# THE SONIFICATION OF NUMERICAL FLUID FLOW SIMULATIONS

*Edward Childs, Ph.D.*

Dartmouth College
The Bregman Electronic Music Studio
Electro-Acoustic Music
Department of Music
6187 Hopkins Center
Hanover, NH 03755-3599, USA
Edward.Childs@dartmouth.edu

**ABSTRACT**

Computational Fluid Dynamics (CFD) software simulates fluid, air flow and heat transfer by solving the Navier-Stokes (N-S) equations numerically. Realistic 3-D engineering simulations typically yield the values of 7 or more variables (e.g. fluid component velocities and temperatures) at hundreds of thousands of points in space, all as a function of time. It has been noted that solutions of the N-S equations sometimes yield highly complex, non-linear flow fields which can be aesthetically interesting from a purely visual standpoint.

The analysis of CFD results may benefit substantially from sonification, to depict convergence behavior, scan large amounts of data with low activity, or codify global events in the flow field. As a corollary to this interest in developing CFD sonification techniques, we can explore its unusual potential as a tool for algorithmic musical composition.

This paper will report the results of an initial implementation of the author's port of the two-dimensional, steady, laminar CFD code TEACH-L on a JAVA platform, in which the numerical output is linked in real time to the JSyn digital audio synthesis package. The sonification of steady, laminar, developing flow in a two dimensional duct will be described in detail.

## 1. INTRODUCTION

### 1.1. History

The non-linear partial differential equations governing the conservation of mass, momentum and energy in fluids were derived from first principles in the first half of the 19th century by J. Navier [1] and G. Stokes [2]. Except for a few very restricted cases (e.g. fully-developed laminar flow in a duct), these equations could not be solved by analytical methods, and thus remained a mathematical curiosity until numerical methods on high speed computers became available in the second half of the 20th century [3]. During the past twenty years, general-purpose CFD software has emerged as a practical tool for applying the Navier-Stokes equations to the solution of realistic fluid flow problems in engineering and physics. Today, several commercial CFD packages are routinely used by engineers and scientists in such diverse fields as hyrdrodynamics, aerodynamics, biomedical engineering, process industry, heating, ventilating and air conditioning and environmental engineering to name a few.

### 1.2. The CFD Process

A typical CFD analysis is carried out in six stages:

1. The complex, real-world situation to be analyzed is reduced to a practical CFD project based on:
   - The limitations of the CFD model being used.
   - The available time and computational resources.
   - Engineering judgment as to what details of the flow field are essential to the anlysis.
2. The **geometry** of the region of interest is either imported from a CAD package or constructed from scratch in the CFD package. Imported geometries often contain details which are extraneous to the CFD analysis and must either be modified or removed.
3. A **computational grid** is generated which must generally satisfy the following constraints:
   - The total number of grid points must not be so large as to overwhelm the limitations of CPU storage and speed.
   - The grid must be fine enough to resolve the details of the flow field which are of interest to the user.
   - The characteristics of the grid must be compatible with the solver: there can be no sudden changes in cell size and cells may not have too high an aspect ratio or be extremely skewed.
4. **Boundary conditions** must be applied to all regions of the computational domain and the **physical properties** of the fluid(s) must be specified.
5. Various solution control parameters and solver options must be set. The **solver** is then started and must be monitored until a converged solution is achieved.
6. The results are then **post-processed**, sometimes within the CFD package itself, or else exported to a data visualization package.

This cycle is often repeated several times before a final, satisfactory result is obtained.

### 1.3. CFD and Sonification

To the author's knowledge the above-described process is currently carried out entirely in the visual domain. There are no commercial CFD packages which make use of sound to enhance the interaction between the engineer and the data.

Furthermore, very little research into sonification and CFD has been published. McCabe and Rangwalla [4] presented two examples of auditory display: the simulation of an artificial heart pump and rotor-stator interaction in turbomachinery. In the first, MIDI sound was used to enhance the post-processing of the artificial heart simulation, in particular to signal global changes in the system such as the opening and closing of a heart valve. In the second, the time-varying pressure field predicted by the model was rendered directly into sound. The simulated sound was then compared with known characteristics of the actual sound, and conclusions then drawn about the validity and accuracy of the CFD model.

