Denver, Colorado
NOISE-CON 2013
2013 August 26-28

Auralization Architectures for NASA's Next Generation Aircraft Noise Prediction Program

Stephen A. Rizzi
Leonard V. Lopes
Casey L. Burley
NASA Langley Research Center
2 N. Dryden St.
Hampton
VA 23681
stephen.a.rizzi@nasa.gov

Aric R. Aumann
Analytical Services and Materials, Inc.
2 N. Dryden St.
Hampton
VA 23681

ABSTRACT

Aircraft community noise is a significant concern due to continued growth in air traffic, increasingly stringent environmental goals, and operational limitations imposed by airport authorities. The assessment of human response to noise from future aircraft can only be afforded through laboratory testing using simulated flyover noise. Recent work by the authors demonstrated the ability to auralize predicted flyover noise for a state-of-the-art reference aircraft and a future hybrid wing body aircraft concept. This auralization used source noise predictions from NASA's Aircraft NOise Prediction Program (ANOPP) as input. The results from this process demonstrated that auralization based upon system noise predictions is consistent with, and complementary to, system noise predictions alone. To further develop and validate the auralization process, improvements to the interfaces between the synthesis capability and the system noise tools are required. This paper describes the key elements required for accurate noise synthesis and introduces auralization architectures for use with the next-generation ANOPP (ANOPP2). The architectures are built around a new auralization library and its associated Application Programming Interface (API) that utilize ANOPP2 APIs to access data required for auralization. The architectures are designed to make the process of auralizing flyover noise a common element of system noise prediction.

1. INTRODUCTION

Auralization of aircraft flyover noise is the process by which source noise predictions, often performed in the frequency domain, are turned into audible sound at some distant observer location(s). System noise prediction programs, such as ANOPP, are oriented toward generation of noise metrics and generally lack a built-in auralization capability of their own. However, by combining synthesis and simulation tools with source noise predictions from ANOPP, auralizations of a reference state-of-the-art tube-and-wing aircraft and a conceptual hybrid wing body (HWB) aircraft were recently made possible.[1] As a validated tool, auralization can be used to more effectively communicate the societal benefit of low noise concepts to stakeholders than can tabulated metrics alone. Further, auralization provides a feedback mechanism to the technologists developing noise reduction concepts. With this capability, it is now possible to assess human response to flyover noise by systematically evaluating source noise reductions within the context of a system level simulation.

The main elements of auralization include synthesis, propagation, and playback. One possible implementation of these elements is shown in Figure 1.[2] In the synthesis element, source noise predictions, e.g., from ANOPP, are performed *a priori*, and serve with other data as input. The synthesis block generates a continuously evolving source pressure time history at the emission angles determined by the propagation path (straight or curved). In the propagation element, the pressure time history output by the synthesis block is propagated to a ground observer through application of a time-varying time delay, gain and filter. These operations may be performed on the entire record as shown, or buffer by buffer as part of the playback element. Simulation of a flyover in the context of an interactive virtual environment is made possible through the use of the Community Noise Test Environment (CNoTE)[3] and a specialized real-time audio processor.[4] In such an environment, listener tracking data is passed to the real-time processor and multi-channel audio is passed back to the listener.

