lt.ps, paused")')
ccw                  read (5,*) dum
                  error = .true.
                  return
               end if
               Ptfp = Ptf*(Ttfp/Ttf)**(-gac*.25*.25/2.)
```

```
                  dum = 2./(gac-1.)*( (Ptfp/Psp)**((gac-1.)/gac) - 1.)
                  if (dum.lt.0.0) then
                     write (6,'("Mfp negative ***")')
                     Mfp = -sqrt(-dum)
                  else
                     Mfp = sqrt(dum)
                  end if
                  ufp = Mfp*sqrt(gac*Rgasc*Ttfp/(1.+(gac-1.)/2.*Mfp**2))
                  if (notanum(ufp)) then
                     write (6,'("ufp=NaN, enter dummy num to cont.")')
                     read (5,*) dum
                  end if

                  if (mcint.le.1.e-14) then
                     ur = (usp + mcfilm*ufp)/(1.+ mcfilm)
                  else
                     Tcp = Tw*(Psp/Ptcp)**((gac-1.)/gac)
                     if (Tw.lt.Tcp) then
                        write (6,'("tw.lt.tcp, paused")')
ccw                        read (5,*) dum
                        error = .true.
                        return
                     end if
                     ucp = sqrt(2.*cpc*(Tw-Tcp))
                     if (notanum(ucp)) then
                        write (6,'("ucp=NaN, enter dum num to cont.")')
                        read (5,*) dum
                     end if
                     ur = (usp + mcfilm*ufp + mcint*ucp) /
     *                    (1.+mcfilm+mcint)
                  end if
               else
                  if (Psp.gt.(Patinj+Patinj*1.e-10)) then
c                     write (6,'("@@@@@@@  Patinj too small, changing",
c    *                      " it at i=",i3)') i
                     Patinj = Psp
                  end if
                  Tcp = Tw*(Ptsp/(1.+(gat-1.)/2.*Msp**2)**(gat/(gat-1.))
     *                 /Ptcp)**((gac-1.)/gac)
                  if (Tw.lt.Tcp) then
                     write (6,'("tw.lt.tcp, paused")')
ccw                     read (5,*) dum
                  end if
                  if (Tw.lt.Tcp) then
                     ucp = -sqrt(-2.*cpc*(Tw-Tcp))
                  else
                     ucp = sqrt(2.*cpc*(Tw-Tcp))
                  end if
                  if (notanum(ucp)) then
                     write (6,'("ucp=NaN, enter dummy num to cont.")')
                  read (5,*) dum
               end if
               ur = (usp + mcs_ms(i)*ucp)/(1. + mcs_ms(i))
            end if
            Tr = Ttb(i) - ur**2/2./cpt
            Mr = sqrt(ur**2/gat/Rgast/Tr)
            if (notanum(Mr)) then
               write (6,'("Mr=NaN, enter dummy num to cont.")')
               read (5,*) dum
            end if
            Ptb(i) = Ptsp*((1.+(gat-1.)/2.*Mr**2)/
     *              (1.+(gat-1.)/2.*Msp**2))**(gat/(gat-1.))
         end if
      end if



c--   Calculate RR, alphaT
c     (the next two lines assume Mt=0.5, gat=1.3)
      RR = 1. - (1.-taus)*(1.+0.15*Mr*Mr)/0.15
      alphaT = 1. + (4.*RR-3.)*0.0375/(1.+0.15*Mr*Mr)