The potential, however, for the use of sound in almost every aspect of the CFD process, seems considerable, and could be considered at many of the stages:

1. The model **geometry** could be sonified so that glitches and discontinuities in curves and surfaces, which often occur upon transfer from a CAD package to a CFD package, could be quickly identified. In this mode, a smooth geometry would have a pleasing, harmonious sound, and discontinuities could be represented by bursts of noise or discordant pitches.
2. The **grid** could be sonified in an analogous manner, through the use of unpleasant sounds to highlight badly skewed or high aspect ratio cells.
3. Real-time sonification of the **solver** would be analogous to the frequently cited example of the auto mechanic listening to the car engine. CFD solvers work iteratively on nonlinear mathematical systems. They frequently diverge or "hunt" without reaching a converged solution. The CFD analyst is required to adjust many solution control parameters, and even choose between alternative solution stragies, in the hope that the solver engine will "behave" and provide a converged solution. Monitoring the solution process via the display alone is monotonous and unproductive; adding sound would allow the analyst to:
    - Listen to the solution in the background while pursuing other tasks.
    - Recognize favorable sound patterns which indicate good choices of solution control parameters and reflect a smoothly running computational "engine."
4. There are many aspects of adding sound to the **post-processing** of CFD data:
    - The recognition of significant patterns in the (sonified) data which are not apparent from visual displays.
    - Increased productivity in examining large amounts of data by concurrent visual and aural displays.
    - The sonic codification of global events in the flow field which are difficult to perceive from the visual diplay of local details [4].
    - The comparison of sound rendered from simulations of the pressure field with recorded sounds from the corresponding experiment [4].

### 1.4. CFD and Music

As a corollary to sonification work, there are various ways to consider CFD as a source of new musics. First, CFD provides a numerical mirror into the natural world, which has long been an inspiration to composers. For example, a classic CFD result is to predict von Karman vortex streets [5] in the wake of a circular cylinder in a cross wind. This periodic vortex shedding phenomenon sometimes gives rise to audible frequencies, as when the telephone wires "sing" in a breeze. The scope of CFD also includes the modeling of the compression waves which produce sounds, even in brass or wind instruments. So CFD could be thought of as a tool to enhance the relationship of the composer with sound phenomena in the natural world.

Second, the graphical representations of the results of CFD models are often aesthetically interesting. A common method of visual rendering is to trace the paths taken by fluid particles, see for example Fig. 1. In flowfields with obstructions and/or complex

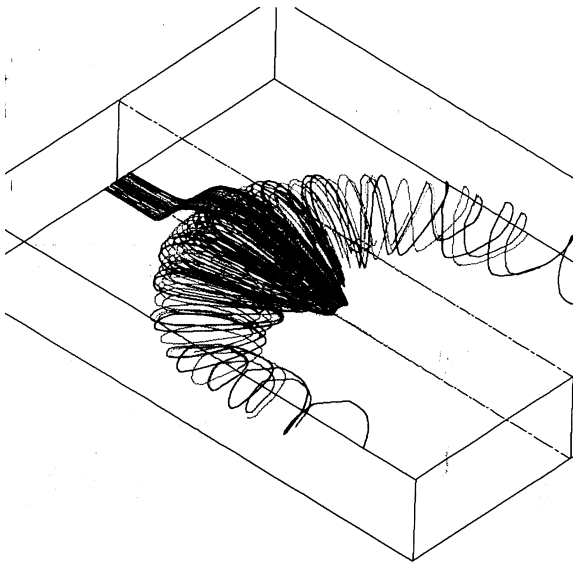

Figure 1: *Particle Paths*

geometry, especially if natural convection is present, the particles often get caught up in a complex structure of vortices, such as the horseshoe vortex shown in Fig. 1. Under the appropriate conditions, smaller and smaller vortices are spawned from their parents, until a turbulent or chaotic flow structure results. Other graphical renderings include the representation of local fluid velocities by vectors whose size, direction, and color depend on velocity magnitudes and other parameters, see Fig. 2. Both styles of visual renderings (and others not mentioned here) could be taken as a point of departure for sound exploration.

