Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, July 1, 2010, Volume 224, Number 7, Pages 803-815

A dynamic convergence control scheme for the solution of the radial equilibrium equation in through-flow analyses

V Pachidis[1*], **I Templalexis**[2], **P Pilidis**[1], and **P Kotsiopoulos**[2]

[1]Cranfield University, School of Engineering, Department of Power and Propulsion, Gas Turbine Engineering Group, Cranfield, Bedfordshire, MK43 0AL, England

[2]Hellenic Air Force Academy, Section of Thermodynamics, Power and Propulsion Systems, Hellenic Air Force, Dekeleia Air Base, Greece

Abstract: One of the most frequently encountered numerical problems in scientific analyses is the solution of non-linear equations. Often the analysis of complex phenomena falls beyond the range of applicability of the numerical methods available in the public domain, and demands the design of dedicated algorithms that will approximate, to a specified precision, the mathematical solution of specific problems. These algorithms can be developed from scratch or through the amalgamation of existing techniques. The accurate solution of the full Radial Equilibrium Equation (REE) in Streamline Curvature (SLC) through-flow analyses presents such a case. This manuscript discusses the development, validation and application of an 'intelligent' Dynamic Convergence Control (DCC) algorithm for the fast, accurate and robust numerical solution of the non-linear equations of motion for two-dimensional flow fields. The algorithm was developed to eliminate the large user intervention, usually required by standard numerical methods. The DCC algorithm was integrated into a turbomachinery design and performance simulation software tool and was tested rigorously, particularly at compressor operating regimes traditionally exhibiting convergence difficulties (i.e. far off-design conditions). Typical error histories and comparisons of simulated results against experimental are presented in this manuscript for a particular case-study. For all case-studies examined, it was found that the algorithm could successfully 'guide' the solution down to the specified error tolerance, at the expense of a slightly slower iteration process (compared to a conventional Newton-Raphson scheme). This hybrid DCC algorithm can also find use in many other engineering and scientific applications that require the robust solution of mathematical problems by numerical, instead of analytical means.

Keywords: Numerical Analysis, Dynamic Convergence Control, Streamline Curvature, Radial Equilibrium Equation

1 INTRODUCTION

Numerical methods are usually based on calculating an approximation to the true value of a root of the equation $f(x)=0$ and then successively refining this approximation until further refining would achieve no useful purpose [**1**]. Also included in this type of problem is the determination of the points of intersection of two curves.

** Corresponding author: Dep. Director of the Cranfield Rolls-Royce UTC in Performance Engineering, Department of Power and Propulsion, Gas Turbine Engineering Group, Cranfield, Bedfordshire MK43 0AL, UK. email: v.pachidis@cranfield.ac.uk*

If the curves are represented by functions $f(x)$ and $g(x)$ respectively, the intersection points correspond to the roots of the function $F(x)=f(x)-g(x)$. The vast majority of the root-solving techniques available today is of this iterative nature. An initial guess for the root is specified, or an interval that is known to contain a root, and the various numerical schemes will return an improved guess or a more limited interval. The same procedure is then repeated using the new values until a root of desired accuracy is obtained or until the method encounters difficulties and fails. Although some of the schemes are guaranteed to find a root eventually, they may take considerable computer time to arrive at the answer. On the other hand, faster schemes may be able to converge to a root quicker, but tend to be susceptible to problems of divergence [**2**].

The physical characteristics of the computational platform used, as well as the nature itself of the problem being solved, can impose major limitations on the numerical scheme applied, and ultimately determine its success or failure. The effects of these constraints need to be both predictable and controllable. An excellent introduction to the theory of numerical analysis is given in references [**3, 4**]. References [**5-7**] cover the detailed aspects of more involved computational techniques, their range of applicability and their limitations. The practical application of numerical analyses by means of computer algorithms is covered well in [**8-10**]. To facilitate the solution of specific mathematical problems, e.g. the solution of the full REE in through-flow analyses, computer algorithms such as those described in [**11-15**] for numerical integration, or the solution of partial differential equations, can provide useful inspiration for the development of more advanced, hybrid computational techniques.

1.1 Solution of the REE in Through-Flow Analyses

The SLC through-flow method basically considers the flow within the compressor as axi-symmetric, compressible and inviscid. It solves the discrete equations of continuity, motion, energy and state in a form that incorporates the full three-dimensional compressor geometry and on a computational grid which is constructed in the meridional plane. After the transformation of the discrete equations and the solution of a system of equations, the result is a non-linear partial differential equation known as the REE [**16**]. This equation represents the gradient of the meridional velocity in the span-wise direction and needs to be solved by a finite difference approximation via an iterative approach.

Equation 1 represents the most general form of the REE [**17**] for the case of axi-symmetric inviscid flow, and can be used to describe the flow both into and out of stator and rotor blade rows with minor alterations (i.e. for stators relative velocity terms are replaced by the absolute velocity terms, the relative angle β is replaced by the absolute angle α and the rotational speed is set to zero).

$$V_m\frac{dV_m}{ds}=V_m^2\begin{pmatrix}\sin(\varepsilon-\gamma)\tan(\varepsilon+\gamma)\frac{d\varepsilon}{dm}-\\ -\frac{d\varepsilon}{ds}\frac{\sin(\varepsilon-\gamma)}{\cos(\varepsilon+\gamma)}-\\ -\frac{\sin\varepsilon\sin(\varepsilon-\gamma)}{r}-\\ -\frac{1}{\rho}\frac{d\rho}{dm}\sin(\varepsilon-\gamma)+\\ +\frac{\cos(\varepsilon-\gamma)}{r_c}\end{pmatrix}+$$
$$+V_m\begin{pmatrix}\tan\lambda\frac{d(rW_w)}{dm}\frac{1}{r}+\\ +2\omega\sin\varepsilon\tan\lambda\end{pmatrix}+$$
$$+\begin{pmatrix}-\frac{W_w}{r}\frac{d(rW_w)}{ds}-2\omega W_w\cos\gamma+\\ +\frac{dI}{ds}-T\frac{dS}{ds}+\\ T\frac{dS}{dm}\begin{pmatrix}\sin(\varepsilon-\gamma)\cos^2\beta-\\ -\tan\lambda\sin\beta\cos\beta\end{pmatrix}\end{pmatrix} \quad (1)$$