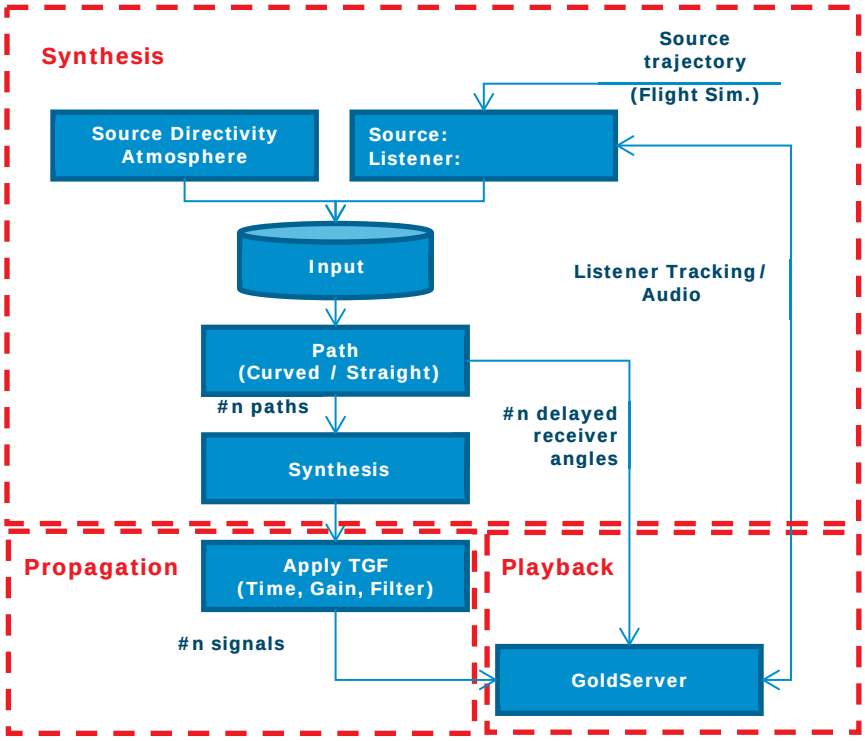

Figure 1: An auralization framework using system noise predictions as input.[2]

While the recent study[1] demonstrated the feasibility and benefits of a full aircraft auralization using such a framework, it also identified the need for a more seamless connection between the prediction and auralization tools. In particular, one-way data exchange from the ANOPP source noise prediction to the auralization processes was performed manually using Network Common Data Form (NetCDF)[5] files. This cumbersome data exchange, in combination with bookkeeping necessary to track applied noise control treatments, manual preprocessing for source noise de-dopplerization, and duplicative input to the source noise prediction and the auralization elements, made the process both labor intensive and error-prone.

In recognition of these limitations, it was decided to develop a more integrated approach to allow auralization of flyover noise to be a common element of system noise prediction. A more integrated approach would enable two-way data exchange, whereby the noise prediction process would provide its output to the auralization process, and the auralization process would provide its output back to the noise prediction program for additional processing or reporting. The new

approach is facilitated by two developments: the next generation aircraft noise prediction program (ANOPP2) software library, and a new auralization software library. This paper describes both of these libraries and their associated APIs at a high level, and offers three architectures for more closely integrating system noise prediction and auralization. In particular, the approach taken allows the customization of user written code while retaining more easily maintained and extensible software libraries that provide core functionality.

2. NEXT-GENERATION AIRCRAFT NOISE PREDICTION PROGRAM

NASA initiated the development of ANOPP[6,7] approximately 30 years ago and has continued that development to provide the U.S. Government with the ability to assess aircraft noise. The prediction methodologies that have been implemented within ANOPP are predominantly based on empirical, semi-empirical, and analytical models, validated with the best available experimental data. Many of the prediction methods work well for conventional aircraft configurations but lack capability and fidelity required for unconventional configurations. The recent push for unconventional aircraft designs requires more robust, higher-fidelity, physics-based noise prediction tools that can predict outside of the experience base for which ANOPP methods were developed.

The next generation ANOPP, called ANOPP2, is designed to address the limitations of ANOPP in order to accurately predict noise for current and future concepts. Figure 2a shows a conventional tube-and-wing aircraft configuration with rear-mounted engines. Noise sources from this configuration typically include the engines and airframe components such as the landing gear, flaps, and slats. Prediction of noise from these sources in isolation is not the same as when installed. For example, the noise from an installed main landing gear is directly related to the local flow field, which is affected by the freestream flow velocity and the aircraft's high-lift system settings.[8] Similarly, engine noise can be reflected from the wing for under-the-wing configurations, or partially shielded from the ground for over-the-wing configurations. These effects, as well as other source noise phenomena, are more evident for unconventional configurations as shown in Figure 2b. This HWB design offers significant noise reduction potential due to shielding of the engine sources by the airframe. Using ANOPP to predict the noise for this configuration requires novel application of the methods since the HWB configuration is beyond the database for which the methods were built.