Thirdly, CFD could be used as a tool for algorithmic composition, following a tradition started by Iannis Xenakis [6] [7] who drew extensively on mathematical formulations from science and engineering. CFD is particularly attractive since the algorithms

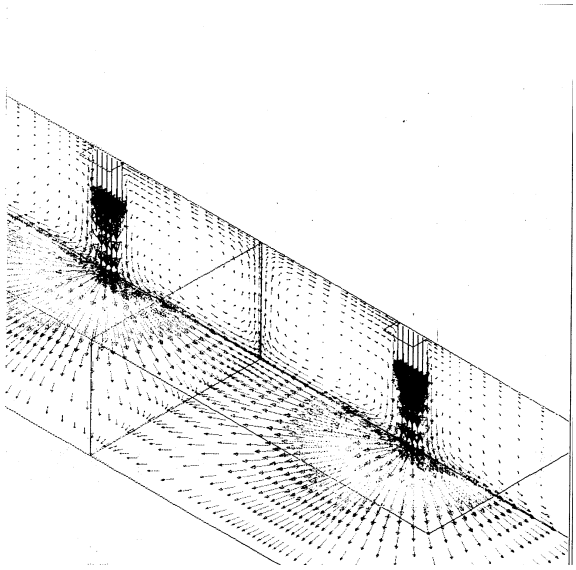

Figure 2: *Vectors*

it uses are iterative by nature, and generate numbers that evolve over time. There is therefore a potential for CFD to generate live musical compositions.

### 1.5. Scope of Present Work

This paper will focus on the real-time sonification of Stage 5 (solution) of the CFD process.

An academic CFD research code TEACH-L [8], originally written in FORTRAN, was ported by the author to Java in order to make use of JSyn [9], a digital audio synthesis package. JSyn (**J**ava **Syn**thesis) is a Java API (**a**pplication **p**rogramming **i**nterface), which provides several classes of objects that can create and modify sound. A fast DSP synthesis package written in C lies beneath JSyn's hood. All Java synthesis calls are passed transparently to the C engine.

TEACH-L solves the steady, laminar equations of the conservation of mass, momentum (Navier-Stokes) and energy on a two-dimensional, cartesian grid, using a hybrid differencing sheme and the SIMPLE (**S**emi-**I**mplicit **M**ethod for **P**ressure-**L**inked **E**quations) algorithm [10] to correct the pressure field. The algebraic equations are solved line-by-line (LBL), using the tri-diagonal matrix algorithm (TDMA). The TEACH-L code is extremely compact. There is no user-interface. To set up the geometry, grid, boundary conditions, and physical properties, the user must write her own subroutines using the templates provided. This structure was retained in the Java port, so that TEACH-L consists of a CFD API.

The compactness of TEACH-L, together with the flexibility of the JSyn API, afforded a reasonable implementation on a Macintosh G3 Powerbook, providing a tool for the in-depth exploration of various sonification strategies for a CFD solver.

In the following sections, the physical and mathematical basis of TEACH-L will be presented. One sonification strategy will be explored, followed by results and conclusions.

## 2. PHYSICAL AND MATHEMATICAL BASIS OF THE CFD SOLVER

### 2.1. The Governing Equations

In the case of steady, two-dimensional flow, the continuity (conservation of mass) equation is:

$$\frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) = 0. \tag{1}$$

where $\rho$ is the local fluid density (kg/m$^3$), $u$ and $v$ are the local fluid velocities (m/s) and $x$ and $y$ (m), correspond to the cartesian coordinate system.

For incompressible flow, the momentum equations are for the $x$ direction:

$$\rho u\frac{\partial u}{\partial x}+\rho v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x}+\frac{\partial}{\partial x}(2\mu\frac{\partial u}{\partial x})+\frac{\partial}{\partial y}(\mu[\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}]), \tag{2}$$

and for the $y$ direction:

$$\rho u\frac{\partial v}{\partial x}+\rho v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y}-\rho g+\frac{\partial}{\partial x}(\mu[\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}])+\frac{\partial}{\partial y}(2\mu\frac{\partial v}{\partial y}), \tag{3}$$

where $g$ is the acceleration due to gravity (m/s$^2$), $p$ is the fluid static pressure (Pa) and $\mu$ is the fluid dynamic viscosity (kg/ms).