Equation 1 is a second order differential equation that can be broken down into the following terms:

$$A=\begin{pmatrix}\sin(\varepsilon-\gamma)\tan(\varepsilon+\gamma)\frac{d\varepsilon}{dm}-\\ -\frac{d\varepsilon}{ds}\frac{\sin(\varepsilon-\gamma)}{\cos(\varepsilon+\gamma)}-\frac{\sin\varepsilon\sin(\varepsilon-\gamma)}{r}\\ -\frac{1}{\rho}\frac{d\rho}{dm}\sin(\varepsilon-\gamma)+\frac{\cos(\varepsilon-\gamma)}{r_c}\end{pmatrix} \quad (2)$$

$$B=\left(\tan\lambda\frac{d(rW_w)}{dm}\frac{1}{r}+2\omega\sin\varepsilon\tan\lambda\right) \quad (3)$$

$$C=\begin{pmatrix}-\frac{W_w}{r}\frac{d(rW_w)}{ds}-2\omega W_w\cos\gamma+\\ +\frac{dI}{ds}-T\frac{dS}{ds}+\\ T\frac{dS}{dm}\begin{pmatrix}\sin(\varepsilon-\gamma)\cos^2\beta-\\ -\tan\lambda\sin\beta\cos\beta\end{pmatrix}\end{pmatrix} \quad (4)$$

Substituting, Equation 1 can be reduced to:

$$V_m\frac{dV_m}{ds}=AV_m^2+BV_m+C \quad (5)$$

In this case the solution of the equation has as follows:

$$\frac{\ln\left(AV_m^2+BV_m+C\right)}{2A}- -\frac{B}{A\sqrt{4AC-B^2}}\tan^{-1}\left(\frac{2AV_m+B}{\sqrt{4AC-B^2}}\right)=s+c \tag{6}$$

The constant of integration c can be calculated by applying equation 6 for two consecutive stream-lines as below:

$$\frac{1}{2A}\left(\ln\left(\frac{AV_{m,j+1}^2+BV_{m,j+1}+C}{AV_{m,j}^2+BV_{m,j}+C}\right)\right)- \frac{B}{A\sqrt{4AC-B^2}}\left(\tan^{-1}\left(\frac{2AV_{m.j+1}+B}{\sqrt{4AC-B^2}}\right)\right)- -\left(\tan^{-1}\left(\frac{2AV_{m.j}+B}{\sqrt{4AC-B^2}}\right)\right)=s_{j+1}-s_j \tag{7}$$

The terms A, B and C contain, apart from certain flow parameters, derivatives along the s and m directions. In order to specify an initial value for the derivative terms, a flow pattern including all the related parameters has to be assumed throughout the compressor. Moreover, the derivatives along the s direction at certain streamline locations can only be updated once the meridional velocity profile is obtained at the corresponding plane (this can be the leading or the trailing edge of a blade). On the other hand, in order for the meridional derivatives to be updated, the meridional velocity profile of the entire compressor needs to be determined first.

In order to start the iteration, computation nodes can be defined for example at the intersections of the streamlines and the blade edges. Obviously, at the beginning, node coordinates are unknown since the streamline positions have not been determined yet. A certain radial position for the streamlines is assumed throughout the compressor's effective flow area, as well as a certain meridional velocity distribution. Then, a meridional velocity profile is defined along every compressor blade leading and trailing edge, such as to satisfy both the REE and the mass flow continuity. In order to achieve this double convergence, a meridional velocity profile is determined first, based on an arbitrary value of meridional velocity at a certain compressor height and then it is successively recalculated until all the streamtube mass flows are satisfied. The REE basically reflects the equation of the pressure forces to the inertial forces. Stream-tubes that contain a fixed amount of mass, as they are confined between two streamlines, are successively refined regarding their radial position in order for the REE to be satisfied.

The overall solution process is obviously not a straight forward one and is largely based on a trial and error approach, involving usually a large number of iterative loops, both nested and crossing over. The whole iterative scheme is by default very complicated and potentially troublesome, especially as far as specifying initial conditions and maintaining a stable and robust convergence is concerned.

During the forty years approximately of the existence of the SLC method, numerous authors have proposed several variations of the SLC calculation scheme [**18-24**]. All these different schemes were mainly influenced by the type of the turbomachine the method was applied to (radial or axial), the nature of the flow being considered (subsonic or supersonic) and the level up to which the flow viscosity and circumferential in-homogeneities were taken into account. In many cases, the various calculation schemes were also influenced by the particular characteristics of the cascade, such as hub to tip ratio, or lean and sweep angle distributions [**25**]. Nevertheless, in all these publications very little is usually mentioned as to the actual numerical scheme or algorithm being used to facilitate convergence. Perhaps, some more useful information can be obtained from [**26-29**], but again not sufficient in its own merit for a wider application in a more generic context.

The authors of this manuscript have included the next two sections (*Section 1.2 and Section 1.3)* in order to capture in a synopsis the reasons behind this work and make this document easier to follow.

1.2 Newton's Method - Background and Overview

Newton's (or the Newton-Raphson) method can usually be relied upon to find a solution quickly and accurately [**30**]. It is used routinely in the solution of complex engineering and scientific problems, as well as in through-flow calculations. In its pure form, it derives from the Taylor series for the function $f(x)$.

$$f(x)=f(x_0)+\left.\frac{df}{dx}\right|_{x=x_0}(x-x_0)+\frac{1}{2!}\left.\frac{d^2f}{dx^2}\right|_{x=x_0}(x-x_0)^2+... \tag{8}$$

If $|x-x_0|$ is small enough, only a few terms in the above series need to be retained. To find a root of the function, an x needs to be identified such that $f(x)=0$ or :

$$f(x)=0=f(x_0)+\left.\frac{df}{dx}\right|_{x_0}(x-x_0)+\frac{1}{2!}\left.\frac{d^2f}{dx^2}\right|_{x_0}(x-x_0)^2+... \tag{9}$$

If it is assumed that the desired root x is near the value x_0, then:

$$f(x)=0\cong f(x_0)+\left.\frac{df}{dx}\right|_{x_0}(x-x_0) \tag{10}$$

If x is not near x_0 this may not even be approximately true. In the last equation everything except x (the root) is known and so we can solve for x considering that $f'=df/dx$. The last equation then becomes:

$$x = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{11}$$

If $|x - x_0|$ is indeed small, then the last equation will be a good estimate of the actual root of the function. But if $|x - x_0|$ is not small, then the solution will lie far from the actual root of the function. Replacing the function $f(x)$ by the first two terms in its Taylor series approximates the function by a straight line through the point (x_0, $f(x_0)$), which has the same slope as the tangent to the curve at that point (Fig. 1). Then setting this approximation to $f(x)$ = 0, gives the point where the line intersects the axis. Although the value generated is closer to the actual root than was the starting point x_0, this procedure will not in general give the actual root of $f(x)$ in one iteration. Newton's method simply consists of repeating this process. That is, starting from an initial guess for the root of $f(x)$, say x_0, calculate an improved guess x_1, and then use the improved value x_1 for the root in the next cycle to calculate a new improvement on the root.

