Effective Resonant Frequency Tracking With Inferno

Jeff Sickel [†*] *and Paul Nordine* [‡]

[†] Corpus Callosum Corporation, Evanston, IL 60202 USA.
[‡] Physical Property Measurements, Inc., Evanston, IL 60202 USA.

ABSTRACT

We describe a digital technique for tracking the resonant frequencies of piezoelectric transducers used to drive an aero-acoustic levitator. Real-time sampling of the voltage and current phase difference is used in a low-priority feedback loop to control drift and stabilization.

The implementation leverages 9p on embedded 16-bit dsPIC33F digital signal controllers. Data collection and processing are performed with various 9p clients. This paper describes the use of 9p and Inferno to track, adjust, and optimize output frequency and sound pressure levels on the instrument.

1. Introduction

The process of developing a new aero-acoustic levitator (AAL)[1] provided an opportunity to solve a persistent problem in prior versions: the six independent piezoelectric transducers, used to create standing waves that provide levitation, position, and spin control, would drift out of resonance. The transducers have well-matched resonant frequencies at ambient temperature, within ± 10Hz in 22,000 Hz. But as they warm up when operated, the resonant frequencies change at rates dependent on the transducer Q-factors, the desired output sound pressure level (SPL), and differences in the piezoelectric material properties. The range of resonant frequencies may increase such that it becomes impossible to maintain equal and constant SPL values from all six transducers. As a transducer goes off resonance at constant voltage operation, the current decreases and the current-voltage phase difference changes. The SPL changes proportionally to the current-voltage phase off of resonance. A constant SPL is maintained using a feedback loop to monitor the resonant and operating frequency difference and adjusting the output voltage to the acoustic amplifier.

A previous generation of the AAL used a piezoelectric film pickup attached to a region on the reverse side of the transducer horn. The pickup provided reference signals for the output frequency and amplitude of the horn. The sensor output was highly sensitive to placement. Correctly placing and securing the film proved problematic, to the point that driving the transducer to high SPL values required for solid-liquid phase transition stabilization tended to separate the film from the horn.

* Author for correspondence (jas@corpus-callosum.com)

Prior runtime protocols required user monitoring of the transducer output levels and manually selecting one transducer as the 'master' for tracking the frequency during an experiment. The logic was to pick the transducer that would sweep up or past the resonant frequencies of the other transducers as its own resonant frequency dropped as it heated up. This action would work for short experiments but ultimately limited the duration of levitation experiments due to SPL and frequency drift as the temperatures of each transducer fluctuated.

The new system removes the user selection criteria by monitoring the voltage, current, and voltage-current phase difference of each transducer. The resonant frequencies are determined from the V-I phase differences and the system is operated at their average value. The SPL of each transducer attained by calculating the amplifier output power is adjusted to match a desired set point for the amplifier output voltage. The range of resonant frequencies is controlled as the transducers warm up by simply turning the cooling fans on or off if the amplifier frequency is greater or less than the resonant values.

The new technique is implemented by sampling the output voltage and current on an embedded system, tracking phase differences between the current and voltage, then controlling the cooling fans and adjusting output gain and setting the operating frequency to the average resonant frequency. A remote program running in Inferno has proven significantly simpler to model and implement, as the profiling data on each transducer is stored in a human readable file and can be easily updated as necessary. The enhanced adjustment and tuning capabilities provide optimal stabilization and simple setup for tuning experiments.

Inferno was chosen as a front end for several reasons, most important being portability between platforms and the support for 9p (Styx)[2]. The programming language Limbo[3] has proven a rapid tool for implementing floating point algorithms[4], and concurrently scheduling measurement processes for running transducer profiling tests.

1.1. Resonance

Each acoustic transducer is composed of solid aluminum alloy with a piezoelectric transducer as the driver designed for a resonant frequency between 22.12 and 22.25 kHz[1]. Forced convection cooling is controlled by a Limbo program (a change from prior versions; see implementation). The manufacturing process introduces certain variations in the overall mass and characteristic profile of a transducer, resulting in a varying Q-factors. The subtle differences in physical characteristics cause each transducer to drift off resonant frequency differently as their temperature fluctuates over different output voltages.