c--   Do rotor calculations
      uT = MT*sqrt(gat*Rgast*Tr)
      Tcmoving = Tc
      mr_m1(i) = mr_ms(i)*ms_m1(i)
      if (Uncooled .or. (alphaT*Ttb(i)).le.Tw) then
         mcr_mr(i) = 0.0
      else
         if (.not.FilmCooled) then
            mcr_mr(i) = St*r*cpt/cpc*(alphaT*Ttb(i)-Tw)/(Tw-Tc)
         else
            etaad = (alphaT*Ttb(i)-Tw)/(alphaT*Ttb(i)-Tcmoving)
            if (etaad.gt.0.6) then
               etaad = 0.6
               mcint = St*r*cpt/cpc*
     *            (alphaT*Ttb(i)-Tw-etaad*(alphaT*Ttb(i)-Tcmoving)) /
     *            (Tw-Tc)
            else
               mcint = 0.0
            end if
            if (etaad.lt.0.37821) then
               m = 0.6976*etaad
            else
               m = 25.0646*etaad**3 - 18.9596*etaad**2 +
     *              4.2830*etaad    +  0.0002
            end if
            mcfilm = m*r*nholerows/x_t
            mcr_mr(i) = mcint + mcfilm
            end if
         end if
         summc_m1 = summc_m1 + mcr_mr(i)*mr_m1(i)
         if ( (TurbineDump .or. FilmCooled) .and.
     *         (alphaT*Ttb(i)).gt.Tw) then
            W_m1(i) = mr_m1(i)*(cpt*(Ttb(i)-(1.+mcr_mr(i))*taus*Tta(i))+
     *                                cpc*mcr_mr(i)*Tc)
         else if (FanDump .and. (alphaT*Ttb(i)).gt.Tw) then
            W_m1(i) = mr_m1(i)*cpt*(Ttb(i) - taus*Tta(i) -
     *                                St*r*(Ttb(i)*alphaT-Tw))
         else
            W_m1(i) = mr_m1(i)*cpt*(Ttb(i) - taus*Tta(i))
         end if
         if (W_m1(i).lt.0.0) then
            write (6,'("_______ neg work !!!!  i=",i2)') i
            NegWorkFlag = .true.
         end if

c--     Do rotor -> stator calculations
         if (TurbineDump.or.FilmCooled) then
            msiplus1_msi(i) = 1. + mcs_ms(i) + mcr_mr(i)*mr_ms(i)
         else
            msiplus1_msi(i) = 1.
         end if
         Tta(i+1) = taus*Tta(i)


c--     Do rotor Pt calculations
         if (Uncooled .or. (alphaT*Ttb(i)).le.Tw) then
            Pta(i+1)= Ptb(i)*(Tta(i+1)/Ttb(i))**(gat/((gat-1.)*etapoly))
         else
            if (FilmCooled) then
               Ttfprel = alphaT*Ttb(i) - etaad*(alphaT*Ttb(i)-Tcmoving)
               Ttfp = Ttfprel + uT**2/2./cpc
               if (mcint.le.1.e-14) then
                  Ttrp = Ttb(i) - W_m1(i)/mr_m1(i)/cpt -
     *                     cpc/cpt*mcfilm*(Ttfp-Ttf)
               else
                  Ttrp = Ttb(i) - W_m1(i)/mr_m1(i)/cpt -
     *                     cpc/cpt*(mcfilm*(Ttfp-Ttf) + mcint*(Tw-Ttc))
               end if
            else
               Ttrp = Ttb(i) - W_m1(i)/mr_m1(i)/cpt -
     *                  cpc/cpt*mcr_mr(i)*(Tw-Ttc)
            end if
            if (FilmCooled) then
               Q_m = mcint*cpc*(Tw-Ttc) + mcfilm*cpc*(Ttfp-Ttf)
            else
               Q_m = St*r*cpt*(alphaT*Ttb(i)-Tw)
            end if
            dum = 1.0
            do j=1,30
c             - M
               dum3 = Mr + (real(j)-.5)/30.0*(Mrp-Mr)
c             - W_m
               dum2 = W_m1(i)/mr_m1(i)
c             - [...]
               dum2 = 1. +
     *              (1.-etapoly)/etapoly*dum2/cpt/(Ttb(i)-Ttrp) -
     *              (1.+ (gat-1.)/2.*dum3**2)*Q_m/cpt/(Ttb(i)-Ttrp)
               dum2 = dum2*gat/(gat-1.)
c             - Tt/Tt
               dum3 = (Ttb(i) + real(j)/30.0*(Ttrp-Ttb(i)))  /
     *                (Ttb(i) + real(j-1)/30.0*(Ttrp-Ttb(i)))
               dum2 = dum3**dum2
               dum = dum*dum2
            end do
            Ptrp = Ptb(i)*dum
            Prp = Ptrp*(1.+(gat-1.)/2.*Mrp**2)**(-gat/(gat-1.))