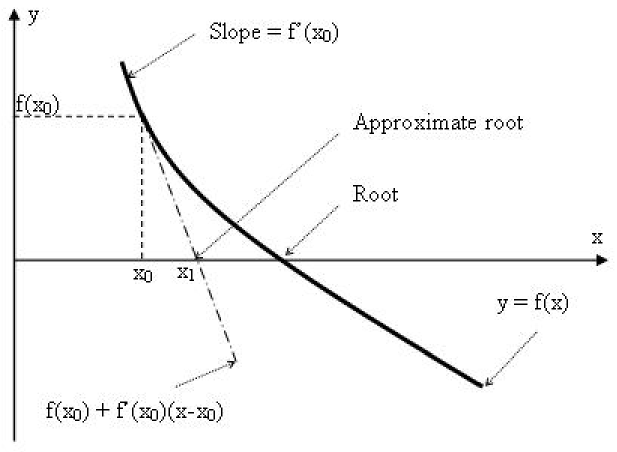

Fig. 1 The Newton-Raphson iteration scheme

1.3 Newton's Method - Limitations

Newton's method attempts to find a root of a function $f(x)$ by repeatedly approximating the function by straight lines. Compared to other numerical schemes, this method incorporates a lot of information about the behaviour of the function into the root-solving algorithm. Specifically, the method monitors at each step not only the value of the function but also its slope (the derivative of the function). It is this particular feature that results in a dramatic improvement in the rate of convergence. However, Newton's method differs from other procedures in that it does not guarantee that a root will be found in all cases, and hence, it will often tend to diverge or completely fail, especially when applied to complex functions having an unpredictable behaviour. There are a few potential problems that can cause the method to fail. These are explained and also graphically illustrated in the following figures (Figs. 2-5). First of all, if the slope of the function is nearly horizontal and the initial guess of x_0 is so poor that $f'(x_0)$ is very small, the first iteration may be thrown out of the region of interest and the solution would then diverge (Fig. 2). Also, the method will likely fail if the initial guess is in a region where the function has a local minimum but no root (Fig. 3).

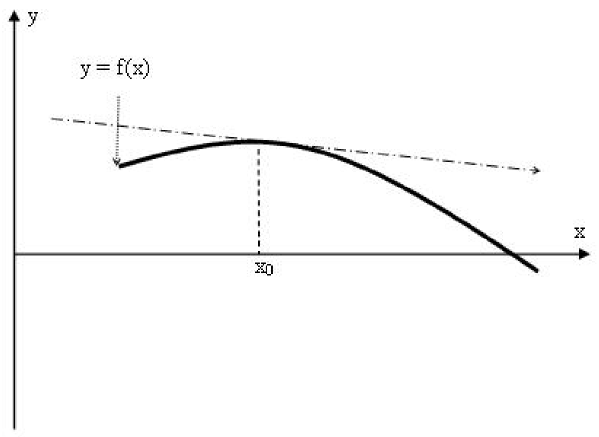

Fig. 2 A horizontal slope may cause a failure

Moreover, the method will not be able to find a root whenever the derivative at the root is infinite (Fig. 4). The method would also have difficulties with multiple roots, as Figure 5 illustrates. At the position of a multiple root, both $f(x)$ and $f'(x_0)$ become nearly zero and therefore the algorithm would exhibit a very slow convergence and ultimately fail when attempting to divide zero by zero.

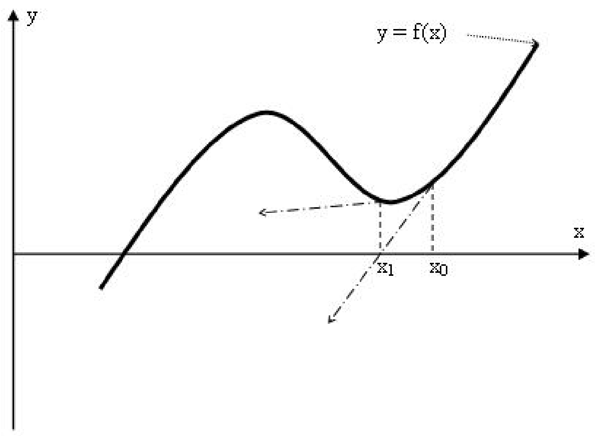

Fig. 3 A local minimum may cause a failure

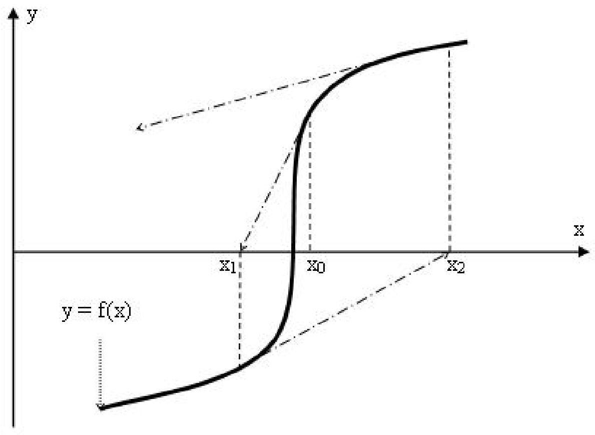

Fig. 4 A vertical slope near a root may cause a failure

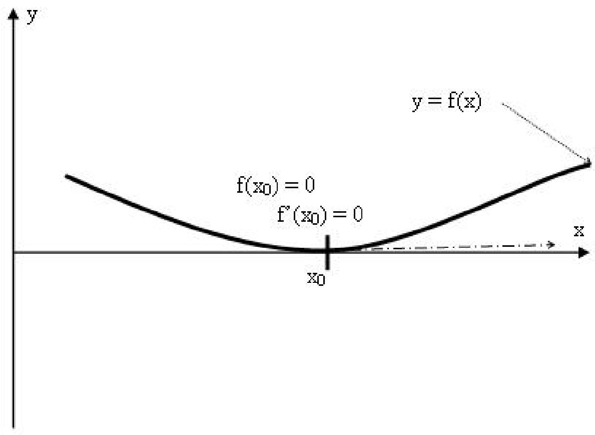

Fig. 5 A multiple root may cause a failure

2 THE HYBRID DCC ALGORITHM

In all numerical schemes potential computational difficulties of any nature can be avoided by gathering as much information as possible in the initial choice of the method used and the accompanying initial guess. Also a rough understanding of the behavior of the function itself should be obtained (analytically or graphically) before root solving is attempted. In practice however, the exact nature of the function is usually unknown. In these cases, Newton's method in its original form as described previously, can not work and a different method, or combination of methods, requiring less initial information about the nature of the function being solved, needs to be implemented. Moreover, a method of monitoring the function needs to be utilized to arrive at some understanding of how the function is behaving during computation.