After assembly, each transducer is profiled by measuring sound pressure levels with a calibrated microphone at selected driving voltages and tracking change over time. The data are used to determine the Q of each transducer, the functional relationships between SPL and amplifier power, and the V-I phase as it relates to the resonant-operating frequency difference. Closely matching transducers are paired on an axis. Three sets of transducers are then installed into the AAL on orthogonal axes, where further calibration testing is required. The profile data are stored in a human-readable file and used in the control program to track and measure SPL as transducers go on and off resonance. The use of separate files allows the user to quickly change configurations, as transducers are rearranged or changed out during the calibration phase of the instrument.

1.2. Voltage-Current Phase Tracking

The voltage-current phase difference is used to determine how a transducer is being driven in comparison to its resonant frequency. When the transducer is off resonance, the output power of the constant-voltage amplifier decreases and an increased voltage is required to attain the desired SPL.

The controller boards paired with each acoustic transducer drive the output through an onboard DAC (Maxim MAX5353) and DDS (Analog Devices AD9834). The dsPIC33F ADC channels are used to measure the output voltage and current. A Xilinx XC9536XL CPLD calculates return voltage and current phases and provides interrupts to a gated timer on the dsPIC33F. This timing data provides the leading or lagging phases of the current to the voltage and is used as feedback into the control system.

2. Firmware Implementation

The main communication board uses the dsPIC33F serial UART module to present a device command mode and direct 9p layering for interaction with Plan 9 and Inferno development machines. 9p on the dsPIC33F was accomplished by porting *lib9* and *lib9p* source from *Plan 9 from User Space* [5] to support the 16-bit architecture of the dsPIC33F.

The communication board is configured to start the serial UART in a raw command mode and will automatically switch to 9p by sniffing for a Tversion message[6] in the serial input stream. On receipt of the Tversion header, the UART driver switches to a DMA mode to reduce the UART receive interrupts on the dsPIC33F from one per byte to two DMA interrupts: a four-byte size field, and the remaining message length. This encapsulation of 9p simplified the code required for error checking and recovery.

2.1. Application Programmers Interface

The cluster of embedded controllers is accessed through a single serial interface. An IOLAN device server is used to extend the 9p connection[1] over the network. Though the cluster can be accessed directly without using the 9p protocol, the majority of the functionality has been implemented by leveraging the file system provided over 9p. The AAL communications controller provides a single-level file system that is accessed through a shell or program. The file system and underlying single level structure is portrayed in Figure 1.

The *ctl, data, stats, status* files are used for direct access to the main communications board. The *t** files align to each transducer control board. As is typical in Plan 9, the *ctl* file is used to send commands to the board. Reading the *stats* file dumps human-readable data for all the online boards as seen in the prior example. The *data* file provides an array of bytes packed for each transducer row in the *stats* file. The *status* provides information on the current state of the main communications board. End user programs can read the current state of the system by opening and reading the *stats* file. Alternatively, individual board data can be read through the *tN* files, providing single channel monitoring and control. A read of the *tNerr* file returns and error summary tracked by the main control board for each card.

[1] An IOLAN device server is configured to only allow one connection at a time.

```
% mount -A tcp!iolan!aal9p /n/aal
% lc /n/aal
ctl    status t1err  t2err  t3err  t4err  t5err  t6err
data   t1     t2     t3     t4     t5     t6
stats  t1ctl  t2ctl  t3ctl  t4ctl  t5ctl  t6ctl

% cat /n/aal/stats
ID Ai Fi Phi Mper Mfreq Flags Vo PhiV Io PhiI F
t1 2000 221677 3756 0 0.00 65 2221 0 1654 3151 0
t2 2029 221677 2048 0 0.00 193 1982 0 1640 3275 0
t3 2400 221677 0 0 0.00 65 2439 0 1538 3212 1
t4 1655 221677 2048 0 0.00 193 2636 2938 1294 0
t5 2031 221677 0 0 0.00 65 2044 0 1691 3075 0
t6 1866 221677 2048 0 0.00 65 2372 0 1724 3295 1
```

Figure 1 Inferno mount command and AAL file system access.

In practice, a scheduled read of the *data* or *stats* files is performed to update the in memory representation of the AAL data. That data are used for any processing until the period ends and a subsequent read has been performed.