The energy conservation equation for the fluid, neglecting viscous dissipation and compression heating, is:

$$\rho c_p(u\frac{\partial t}{\partial x}+v\frac{\partial t}{\partial y}) = \frac{\partial}{\partial x}(k\frac{\partial t}{\partial x})+\frac{\partial}{\partial y}(k\frac{\partial t}{\partial y}), \tag{4}$$

where $c_p$ is the fluid specific heat at constant pressure (J/kg K), $k$ is the fluid thermal conductivity (W/m K), and $t$ is the fluid static pressure (K).

### 2.2. Discretization

To solve the non-linear partial differential equations from the previous section, it is necessary to impose a grid on the flow domain of interest, see Fig. 3. In TEACH-L, discrete values of fluid velocities, properties, pressure and temperature, are stored at each grid point (the intersection of two grid lines). To obtain a matrix of algebraic equations, a control volume is constructed (shaded area in the figure) whose boundaries (shown by dashed lines) lie midway between grid points $P$ and its neighbors $N$, $S$, $E$, $W$. A complex process of formal integration of the differential equations over the control volume, followed by interpolation schemes to determine flow quantities at the control volume boundaries ($n$, $s$, $e$, $w$) in Fig. 3, finally yield a set of algebraic equations for each grid point $P$ [11]:

$$(A_P - B)\phi_P - \sum_c A_c\phi_c = C, \tag{5}$$

where the subscript $c$ on $\sum$, $A$ and $\phi$ refers to a summation over neighbor nodes $N$, $S$, $E$ and $W$, $\phi$ is a general symbol for the quantity being solved for ($u$, $v$ or $t$), $A_P$, etc. are the combined convection-diffusion coefficients (obtained from integration and interpolation), and $B$ and $C$ are, respectively, the implicit and explicit source terms (and generally represent the force(s) which drive the flow, e.g. a pressure difference).

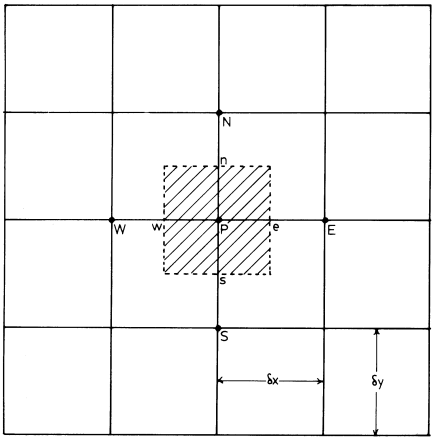


Figure 3: *Control Volume*

### 2.3. Solution

Equation (5) must be written at each node where the value of $\phi_P$ is required; doing so will generate an $N \times N$ matrix of simultaneous equations, where $N$ is the number of nodes in the solution domain. It is usually impractical to invert this matrix directly, so instead a line-by-line (LBL) scheme is used, see Fig. 4. In the LBL scheme,

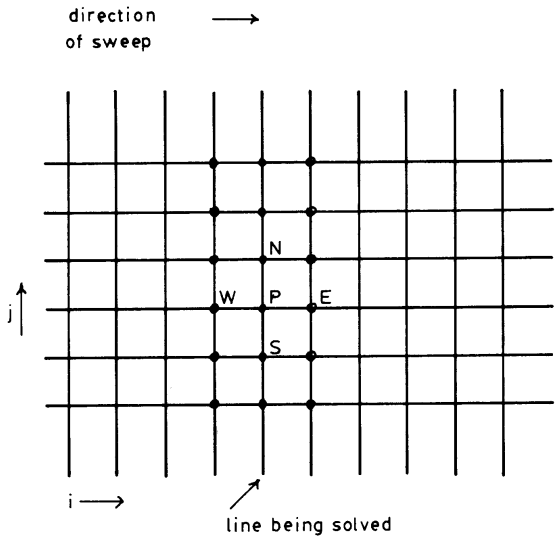


Figure 4: *Line-By-Line Scheme*

only one line at a time is solved, quantities on other lines are considered to be "known". Thus, a smaller $n \times n$ tri-diagonal matrix results, where $n$ is the number of nodes in the $j$ direction. The solver starts at the first $i$-index line in the solution domain, and "sweeps" from left to right.