This is the general approach taken by the work reported here. The hybrid DCC algorithm is largely based on Newton's method but addresses effectively the inherent limitations of the method described in *Section 1.3*. It also combines the speed of execution of the method with a slower, but more robust numerical scheme which is only employed while the solution is close to the true root and when numerical instabilities are encountered. Furthermore, the algorithm includes a monitoring facility that can track the evolution of errors generated during successive iterations and take appropriate intervening action during run-time. More specifically the algorithm:

- Dynamically monitors the convergence history of the various iterative processes during code execution
- Automatically detects the nature of the convergence mode that the solution has fallen into (i.e. stable convergence, divergence, various types of numerical oscillations etc.)
- Dynamically changes the numerical scheme employed and also the rate of convergence according to the detected mode
- Traps diverging iterations and attempts to return the solution to steady convergence, hence preventing it from diverging any further
- Prevents the software from crashing due to numerical instabilities
- Dynamically 'drives' the solution down to the specified error tolerance by intervening appropriately throughout
- Informs the user when a particular solution has steadily converged above the specified error tolerance and prompts for action
- When a particular iteration converges below the specified error tolerance, it automatically resets all initial settings back to their default values for the monitoring of the next iterative process

More details are given in the following section.

2.1 Algorithm Inner-Workings

As discussed previously, Newton's method requires the values of the function itself, as well as the values of the first derivative of the function. Calculating the derivative of Equation 1 or Equation 7 can be extremely computationally expensive (if not impossible). Finding the solution of this function is also only an incidental part of a much larger problem and not the sole objective of a through-flow computation.

We could therefore adopt an easier strategy and replace the derivative with an estimate, obtained numerically from the slope of the line joining the last two iterates. The point where this line, called the 'secant' (hence the name of this method), cuts the x-axis is taken to be the next iterate. This approach requires two values of x to start the iteration. These two values of x can be closely spaced together and need not to be bracketing the root. In other words, specifying these two initial guesses should not be difficult at all. An approximate expression of the derivative can be obtained, for example, from the first backward difference as follows:

$$f'(x_0) = \frac{f(x_0) - f(x_0 - \Delta x)}{\Delta x} \tag{12}$$

Using Equation 12, Equation 11 can then be modified accordingly.

Since the 'secant' method is based on Newton's method it will exhibit similar vulnerabilities and tendency to fail in those cases discussed in *Section 1.3*. First of all, the slow convergence rate at multiple roots can be addressed as follows. Newton's method will fail at a multiple root because of the operation of dividing zero by zero, a forbidden operation on any computer. In mathematics however, the operation 0/0 is not forbidden; it is simply an undetermined form (neither 0 nor infinity). It has no numerical value until it is defined to be a specific value in a particular problem [**30**]. In addition to suffering from a slow convergence rate for multiple roots, Newton's method may yield a result that is invalidated by round-off errors caused when both the value of the function and its derivative are extremely small.

Amending the previous procedure, if we consider for example the following simple function $f(x)$ that has a multiple root of multiplicity m at $x = r$, we can write:

$$f(x) = (x - r)^m g(x) \tag{13}$$

Since the value of r is yet unknown, $g(x)$ is likewise unknown. We can, however, assume that $g(x)$ is not zero (or extremely small) at $x = r$. Then applying Newton's method with an initial guess of x_0 near r, we have:

$$f(x_0) = (x_0 - r)^m g(x_0) \tag{14}$$

$$\begin{aligned} f'(x_0) &= m(x_0 - r)^{m-1} g(x_0) + (x_0 - r)^m g'(x_0) = \\ &= (x_0 - r)^m g(x_0)\left[\frac{m}{x_0 - r} + \frac{g'(x_0)}{g(x_0)}\right] = \\ &= f(x_0)\left[\frac{m}{x_0 - r} + \frac{g'(x_0)}{g(x_0)}\right] \end{aligned} \tag{15}$$

The assumptions made regarding x_0 and $g(x)$ ensure that the first term in the brackets in the last equation is much larger than the second. Thus we can approximate this equation as:

$$f'(x_0) \cong f(x_0)\frac{m}{x_0 - r} \tag{16}$$

Solving this for the root r, we obtain:

$$r = x_0 - m\frac{f(x_0)}{f'(x_0)} \tag{17}$$

This is almost the same as the original Newton algorithm, with the replacement $x \to x - (f / f')$ becoming $x \to x - m(f / f')$ for a function with a root of multiplicity m. With this simple amendment to Newton's method the desired convergence rate is restored.

As mentioned previously, the other inherent limitations of Newton's method have been addressed through the implementation of an error monitoring facility and a more robust numerical scheme. This additional numerical scheme, being somewhat slower, is employed only towards the last stages of convergence, when a solution is found to orbit dangerously around a root, and when numerical instabilities are encountered. More specifically, at the beginning of an iteration, generated errors are stored in a memory register. After a specified number of loops this error history is examined by the algorithm and unless the convergence process is stable, the Newton-Raphson/Secant numerical scheme switches to:

$$x_1 = x_0\left(1 - f_{relax}\left(\frac{f(x_0) - f_{root}}{f(x_0)}\right)\right) \tag{18}$$

Where f_{relax} is a relaxation factor and f_{root} is the target solution. For example, when applying this scheme for the solution of the REE, f_{root} becomes a particular Δs (or s coordinate) in the tangential direction along the blade span, and x_1 becomes a meridional velocity value that can satisfy the REE at that particular coordinate.

When the iteration does not reach convergence after a specified number of loops, f_{relax} needs to be reduced. Similarly, if successive iterates are found to oscillate around a solution, f_{relax} is reduced again to help meet the error tolerance requirements. When the solution is diverging, f_{relax} is reduced and its sign is reversed in order to trap the solution and re-attempt convergence, starting however from a different set of initial conditions this time.

Practical application has shown that the ideal starting value of the relaxation factor depends on the nature of the iteration conducted. Recommended values may range from 0.1 to 0.001. Experience has also shown that subsequent reductions of f_{relax}, of the order of 2, when numerical instabilities are encountered, can facilitate successful convergence to error tolerances as low as 1.0E-7 at reasonable convergence rates. Equation 18 and the fairly straightforward mathematical manipulation of f_{relax}, facilitates control over the actual convergence process and its rate, maximizing the chances for reaching the correct solution without being hindered by the limitations of the numerical scheme employed. The following section discusses in more detail the practical application of the hybrid DCC algorithm.