2.1.1. Ctl commands

The main communication board and six transducer controller boards can be controlled with a simple set of commands. Writing a string to the *ctl* file either produces a global change or a change along an axial pair of boards as described in Table 1.

`fb`	*N*	Enables ’1’ or disables ’0’ position feedback processing on all transducer control boards.
`freq`	*N*	Globally set frequency to *N* dHz.
`gain`	*N*	Globally set the DAC gain to *N*. Linear from min 3584 to max 512.
`reset`		Reset transducers to default output values.
`spin`	*str str N*	Change spin of *X*, *Y*, or *Z* axis ’up’ or ’down’ *N* (DDS 0–4096 → 0–360 degrees.)

Table 1 Communication board *ctl* commands.

2.1.2. Transducer ctl commands

Changes to independent boards are handled through the *t?ctl* files as depicted in Table 2. The specified transducer commands are used to tune the output during a running experiment. They can also be used for custom applications that test a particular transducer.

`debug`	*N*	Enables '1' or disables '0' debugging output over external UART.
`fan`	*N*	Turn transducer cooling fan on '1' or off '0'.
`fb`	*N*	Enables '1' or disables '0' position feedback processing.
`freq`	*N*	Set frequency to *N* dHz (use main *ctl* instead).
`gain`	*N*	Set channel output DAC gain to *N*.
`mod`	*N n*	Amplitude modulate *N* (±100)% at *n* Hz.
`phase`	*N*	Set DDS phase to *N* (DDS 0–4096 → 0–360) degrees.
`reset`		Reset to startup values.

Table 2 Transducer *ctl* commands

3. Modeling

The file system representation of our AAL controller boards allowed us to create simple shell scripts or programs to provide quick data collection and analysis. For example, the use of an *rc* script to scan the transducers and collect data for testing further aspects of our control loop can be seen in Figure 2.

```
log = $home/tests/gainfreqsweep.log
echo Start `{date} >> $log
echo reset > ctl

for(g in `{seq 3000 -20 2400}) {
  echo gain $g > ctl
  for(f in `{seq 221500 10 222500}) {
    echo freq $f > ctl
    echo gain: $g   freq: $f
    cat stats >> $log
  }
}

echo reset > ctl
echo Completed `{date} >> $log
```

Figure 2 Example of a gain and frequency sweep.

The transducer controller boards do not sample the output from the amplifiers directly. We manually verify the linear output for voltage and current and use those data to construct a calibrated profile set of constants used in subsequent equations. The calibration procedure is also performed with a simple Rc function and manual entry as in Figure 3.

```
log = $home/tests/run.log
fn s {
    ts = ‘{date -n}
    ID=‘{echo t$1}
    delta=‘{echo $2}
    PkV=‘{echo $3}
    PkI=‘{echo $4}
    measure=‘{cat $ID}
    Ai=‘{echo $measure | awk '{print $2}'}
    Fi=‘{echo $measure | awk '{print $3}'}
    Vo=‘{echo $measure | awk '{print $8}'}
    PhiV=‘{echo $measure | awk '{print $9}'}
    Io=‘{echo $measure | awk '{print $10}'}
    PhiI=‘{echo $measure | awk '{print $11}'}
    echo $ts $ID $Ai $Fi $Vo $PhiV $Io $PhiI $delta $PkV $PkI >> $log
}

cd /n/aal
echo freq 221900 > ctl
echo gain 2920 > t6ctl

# record transducer 6 measurements
# delta in µs
# output voltage and current recorded in volts from a scope
s 6 7.2 2.98 .184
```

Figure 3 Rc data collection function and commands.

3.1. Equations

The data read from the *data* and *stats* files is in integer format. All floating point conversion is done in Limbo. A table of calibrated profile data is used with the measured data from our calculations. The following equations use measured and set values from Table 3.

V_o	ADC 12-bit measured voltage output to transducer.
I_o	ADC 12-bit measured current output to transducer.
F	Integer representation of frequency in dHz.
V_g	12-bit value written to the DAC.
ϕ_V	CPLD calculated 12-bit voltage phase.
ϕ_I	CPLD calculated 12-bit current phase.
$Fosc$	dsPIC33F system clock, 7.378×10^7

Table 3 Measurable input and output from transducer controller.