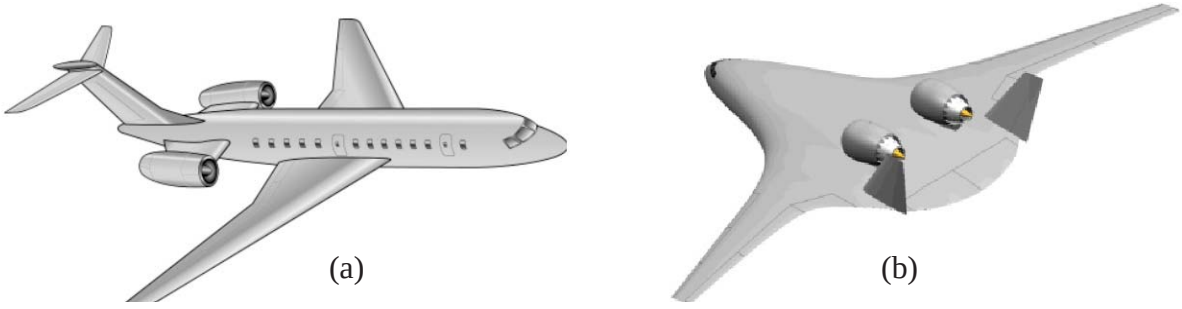

Figure 2: Conventional (a) and unconventional (b) aircraft designs.

An important advantage of ANOPP2 is that it provides a framework to integrate acoustic modeling approaches of varied fidelity for source noise component prediction, installation effects, and propagation to the far-field. This multi-fidelity framework is accomplished via a plugin system, which allows ANOPP2 to include fast prediction methods for design optimization while also including physics based prediction methods that contain the fidelity required for

understanding and controlling noise. ANOPP2 hence offers the user several options for predicting specific noise depending on requested fidelity and computational resources.

The ANOPP2 framework is built around a software library, called the ANOPP2 library, which consists of a number of sub-libraries for performing specific functions. The most relevant sub-libraries, vis-à-vis auralization, are Flight Path, Atmosphere, Observer, and Acoustic Analysis. Each sub-library within the ANOPP2 library contains functions for the generation, retrieval and storage of the specific data needed by that library using documented APIs. For example, the Observer library contains functionality needed to define, store, and retrieve the locations of multiple observers so predictions can be propagated to observer locations. These APIs are the interface to the ANOPP2 functionality that is used by a user-written FORTRAN or C++ code. The use of ANOPP2 APIs in this manner allows the user to customize code for a specific application as opposed to relying on a single general-purpose code.

3. AURALIZATION LIBRARY

Central to the auralization architectures to be discussed in Section 4 is a new Auralization software library and its associated API. The Auralization library is an object oriented library with functions and data structures for path, synthesis, and propagation processing. The interface to the Auralization library is through an Auralization API, which provides functionality similar to that represented by the blocks in Figure 1, but is seamlessly connected with external code and data. The primary interface with an external noise prediction framework is ANOPP2 via the ANOPP2 APIs, but additional interfaces may be added in the future.

Functionality needed to provide data on the propagation path from the noise source to the observer at each emission time increment is implemented using a *Path* object. This data includes the emission angle used to synthesize the source noise, the ray path used to propagate that noise to the observer, and the receiver angle used to optionally locate the source in a virtual three-dimensional listening environment. Note that the data is determined for each ray path between the source(s) and observer(s). The main elements of the *Path* object needed to implement this functionality are notionally listed in Table 1. Here, public functions refer to those functions accessible through the Auralization API, methods refer to the specific means of performing the specified function, and data refers to those quantities operated on by the specified function.

Table 1: Main elements of *Path* object in the Auralization API.

Public Functions	Methods	Data
Fetch_Source_Position	Flight Path API	Location and orientation
	Read external file	
Fetch_Observer_Position	Observer API	Location
	Read external file	
Calc_Path	Straight (built-in)	Ray(s), emission angle(s), receiver angle(s)
	Curved[2,9]	
	Curved GPU[10]	
Calc_TGF	Atmosphere API	Time delay(s), gain(s), filter(s)
	Ground plane (TBD)	

Likewise, a *Synthesis* object is used to provide the functionality to produce a buffer of pressure time history data for each source-receiver path at the instantaneous emission angle based upon provided source noise predictions, e.g., from ANOPP2. The path emission angle and source directivity are fetched buffer by buffer to generate a smoothly evolving sound.[3,11-13] The main elements of the *Synthesis* object are notionally provided in Table 2. After synthesis, the

buffer is propagated to the receiver using the *Propagation* object, described below. Propagation of the synthesized buffer may occur individually for each source-receiver path, or collectively for collocated sources sharing a common path to the receiver. For the latter, a mixing function is required.