2.2 Testing and Validation of the DCC Algorithm

The hybrid DCC algorithm is a versatile tool that can have many applications. Nevertheless, it was mainly developed in the context of this work to meet the requirements imposed by SLC-type of through-flow analyses in turbomachinery components. It was therefore fully integrated into *SOCRATES* (***S****ynthesis* ***O****f* ***C****orrelations for the* ***R****apid* ***A****nalysis of* ***T****urbomachine* ***E****ngine* ***S****ystems*), a turbomachinery design and performance simulation tool developed by researchers at Cranfield University [**25, 31-34**] (Fig. 6). The hybrid DCC algorithm was called by the code during run-time to handle the execution of several different iterative loops. For example it handled the solution of the REE at hundreds of computation nodes, the stream-tube and overall mass flow convergence processes, the change in streamline radii, the calculation of static temperature, C_p and γ from Mach number etc.

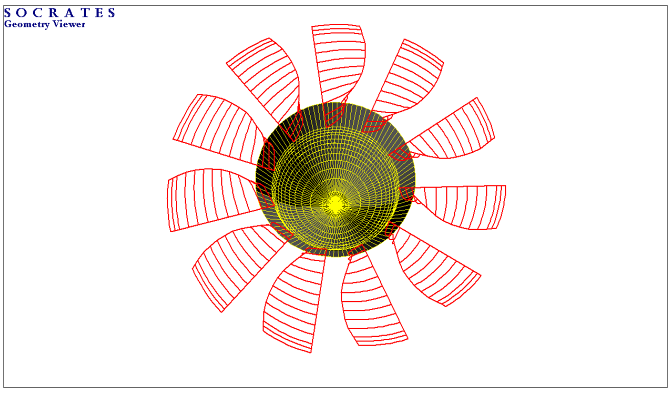

Fig. 6 Example of *SOCRATES'* turbomachinery design environment – single stage fan with cone

Figure 7 illustrates the complete error history of several iterative processes, as evolved during a typical code execution for the performance analysis of a 2-stage compressor (its performance is discussed in more detail in *Section 3*). By zooming in closer to the x-axis in the above figure, Figures 8 to 12 provide typical samples of the operation of the DCC algorithm. For example in the first case (Fig. 8), after the rapid convergence to a moderate error tolerance using Newton's method, the controller detects the diverging solution and intervenes accordingly, so that a steady converged solution is achieved at a smaller tolerance value after a few iterations. Similarly, Figures 9 and 10 illustrate converged cases where the solution was finally driven by the dynamic controller to lower absolute values of error tolerance.

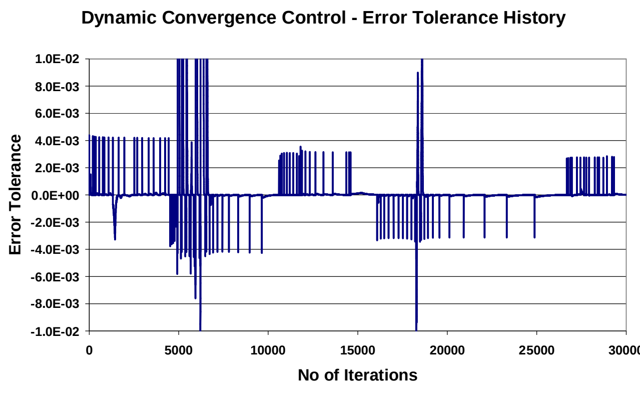

Fig. 7 Typical overall convergence history during code execution for the performance analysis of a 2-stage compressor

Figure 11 illustrates an example of successful convergence without DCC intervention, and Figure 12 again demonstrates the ability of the algorithm to help achieve really small error tolerances even after a diverging solution.

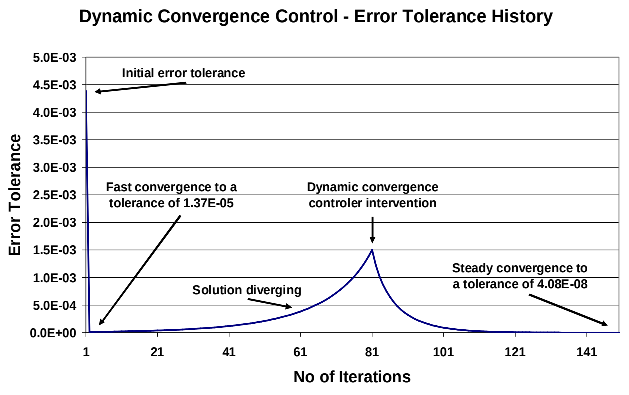

Fig. 8 Typical convergence history with DCC algorithm intervention after a divergent solution

Testing rigorously the performance of the hybrid DCC algorithm over a wide range of compressor operating conditions, even far off-design, led to the conclusion that dynamic convergence control can be a very versatile and valuable asset in the toolkit of an SLC software, improving significantly its capabilities and overall robustness. Actual numerical results, compared against experimental data are presented in the next section in support of this conclusion.

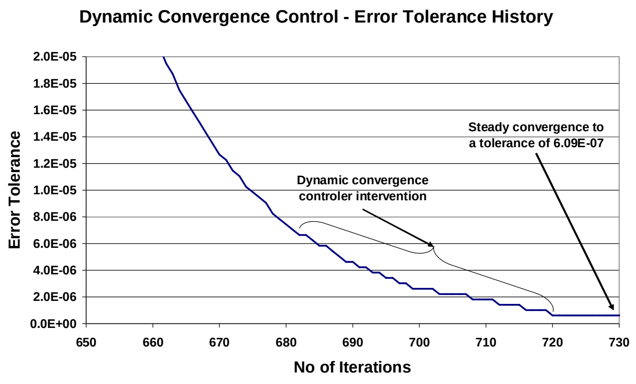

Fig. 9 Typical convergence history with DCC algorithm intervention to achieve an even smaller error tolerance (absolute value)

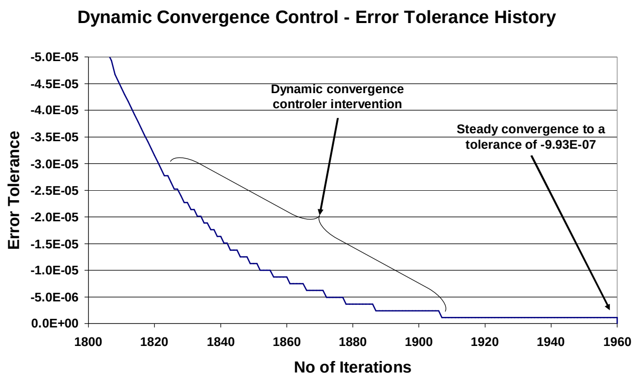

Fig. 10 Typical convergence history with DCC algorithm intervention to achieve an even smaller error tolerance (absolute value)