Calibration constants from our profile are: A_V, A_I, B_I, D_I, A_ϕ, B_ϕ, A_{SPL}, A_G, B_G.

The first step is to calculate the amplifier output voltage, described by

$$V = A_V \cdot V_{o,} \tag{1}$$

where V is the calculated peak-to-peak voltage supplied to the transducer. The measured output voltage V_o is used with a calibration constant A_V defined through prior probed experiments.

The calculated current delivered to the transducer is

$$I = A_I I_o^2 + B_I \cdot I_o + D_I \tag{2}$$

with I the calculated current, I_o is measured output current, and the profile calibration constants are A_I, B_I, and D_I. If a transducer is swapped out, or we need to change additional hardware, we can easily calculate new profile constants as required to fit the linear range used when running the system.

The measured current-voltage phase difference in amplifier output is determined through

$$\phi_C = Sign(\phi_I - \phi_V) \cdot \left[1 - \frac{(\phi_I + \phi_V)}{Fosc \cdot 2} \cdot \omega\right] \cdot 180 \tag{3}$$

The frequency, ω, is reported in kHz with resolution determined by the DDS in decihertz. The result from equation 3 will be corrected in equation 4 to accommodate for a phase shift induced in our electronics. We can use ϕ_C without adjustments to the data collected from each transducer in order to depict differing resonant frequencies at a fixed output gain as seen in Figure 4.

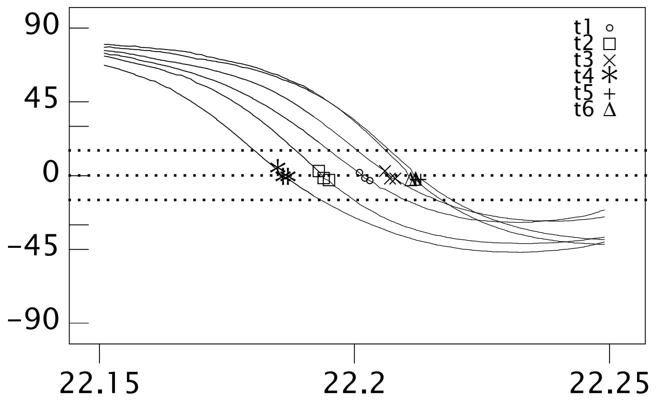

Figure 4 Plot of measured voltage-current phase difference. The frequency sweep was performed at a fixed gain, showing the resonant frequency variation across all six transducers.

To accomodate for a phase shift introduced in our electronics, a true V-I phase difference correction is performed,

$$\phi = \phi_C + B_\phi \tag{4}$$

Amplifier power, P, in units based on peak-to-peak V and I measurements is defined as

$$P = \frac{V \cdot I}{100} \cdot cos(\phi) \tag{5}$$

The division by 100 is the result of our analysis using a V measurement with a 10:1 probe and reporting I in ma.

The power required to match a specified SPL, measured in volts from a Bruel & Kjær pressure–field microphone located 7.5cm from the transducer surface, is

$$SPL = A_{SPL} \cdot Sqrt(P_s), \tag{6}$$

$$P_s = \left(\frac{SPL}{A_{SPL}}\right)^2 \tag{6.1}$$

The new gain values are calculated by

$$V_C = A_G \cdot G + B_{G}, \tag{7}$$

where the G value is a gain parameter supplied by the end user or the software during the feedback loop.

The new voltage for a specified SPL are

$$V_n = V \cdot Sqrt\left(\frac{P}{P_s}\right)\ ,\ V_{nC} = V_C \cdot Sqrt\left(\frac{P}{P_s}\right) \tag{8}$$

and then fed back to calculate a new gain value to write to the DAC. The process is repeated at a defined interval in a thread running in our Inferno application.

The prior equations are implemented in Limbo and reduce into the following Limbo function:

```
splgain(spl: real, tp: ref Tprofile, ts: ref Tstats): real
{
    ng := 0.0;
    V := math->fabs(ampvolt(tp, ts));
    P1 := math->fabs(amppower(tp, ts));
    if(P1 != 0.0) {
        P2 := math->fabs(splpower(spl, tp));
        V2 := V * math->sqrt(P2/P1);
        Vc2 := V2/tp.Av;
        ng = (-tp.Bg + Vc2)/tp.Ag;
    }
    return ng;
}
```

Figure 5 Limbo function used to attain a linear 12-bit gain value. The change is performed by writing the new gain to the transducer controller *ctl* file.