Because the partial differential equations are non-linear, the resulting algebraic matrix equations will be also, that is, the convection-diffusion coefficients $A_P$, etc. will themselves be functions of the $\phi$'s. An iterative solution is thus required, as follows:

1. The coefficients for Eq.(5) in which $\phi = u$ are formed, and the current global "error" or "residual" is calculated. The matrix of coefficients is solved by LBL and new values of $\phi = u$ are obtained.
2. The same is done for $\phi = v$ and $\phi = t$.
3. A "pressure correction" equation, derived from the mass conservation equation is solved in a similar manner, the values of pressure at each note are updated [10].
4. The global errors calculated in each of the above steps are then all compared to a set of target values. If these errors fall below the target, the solution has converged and the calculation stops. Otherwise, the calculation resumes at step 1.

### 2.4. Solution Control Parameters

There are two sets of parameters which are set by the user to control the progress of the solution, the underrelaxation factors and the sweep controls. In an iterative, numerical solution, underrelaxation slows the rate of change of a variable from one iteration to the next. Underrelaxation is necessary for numerical stability and to avoid divergence of the solution. Sweep controls for each variable are set to control the number of times, per iteration, the LBL procedure is applied to the coefficient matrix. The greater the number of sweeps, the better the matrix inversion at a particular iteration (but the greater the amount of CPU time required.)

## 3. DUCT FLOW EXAMPLE

As an example of a solver sonification, the simple problem of steady, laminar, two-dimensional developing flow in a planar duct will be considered (see Fig. 5). In this flow situation, fluid enters at the left with a uniform velocity profile, which develops, as the fluid reaches the end of the duct on the right, into a parabolic profile which is characteristic of "fully-developed" flow, Eq. 6:

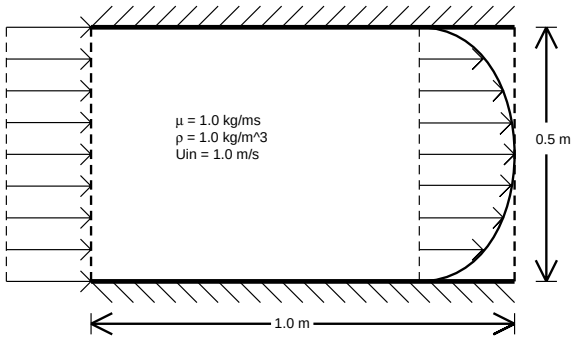


Figure 5: *Developing Flow in a 2D Duct*

$$u = \frac{3}{2} u_m (1 - \frac{4y^2}{c^2}), \tag{6}$$

where $u_m$ is the average velocity (in this case, 1 m/s), $c$ is the height (in this case 0.5 m) and $y = 0$ at the centerline of the duct.

When the flow is fully developed, the transverse velocity component $v$ vanishes, and the pressure $p$ changes only linearly with $x$:

$$\Delta p = \frac{12 u_m \mu \Delta x}{c^2} \tag{7}$$

which gives the pressure drop $\Delta p$ over some $x$-direction length $\Delta x$, and where the viscosity $\mu$ is in this case 1 kg/ms.

A coarse $7 \times 7$ evenly spaced grid was used, yielding a total of $5 \times 5 = 25$ internal or "live" cells at which the values of $u$, $v$ and $p$ are updated at each iteration by the solver. With this very simple configuration, the solver converges in about 20 iterations.

### 3.1. Sonification Strategy

The purpose of this sonification was to gain insight into the solver by listening to its progress in real time. To accomplish this, 5 sine oscillators were set up to correspond to each column of "live" grid points. As the solver (for $u$, $v$ then $p$) sweeps through the domain from left to right, first the column at $i = 1$ sounds, from $j = 1, 5$, with slight arpeggiation, and so on, with a slight pause for each column, through $i = 5$. Thus, each iteration produces 25 notes per variable, for 75 total.

In most CFD simulations, general trends and flow behavior are known to the engineer in advance of the calculation. In this sonification, the pitch strategy for each variable was selected so that the anticipated behavior in the flow direction could be "heard":

1. Development of $u$ from a flat to a parabolic profile.
2. Vanishing of $v$.
3. Linear decrease of $p$ in the flow direction.