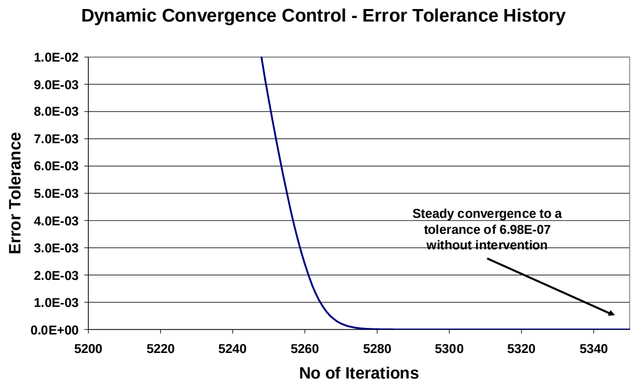

Fig. 11 Typical convergence history using Newton’s method with no DCC algorithm intervention

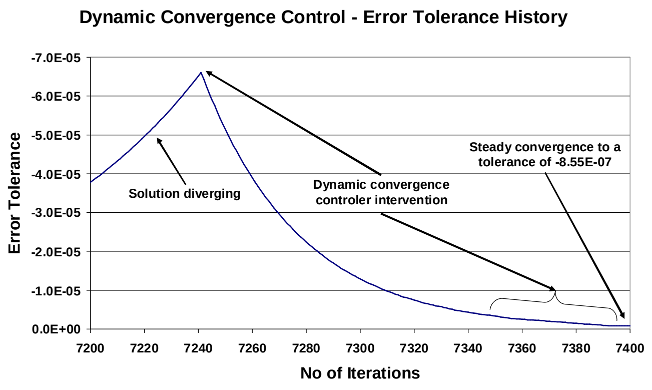

Fig. 12 Typical convergence history with DCC algorithm intervention after a divergent solution

3 APPLICATION IN *SOCRATES*

The *SOCRATES* turbomachinery design and performance analysis tool is equipped with several different numerical schemes largely from the public domain. These numerical schemes, having a different range of applicability and different limitations, have to be ‘mixed and matched’ by the user, according to the needs of the case being studied. This can turn into a fairly laborious, ‘trial and error’ exercise, requiring unnecessarily in the process the user to become familiar with the inner-workings of the various schemes available.

The integration of the DCC algorithm into *SOCRATES* resulted in an enhanced version of the code which was subsequently tested at far off-design conditions, traditionally exhibiting convergence challenges. The analysis of compressor performance at rotational speeds away from the design point, where the flow field can be approximated reasonably well, typically presents convergence difficulties, since any initial guessed variables may have to be ‘initialized’ away from their true values.

The objectives of this exercise were mainly to i) to assess the stability and robustness of the new iteration scheme, particularly at extreme conditions, and ii) to validate the results against experimental data. The comparison against experimental results was carried out based on [**35**]. This technical report provides a very detailed description of measured flow field data for a NASA experimental two-stage compressor, from hub to tip and for a number of different compressor operating conditions (Table 1).

Pressure Ratio	2.399
Temperature Ratio	1.334
Isentropic Efficiency	0.849
Corrected Mass Flow [kg/s]	33.248
rpm	16042.800
Inlet Hub to Tip Ratio	0.375
1st Rotor Tip Speed [m/s]	428.896
2nd Rotor Tip Speed [m/s]	405.341

Table 1 NASA two-stage compressor design overall parameters

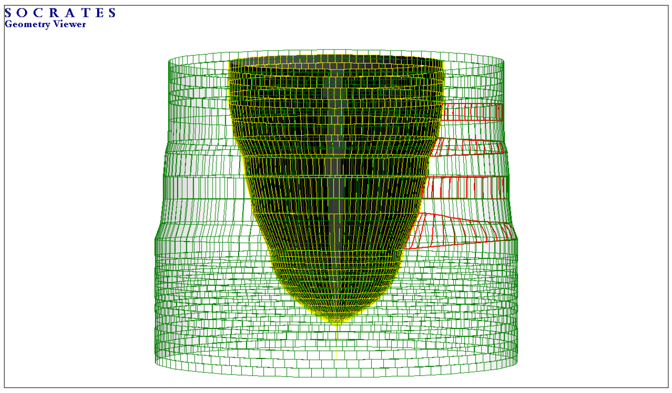

Fig. 13 3D model of the NASA two-stage compressor in *SOCRATES*

A two-stage compressor model (Fig. 13) was constructed in *SOCRATES* according to the detailed geometric data published in [**35**]. It is worth pointing out that *SOCRATES* is based on an inviscid flow analysis that is coupled with various loss, deviation and boundary layer models. Previous publications by the same authors [**25, 31-34**] reported thoroughly on the synthesis of the various viscous and loss models that have been included in this particular software over the years. For the analysis reported here, loss and deviation data were used from [**35**].

Using the new version of the software tool, 2D profiles of various flow properties for all four blade rows were calculated and compared against the measured performance data provided in the aforementioned report. Figures 14 and 15 present the comparison between the measured (NASA) and simulated (SLC) meridional velocity variation, at the inlet and outlet of all four blade rows and at 50% design rotational speed. The span-wise meridional velocity variation is obviously the direct outcome of the iterative solution of the REE. Similarly, Figures 16 and 17 present comparisons of the absolute velocity span-wise distribution at the same power setting.