The resonant frequency of a transducer is calculated using constants A_3 to A_0 in a cubic equation relating frequency to V-I phase difference,

$$\omega_r - \omega = A_3\phi^3 + A_2\phi^2 + A_1\phi + A_{0}, \tag{9}$$

where ω_r is the resonant frequency and ω is the operating frequency.

4. Inferno control system

The prior algorithms are easily implemented in a Limbo module to handle all calibration routines required for the running system. In order to optimize the system, we use a timer to periodically poll the */n/aal/data* file of the AAL and use the data to update the operating frequency and match pre–amp gain required to achieve the desired SPL values for each transducer. The timer callback uses the calculated average of resonant frequencies determined through the result of equation 9.

By measuring the V-I phase difference, the tracking algorithm is used to tune the system under varying loads. The cooling fans for each transducer are turned on or off if the operating frequency is greater or less than the resonant frequency. This automated control brings the transducers into a ±10 Hz range from the operating frequency. The result of this control is that the resonant frequencies converge towards the operating frequency as the transducers warm up, as illustrated in Figure 6.

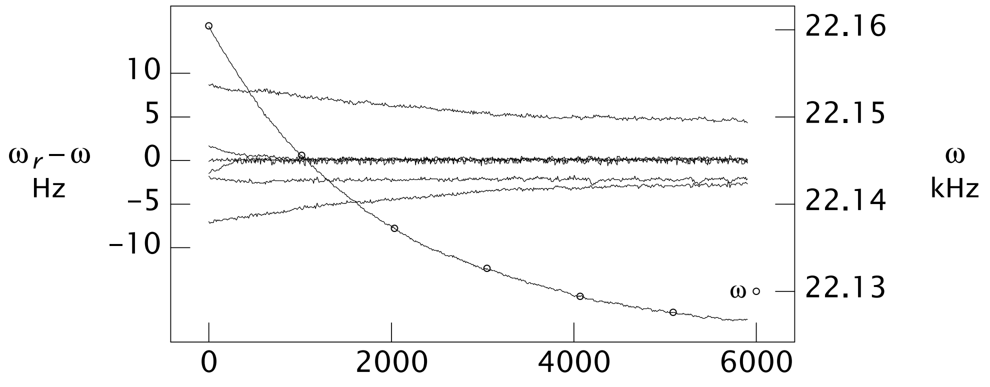

Figure 6 Frequency tracking 0.9 SPL over time in seconds. The left axis depicts the ability to keep transducer resonant frequency ω_r, within 10 Hz of the operating frequency ω. The right axis presents the drop of the operating frequency.

The control system allows for fast startup times where we can quickly bring all six transducers into a working state from a cold start. Evidence of this is detailed in Figure 7 where we plot a period where the tracking system is not in operation.

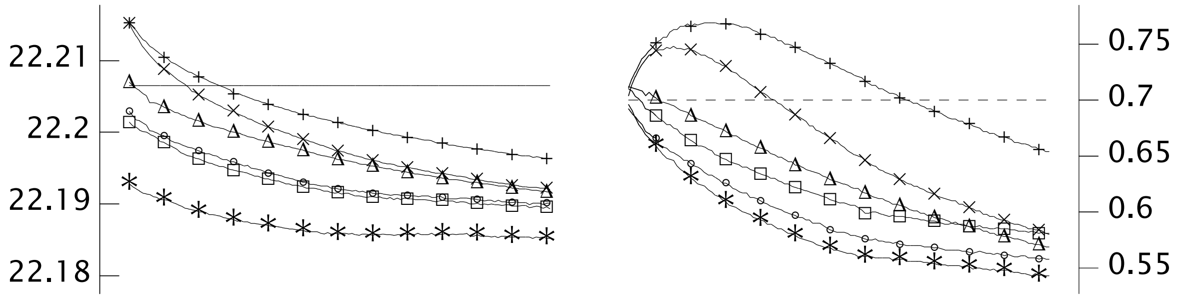

Figure 7 Frequency and SPL tracking off. The left and right sides cover the same time period of twenty minutes with the frequency a constant 22.2065 kHz and the SPL set to 0.7. On the left, the six transducers resonant frequencies drift lower as they heat up over time. On the right, the actual SPL for each transducer varies as the voltage-current phase difference changes.