#### *3.1.1. Pitch Mapping*

The values of $u$ range from uniformly 1.0 at the inlet (near $i = 1$) to values in the range $0.0 \leq u \leq 1.5$ at the flow exit (near $i = 5$). The initial guess for $u$ throughout the domain is 0.0. As the solution iterates, values of $u$ well in excess of 1.5 may result. The values of $u$ were mapped to pitch and scaled so that a value of $u = 1$ would yield 440 Hz, with 55 Hz set as the minimum value for values of $u < 0.125$.

The values of $v$ range from approximately 0.1 near the inlet to very small numbers near the exit (the initial guess throughout the flow field, as for $u$, is $v = 0.0$). The values of $v$ were also mapped to pitch, but scaled up so that a value of $v = 1$ would yield 8400 Hz. All nodes with values of $v <$ about $0.05$ were mapped to a major triad with pitches (150, 300, 375, 450) Hz, for nodes $j = 2, \ldots 5$.

The values of $p$ in incompressible fluid flow are generally calculated relative to some numerically convenient reference value, in this case $p_{ref} = 1.0$ Pa at node $(i, j) = (1, 1)$. Values of $p$ are calculated **relative** to $p(1, 1)$, where the significant pressure information is the $\Delta p$ between nodes which drives the flow. Using this scheme, the values of $p$ range from $-40. \leq p \leq 1.0$. The local value of $p$ was added to a different major triad (100, 200, 300, 400, 500) Hz for $j = 1, \ldots 5$. Because the values of the pressure are fairly uniform at each $i$ location, one hears, for this scheme, a succession of major triads whose pitch either increases, decreases or stays the same.

#### *3.1.2. Envelope*

The envelope (attack, sustain, decay) characteristics were derived from the matrix coefficients for each variable at each node:

$$\begin{aligned} t_1 &= A_N, \\ a_1 &= A_N / A_P, \\ t_2 &= A_S, \\ a_2 &= A_S / A_P, \\ t_3 &= A_E, \\ a_3 &= A_E / A_P, \\ t_4 &= A_W, \\ a_4 &= A_W / A_P, \\ t_5 &= 1.0, \\ a_5 &= 0.0. \end{aligned} \tag{8}$$

Equations 8 represent a 5 frame envelope where $a_1$ is the amplitude of the first frame, $t_1$ is its rise time, etc. The values for the fifth frame are set to constant values of 1.0 and 0.0 respectively to ensure that the oscillator will turn off. The values of $t_1$ were constrained to be at least 0.05 secs, to avoid clicks due to zero-length frames. Division of all neighbor coefficients $A_N$, etc. by the center coefficient $A_P$ ensures that $0.0 \leq a_n \leq 1.0$, since the center coefficient is always greater in magnitude than its corresponding neighbors.

#### *3.1.3. Duration*

In general, as the solver proceeds and makes available the latest values of variable and coefficient at each node, the mapped pitches and envelopes were queued to the oscillators. However, to make the sonic result more intelligible, some delays were added. Firstly, very slight delays were added between nodes in each column, to produce something like an arpeggiated chord. Secondly, a longer delay was added at the conclusion of each column, before proceeding to the next column. Thirdly, at the conclusion of the sonification of a variable at all 25 nodes, a longer delay was added proportional to the global error calculated for that variable. Thus, as the calculation proceeds, this delay decreases as the error is reduced. Finally, at the conclusion of a single iteration, a longer delay was added. It is thus possible for the listener to distinguish, based on the delay between notes, the different stages of the calculation.

#### *3.1.4. Timbre*

No specific timbral mapping was attempted, since a sine oscillator was used for all notes. However, because each note was given a unique envelope, some striking timbral differences resulted, mainly from different attack times.