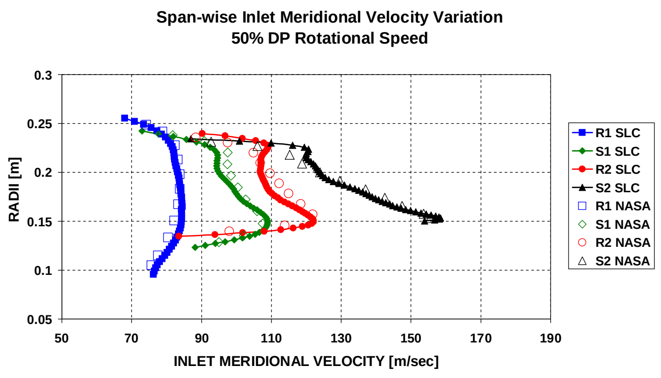

Fig. 14 Inlet meridional velocity variation from hub to tip at 50% DP rotational speed

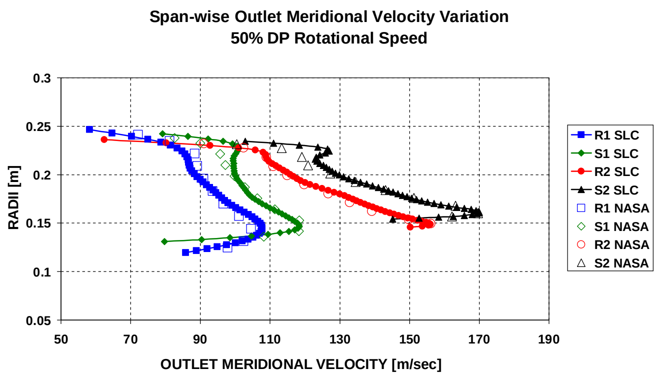

Fig. 15 Outlet meridional velocity variation from hub to tip at 50% DP rotational speed

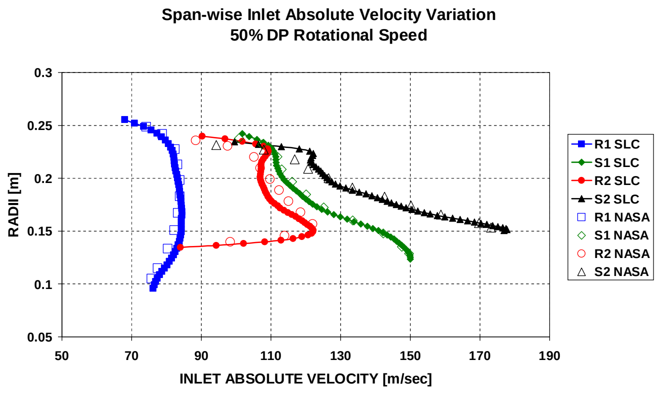

Fig. 16 Inlet absolute velocity variation from hub to tip at 50% DP rotational speed

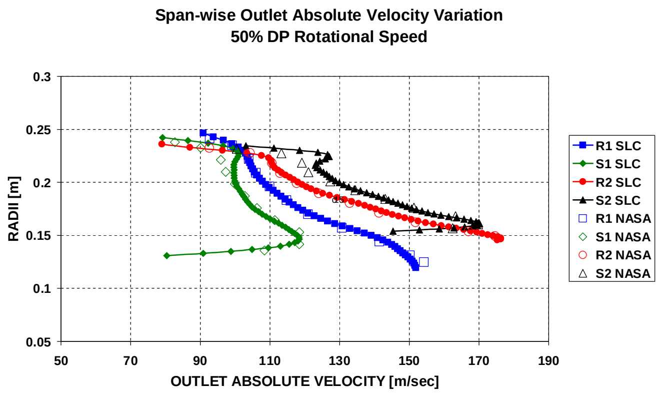

Fig. 17 Outlet absolute velocity variation from hub to tip at 50% DP rotational speed

In all cases examined, the incorporation of the DCC algorithm enabled a fast and faultless convergence, without requiring any manual intervention. More importantly, even at part-speed operation, the comparison of the calculated compressor performance against experimental data revealed that reasonable results can be obtained, meaning that the algorithm is not just converging randomly on unrealistic solutions. Actually, the calculated flow properties were found to be in a very good overall qualitative and quantitative agreement with the experimental measurements, as illustrated above.

4 MAIN CONCLUSIONS

This manuscript discusses the development, validation and deployment of an 'intelligent' DCC algorithm for the fast, accurate and robust numerical solution of the non-linear equations of motion for two-dimensional flow fields. The algorithm was specifically developed to address the computational challenges presented by SLC-type of analyses, and in particular the fast and accurate solution of the full REE.

This hybrid DCC algorithm is largely based on Newton's method but addresses effectively the inherent limitations of the method. It also combines the speed of execution of the method with a slower, but more robust numerical scheme introduced by the authors. This is only employed while the solution is very close to the true root and when numerical instabilities are encountered. Furthermore, the algorithm includes a monitoring facility that can track the evolution of errors generated during successive iterations and take appropriate intervening action during run-time, thus eliminating the large user intervention, usually required by standard numerical methods.

The DCC algorithm was subsequently integrated into a turbomachinery design and performance analysis tool and was tested rigorously, particularly at compressor operating regimes traditionally exhibiting convergence difficulties (i.e. part-speed performance). For all case-studies examined, the DCC algorithm enabled a fast and crash-free convergence, 'guiding' successfully the solution down to the specified error tolerance, at the expense of a slightly slower iteration process (compared to a conventional Newton-Raphson scheme). Even at part-speed operation, the comparison of the calculated compressor performance against experimental data revealed a very good qualitative and quantitative agreement.

Overall, the hybrid DCC algorithm, developed in the context of this work, was found to provide considerably enhanced convergence stability and robustness features. It was proven to be a versatile and valuable asset in the toolkit of through-flow analysis software but it can also find other applications in engineering as well, particularly wherever complex numerical analyses need to be carried out.

REFERENCES

1 **Faddeev, D. K.** and **Faddeeva, V. N.** Computational Methods of Linear Algebra, San Francisco: Freeman, 1963.

2 **Todd, J.** Basic Numerical Mathematics, 2 vols, New-York: Academic Press, 1978-1980.

3 **Hildebrand, F. B.** Introduction to Numerical Analysis, 2nd Edition, New-York: McGraw-Hill, 1974.

4 **Todd, J.** Survey of Numerical Analysis, New-York: McGraw-Hill, 1962.

5 **Dahlquist, G.** and **Björck, A.** Numerical Methods, Englewood Cliffs, NJ: Prentice-Hall, 1974.

6 **Isaacson, E.** and **Keller, H. B.** Analysis of Numerical Methods, New-York: Wiley, 1973.

7 **Scheid, F.** Theory and Problems of Numerical Analysis, New York: McGraw-Hill, 1968.

8 **Forsythe, G. E., Malcolm, M. A.** and **Moler, C. B.** Computer Methods for Mathematical Computations, Englewood Cliffs, NJ: Prentice-Hall, 1977.

9 **Hennie, F.** Introduction to Computability, Reading, MA: Addison-Wesley, 1977.

10 **Horowitz, E.** and **Sahni, S.** Fundamentals of Computer Algorithms, Rockville, MD: Computer Science Press, 1985.

11 **Smith, G. D.** Numerical Solution of Partial Differential Equations, London: Oxford University Press, 1974.