            if (FanDump) then
               Pta(i+1) = Ptrp
            else
               urp = Mrp*sqrt(gat*Rgast*Ttrp/(1.+(gat-1.)/2.*Mrp**2))
               if (notanum(urp)) then
                  write (6,'("urp=NaN, enter dummy num to cont.")')
                  read (5,*) dum
               end if
               if (FilmCooled) then
                  Pr = Ptb(i)*(1.+(gat-1.)/2.*Mr**2)**(-gat/(gat-1.))
                  if (Pr.gt.(Patinj + Patinj*1.e-10)) then
                     write (6,'("@@@@@@@  Patinj too small, changing",
     *                  " it at i=",i3)') i
                     Patinj = Pr
                  end if
                  if (Ptf.lt.Pr) then
                     write (6,'("ptf.lt.pr, paused")')
ccw                     read (5,*) dum
                     error = .true.
                     return
                  end if
                  Ptfp = Ptf*(Ttfp/Ttf)**(-gac*0.25*0.25/2.)
                  dum = 2./(gac-1.)*( (Ptfp/Prp)**((gac-1.)/gac) -1.)
                  if (dum.lt.0.0) then
                     write (6,'("IN ROTOR: Mfp is negative ***")')
                     Mfp = -sqrt(-dum)
                  else
                     Mfp = sqrt(dum)
                  end if
                  ufp = Mfp*sqrt(gac*Rgasc*Ttfp/(1.+(gac-1.)/2.*Mfp**2))
                  if (notanum(ufp)) then
                     write (6,'("ufp=NaN, enter dummy num to cont.")')
                     read (5,*) dum
                  end if

                  if (mcint.le.1.e-14) then
                     us = (urp + mcfilm*ufp)/(1.+ mcfilm)
                  else
                     Tcp = Tw*(Prp/Ptcp)**((gac-1.)/gac)
                     if (Tw.lt.Tcp) then
                        write (6,'("tw.lt.tcp, paused")')
ccw                        read (5,*) dum3
                        error = .true.
                        return
                     end if
                     ucp = sqrt(2.*cpc*(Tw-Tcp))
                     if (notanum(ucp)) then
                        write (6,'("ucp=NaN, enter dum num to cont.")')
                        read (5,*) dum
                     end if
                     us = (urp + mcfilm*ufp + mcint*ucp) /
     *                     (1.+mcfilm+mcint)
                  end if
               else
                  if (Prp.gt.(Patinj+Patinj*1.e-10)) then
c                     write (6,'("@@@@@@  Patinj too small, changing",
c    *                   " it at i=",i3)') i
                     Patinj = Prp
                  end if
                  Tcp = Tw*(Ptrp/(1.+(gat-1.)/2.*Mrp**2)**(gat/(gat-1.))
     *                  /Ptcp)**((gac-1.)/gac)
                  if (Tw.lt.Tcp) then
                     write (6,'("tw.lt.tcp, paused")')
ccw                     read (5,*) dum3
                  end if
                  if (Tw.lt.Tcp) then
                     ucp = -sqrt(-2.*cpc*(Tw-Tcp))
                  else
                     ucp = sqrt(2.*cpc*(Tw-Tcp))
                  end if
                  if (notanum(ucp)) then
                     write (6,'("ucp=NaN, enter dummy num to cont.")')
                     read (5,*) dum
                  end if
                  us = (urp + mcr_mr(i)*ucp)/(1. + mcr_mr(i))
               end if
               Ts = Tta(i+1) - us**2/2./cpt
               Ms = sqrt(us**2/gat/Rgast/Ts)
               if (notanum(Ms)) then
                  write (6,'("Ms=NaN, enter dummy num to cont.")')
```