Once the frequency and SPL tracking system is enabled, the transducers are brought into a manageable range as we can see in Figure 8. Though there is automated control of the system, an operator can easily change parameters that effect SPL, sample position, spin while monitoring the Inferno console and performing visual inspection of the gas flow rate, temperature, and position of the sample.

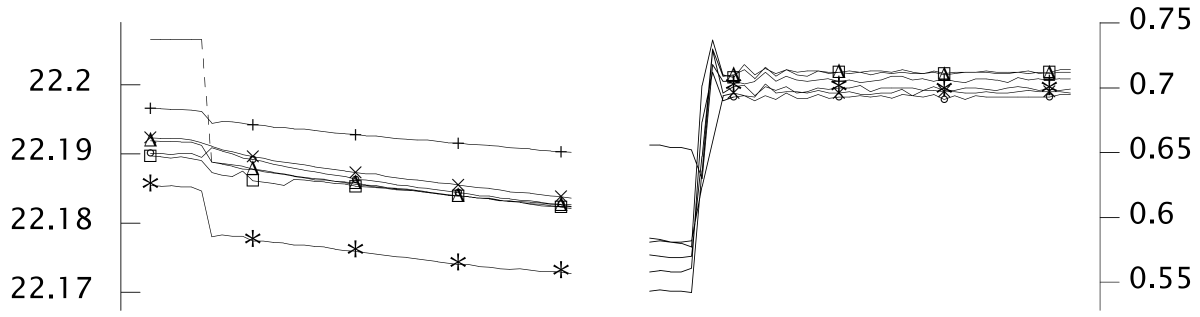

Figure 8 Frequency and SPL tracking turned on and run for approximately 10 minutes. The left shows the average frequency dropping from 22.1888 to 22.1823 kHz. On the right, the SPL stabilized at the target output of 0.7.

4.1. Console

Most of the tasks used in controlling the system are performed within the user console. By default these are simple functions spawned off after mounting the AAL file system. The console provides the operator with simple management tools to set the SPL, operating frequency, phase relationships for position and spin control, and amplitude modulation. It also provides a simple graphically driven mechanism to turn SPL and frequency tracking or off and enabling position feed-back processing used to stabilize a sample.

The graphics applications that define the console are currently made up of four separate screens: the primary console, a graphical scope, a statistics screen, and a position sensor screen. These screens are used to manage the system as well as provide logging data of the statistics collected during the runtime.

5. Conclusion

The use of a 9p based file system has facilitated the creation of a working interface to the AAL that allows easy user manipulation without requiring additional physical controls. By providing a single level file system with easy to understand commands we have been able to provide a structure for quickly bringing the system up and monitoring its state. The ease of using Limbo programs or other shell environments to access the system has improved our ability to conduct experiments and gather data required for further analysis that have historically been more difficult to accomplish.

6. Acknowledgements

We would like to thank Steve Finkleman, Eduardo Lopez Gil, Dennis Merkley, and James Rix for their efforts in designing and developing the AAL.

7. References

[1] J.K. R. Weber, D. S. Hampton, D. R. Merkley, C. A. Rey, M. M. Zatarski, and P. C. Nordine. Aero-acoustic levitation: A method for containerless liquid-phase processing at high temperatures. *Review of Scientific Instruments*, 65(2):456–465, February 1994.

[2] R. Pike, and D. M. Ritchie, The Styx Architecture for Distributed Systems. *Bell Labs Technical Journal*, 5(2), April–June 1999.

[3] D. M. Ritchie, "The Limbo Programming Language", in *Inferno Programmer's Manual, Volume Two*, Vita Nuova Holdings Ltd., York, 2000.

[4] E. Grosse. Real Inferno. In *Proceedings of the IFIP TC2/WG2.5 working conference on Quality of numerical software*, p.270–279, January, 1997, Oxford, United Kingdom.

[5] R. Cox. *Plan 9 from User Space*. http://swtch.com/plan9port/

[6] http://plan9.bell-labs.com/magic/man2html/5/0intro