12 **Collatz, L.** The Numerical Treatment of Differential Equations, 3rd Edition, New York: Springer, 1966.

13 **Davis, P.** and **Rabinowitz, P.** Methods of Numerical Integration, 2nd Edition, New York: Academic Press, 1984.

14 **Traverso, A.** and **Massardo, A. F.** Optimal design of compact recuperators for microturbine application, 13594311, Applied Thermal Engineering, 2005.

15 **Massardo, A. F.** and **Scialo, M.** Thermoeconomic analysis of gas turbine based cycles, Journal of Engineering for Gas Turbines and Power, 2000.

16 **Smith, L. H.** The Radial Equilibrium Equation of Turbomachinery, Trans. A.S.M.E., Series A, Vol 88, 1966.

17 **Templalexis, I., Pilidis, P., Pachidis, V.** and **Kotsiopoulos, P.** Development of A 2D Compressor Streamline Curvature Code, Transactions of the ASME, Journal of Turbomachinery, TURBO-06-1178, Vol. 129, Issue 4, October 2007. ASME Education Committee Best Paper Award for 2006.

18 **Wu, C. H.** A General Through-Flow Theory Of Three-Dimensioanl Flow In Subsonic And Supersonic Turbomachines Of Axial-, Radial-, And Mixed-Flow Types, NASA TN2604, 1952.

19 **Barbosa, J. R.** A Streamline Curvature Computer Program For Axial Compressor Performance Prediction, Ph.D Thesis, Vol. 1, Cranfield Institute of Technology, School of Mechanical Engineering, 1987.

20 **Boyer, K. M.** An Improved Streamline Curvature Approach for Off Design Analysis of Transonic Compression Systems, Ph.D Thesis, Virginia Polytechnic Institute and State University, 2001.

21 **Denton, J. D.** Through-flow Calculations For Transonic Axial Flow Turbines, Transactions of the ASME, Journal of Engineering for Power, Vol. 100, p. 212-18, 1978.

22 **Frost, D. H.** A Streamline Curvature Through-Flow Computer Program for Analysing the Flow Through Axial-Flow Turbomachines, Aeronautical Research Council, R&M 3687, 1972.

23 **Jennions, I. K.** and **Stow, P.** The Quazi-Three-Dimensional Turbomachinery Blade Design System, Part I: Throughflow Analysis, Part II: Computerized System, Transactions of the ASME, Journal of Engineering for Gas Turbines and Power Vol. 107, p. 308-16, 1985.

24 **Novak, R. A.** Streamline Curvature Computing Procedures For Fluid-Flow Problems, Transactions of the ASME, Journal of Engineering for Power, Vol. 89, p. 478-490, 1967.

25 **Templalexis, I., Pachidis, V., Pilidis, P.,** and **Kotsiopoulos, P.** The Effect of Blade Lean on the Solution of the Radial Equilibrium Equation, GT2008-50259, ASME Turbo Expo, Power For Land, Sea and Air, Berlin, Germany, June 2008.

26 **Wilkinson, D. H.** Stability, Convergence and Accuracy of 2-D Streamline Curvature Methods Using Quasi-Orthogonals, Proc. of the Institution of Mechanical Engineers, Vol. 184, 1969-70.

27 **Katsanis, T.** Use Of Arbitrary Quasi-Orthogonals for Calculating Flow Distribution in the Meridional Plane of a Turbomachine, NASA Technical Note D-2546, 1964.

28 **Katsanis, T.** Use Of Arbitrary Quasi-Orthogonals for Calculating Flow Distribution on a Blade-To-Blade Surface in a Turbomachine, NASA Technical Note D-2809, 1965.

29 **Wood, M. D.** and **Marlow, A. V.** The use of numerical methods for the investigation of the flow in water pump impellers, Proc. Institution of Mechanical Engineers, Vol. 181, Part 1, 1966-1967.

30 **Kreyszig, E.** Advanced Engineering Mathematics, 6th Edition, New-York: John Wiley & Sons, 1988.

31 **Pachidis, V., Pilidis, P., Templalexis, I., Alexander, T.** and **Kotsiopoulos, P.** Prediction of Engine Performance Under Compressor Inlet Flow Distortion Using Streamline Curvature, Transactions of the ASME, Journal of Engineering for Gas Turbines and Power, GTP-05-1192, Vol. 129, p. 97, January 2007.

32 **Pachidis, V., Pilidis, P., Texeira, J.,** and **Templalexis, I.** A Comparison of Component Zooming Simulation Strategies Using Streamline Curvature", Proceedings of the IMechE, Journal of Aerospace Engineering, JAERO147, Vol. 221, Part G, p. 1, 2007.

33 **Pachidis, V., Pilidis, P., Marinai, L.,** and **Templalexis, I.** Towards a Full Two Dimensional Gas Turbine Performance

Simulation, Proceedings of the RASoc, The Aeronautical Journal, AJ-3127, June 2007.

34 **Pachidis, V., Pilidis, P., Templalexis, I.,** and **Marinai, L.** An Iterative Method for Blade Profile Loss Model Adaptation Using Streamline Curvature, Transactions of the ASME, Journal of Engineering for Gas Turbines and Power, GTP-07-1072, Vol.130, Iss.1, December 2007. ASME Cycle Innovations Committee Best Paper Award for 2007.

35 **Urasek, D. C., Gorell, W. T.** and **Cunnan, W. S.** Performance Of Two-Stage Fan Having Low-Aspect-Ratio, First Stage Rotor Blading, NASA Technical Paper 1493, 1979.

LIST OF FIGURES

LIST OF TABLES

APPENDIX 1 - NOMENCLATURE

Abbreviations

DP	Design Point
REE	Radial Equilibrium Equation
SLC	Streamline Curvature
2D	Two-Dimensional

Symbols

A, B, C	Differential equation terms
F	Force
H	Enthalpy
I	Rothalpy
P	Pressure
S	Entropy
T	Temperature
V	Absolute air velocity
W	Relative velocity
c	Constant of integration
f	Function, factor
i,j,k	Unit vectors
m	Meridional direction
r	Radius, radial direction, root
r_c	Radius of curvature
s	Tangential along the blade edge direction
z	Axial direction

Greek Symbols

α	Absolute flow angle
β	Relative flow angle
γ	Sweep angle
ε	streamline slope angle
λ	Lean angle
ρ	Density
ω	Angular speed

Subscripts

D	Drag
P	Pressure
j	Streamline counter
m	Meridional direction
n	Normal direction
r	Radial direction
relax	Relaxation
root	The solution of a function
w	Whirl direction
z	Axial direction