```
                read (5,*) dum
             end if
             Pta(i+1) = Ptrp*((1.+(gat-1.)/2.*Ms**2)/
     *            (1.+(gat-1.)/2.*Mrp**2))**(gat/(gat-1.))
          end if
       end if


c-- Close loop (4-9) on stages i
      end do
      pit = Pta(N+1)/Pt4

c-- Open loop (11-15) on stages i
      sumWi_mdot = 0.
      do i=1,N
         if (Uncooled .or. summc_m1.lt.1.e-15) then
c            Wi_mdot(i) = W_m1(i)*(1. + f)
            Wi_mdot(i) = W_m1(i)*(1. + .03)
         else
            Wi_summcj(i) = W_m1(i)/summc_m1
c            Wi_mdot(i) = (1.+f)/(1./W_m1(i) + 1./Wi_summcj(i))
            Wi_mdot(i) = (1.+.03)/(1./W_m1(i) + 1./Wi_summcj(i))
         end if
         sumWi_mdot = sumWi_mdot + Wi_mdot(i)
      end do
      W = sumWi_mdot

c-- Calculate eps
      if (Uncooled .or. summc_m1.lt.1.e-15) then
         eps = 0.0
      else
         eps = Wi_mdot(1)/Wi_summcj(1)
      end if




      return
      end



c#######################################################################
c#######################################################################
c#######################################################################
```

```
      function notanum(num)

      implicit none

      include 'C7.inc'

      logical notanum
      double precision num


      notanum = .true.
      if (num.gt.0.0 .or. num.lt.1.0) notanum = .false.

      return
      end
```

C.2 Include File: C7.inc

```
c-- Fixed Constants
      double precision h, etab, cpc, cpt, gac, gat, cpav, pid, Rgasc,
     *     Rgast, pib, g,
     *     betaE, betaW, Pmargin
      parameter (h=4.3e7, etab=.95, cpc=1000., cpt=1130., gac=1.4,
     *          gat=1.3, cpav=1065., pid=.95, Rgasc=285.7143,
     *          Rgast=260.7692, pib=.95,
     *          g=9.81, betaE=0.05, betaW=30000., Pmargin=0.95)

c-- Input Variables
      double precision Tw, Tc, T0, alpha, St, r, delPf_Ptc, etapoly,
     *     P0, M0,
     *     Tt4, utarget_a0
      logical TurbineDump, FanDump, ChokedNoz, Uncooled, FilmCooled

c-- Calculated Directly From Input Variables
      double precision tht, th0, tauc, Tt2, taucp, Ptc, Pt4, pic, Pt0,
     *     picp, Tcinp,
     *     pif, tauf
      logical FiveEQFlag

c-- For numerical solution
      double precision fvec(5), xvec(5), dxvec(5)
      double precision dtaut, df, dtauf, dalpha, dpicp
      double precision tautInit, fInit, taufInit, alphaInit, picpInit

c-- Other variables
      double precision taut, f, eps, W, pit, etat
      double precision taus
      integer N
      double precision picf, M8, taucf, T8_T0, u8_a0, F8_ma0, M6, T6_T0,
     *      u6_a0,
     *     F6_ma0, Ftot_mtota, Isp, P8_P0, P6_P0, P7, Ptcp, Patinj,
     *     etaad
      logical NegWorkFlag

c-- Equivalences
      equivalence (xvec(1),taut),(xvec(2),tauf),(xvec(3),alpha),
     *     (xvec(4),f),(xvec(5),picp)
      equivalence (dxvec(1),dtaut),(dxvec(2),dtauf),(dxvec(3),dalpha),
     *     (dxvec(4),df),(dxvec(5),dpicp)

      common /blk/ Tw, Tc, T0, St, r, delPf_Ptc, etapoly,
     *            fvec, xvec, dxvec, Ptcp, Patinj, etaad,
     *            tautInit, fInit, taufInit, alphaInit, picpInit,
     *            P0, M0, Tt4, tht, th0, tauc, Tt2, taucp, Ptc, Pt4,
     *            pic, Pt0, Tcinp, eps, W, pit, etat, pif, TurbineDump,
     *            FanDump, picf, M8, taucf, T8_T0, u8_a0, F8_ma0, M6,
     *            T6_T0, u6_a0, F6_ma0, Ftot_mtota, Isp, taus, N,
     *            ChokedNoz, P8_P0, P6_P0, P7,Uncooled, FilmCooled,
     *            utarget_a0, FiveEQFlag, NegWorkFlag
```