## 4. RESULTS

The parameter/variable mapping choices described in the previous section had the following effects:

- The $u$ velocity sonification captured the oscillations of the solver well. The solver initial guess of $u = 0$ was followed, at the second iteration, by values considerably overshooting the final result. This behavior sounded like a sequence

of low, followed by high frequency chord progressions, finally settling out to a noticeably repeating sequence during the final iteration. Owing to large values of the neighbor coefficients $A_N$, etc. and the large values of center coefficient $A_P$ at most nodes relative to the neighbors, the attack times were long, and the amplitudes low. Thus the $u$ velocity sonification sounded ethereal and slowly evolving and was easily distinguished from the $v$ velocity and $p$ data.

- From the $v$ velocity sonification, it was easy to hear an initial guess of $v = 0$, via the major chord, followed, after oscillations, by non-zero (higher frequency) sounds at the inlet, and vanishing values near the flow exit. The attack times for this variable were shorter than those for the $u$ velocity, and the amplitudes higher. It was thus easy to hear the transition from the $u$ to $v$ sonification.
- The $p$ sonification was very different in timbre to those for $u$ and $v$, owing to much shorter attack times and higher amplitudes. It was thus easy to perceive the onset of $p$ data, and to hear, at each iteration, whether the pressure was increasing, staying the same, or finally, as in the converged solution, decreasing in the direction of flow.

## 5. CONCLUSIONS

The sonification of this simple duct example afforded enhanced interaction with the data in two important ways:

1. The ability to monitor the progress of convergence of the data throughout the flow field. In most CFD packages, visual monitoring of solution progress is only practical at one or two locations. The addition of sound allows the engineer to hear that, on a global basis, the solution is or isn't evolving as expected, and roughly where in the domain a problem might exist if there is one.
2. The ability to notice differences in data (the neighbor and center coefficients), via the envelope characteristics. The different timbres in the three variables aroused curiosity, and triggered further investigation of these coefficients, in order to determine if their values were correct, and to question why they were different for each variable.

In general, it is clear that the addition of sound has enormous potential for the critical examination of CFD simulations and merits further investigation.

## 6. FURTHER WORK

The current sonification was extended to add spatialization of the sound, so as to hear a sweeping progression from left to right. More complex "instruments" were constructed, to enhance the effect of local coefficients, and to add the effect of additional coefficients not currently mapped, such as source terms, and local conservation errors.

The ultimate goal should be to map every available parameter in the numerical world to some recognizable characteristic in the sound domain, in such a way that it can be distinguished and singled out for inspection and further investigation if warranted.

## 7. MUSICAL COMPOSITIONS

The duct flow example was used as an algorithmic composition tool by recording several solver runs with different parameter settings so as to change the tempi or pitch centers. Various soundfiles were created, processed, and mixed using ProTools software.

## 8. REFERENCES

[1] C.L.M.H. Navier, *Memoires Acad. R. Sci.*, Paris, Vol. 6, pp. 389-416, 1830.

[2] G.G. Stokes, *Trans. Camb. Phil. Soc.*, Cambridge, Vol. 8, pp. 287-308, 1845.

[3] J.E. Fromm and F.H. Harlow, "Numerical solutions of the problem of vortex street development," *Phys. of Fluids*, 6, 975-982, 1963.

[4] K. McCabe and A. Rangwalla, "Auditory Display of Computational Fluid Dynamics Data," in *Auditory Display*, Ed. G. Kramer, SFI Studies in the Sciences of Complexity, Proc. Vol. XVIII, Addison-Wesley, Reading, MA, USA, 1994.

[5] H. Schlichting, *Boundary Layer Theory*, McGraw Hill, New York, 1979.

[6] I. Xenakis, *Formalized Music*, Indiana University Press, Bloomington, 1971.

[7] I. Xenakis, *Arts/Sciences: Alloys*, Pendragon Press, New York, 1985.

[8] A.D. Gosman, B.E. Launder, F.C. Lockwood and G.C. Reece, Imperial College of Science and Technology, Mechanical Engineering Department, London SW7 2BX, series prepared under the auspices of the National Development Progam in Computer Assisted Learning, 1975.

[9] JSyn is a new technology developed by Phil Burk at SoftSynth (http://www.softsynth.com).

[10] D.B. Spaulding, *Int. J. Num. Meth. Engng.*, Vol. 4, pp. 551-559, 1972.

[11] S.V. Patankar, *Numerical Heat Transfer and Fluid Flow*, McGraw Hill, New York, 1980.