Table 2: Main elements of *Synthesis* object in the Auralization API.

Public Functions	Methods	Data
Fetch_Source_Directivity	Observer API	Hemisphere: broadband, narrowband, pure tone, time domain
	Read NetCDF file	
Fetch_Emission_Angle	From path object	Emission angle
	Read external file	
Synthesize_Buffer	Broadband[3,11]	$p_s(t)$ – One for each source component and path
	Narrowband[3,11]	
	Pure tone[12]	
	Time domain[13]	
Mix_Buffers	Built-in	$p_s(t)$ – One for each path
Write_Buffers	Observer API	
	Write external file	

Lastly, the *Propagation* object accumulates the sample buffers output from the *Synthesis* object, and applies a time-dependent fractional time delay (T), gain (G) and filter (F) for each path. The time delay accounts for the propagation time, the gain accounts for spherical spreading loss, and the filter accounts for atmospheric absorption. Note that all operations are performed in the time domain. The end product of the TGF processing is a pseudo-recording of the pressure time history at the observer location(s), which may be used to generate aircraft noise metrics with the ANOPP2 Acoustic Analysis API. The pseudo-recording can be played back on any suitable sound reproduction system or be incorporated in an interactive virtual three-dimensional listening environment, through use of the CNoTE[3] software and specialized real-time hardware.[4] The main elements of the propagation object are shown in Table 3.

Table 3: Main elements of *Propagation* object in the Auralization API.

Public Functions	Methods	Data
Fetch_Synth_Buffer	From synth object	$p_s(t)$ – At least one for each path (buffer)
	Read external file	
Fetch_TGF	From path object	Time delay(s), gain(s), filter(s)
	Read external file	
Apply_Delay	Built-in	
Apply_Gain	Built-in	
Apply_Filter	Built-in	
Write_Output_Buffer	Observer API	
	Wave file – CNoTE	
	Sound card	

Finally, note that the all objects in the Auralization API are meant to be extensible to allow user-defined methods to be specified in the future.

4. ARCHITECTURE OPTIONS

Three architecture options are next presented which allow auralization to be performed using the ANOPP2 and Auralization APIs. These are the inline, restart, and plugin options.

A. Inline Option

The inline option performs both the ANOPP2 and auralization analyses within a single user-written executable code. A block diagram of the inline option is shown in Figure 3. Here, the user provides the definition of the aircraft and its mission to the ANOPP2 library functions and internal prediction methods. The resulting noise predictions are then provided to the Auralization API functions which subsequently generates pressure time histories that are then stored on the ANOPP2 observer object. Through the use of ANOPP2 library functions, this information is available to the user. The inline option may have its most utility in a design environment where the user wishes to auralize different variants of the same underlying analyses, e.g., a change in flight operations.

B. Restart Option

The restart option is similar to the inline option except that noise prediction output from ANOPP2 is written to a restart file at the end of the ANOPP2 analysis. The data from that file is then loaded via ANOPP2 library functions, at the start of the auralization, as shown in Figure 4. This architecture is likely to be most useful when the user wishes to change the path or synthesis parameters but otherwise retain the source definition. The auralization portion of the restart option essentially reduces to a standalone auralization code when the source directivity data is written to NetCDF formatted files instead of to an ANOPP2 restart file. This file format is commonly used for storing source directivity data, whether predicted or measured. In this case, the NetCDF library functions are used to load the data instead of the ANOPP2 library.

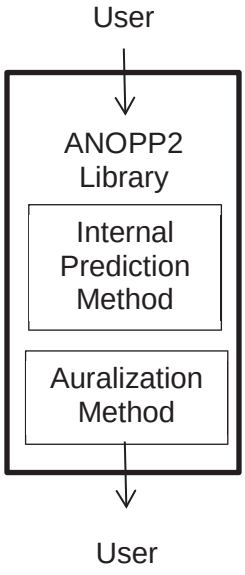

Figure 3: Diagram of the inline architecture.

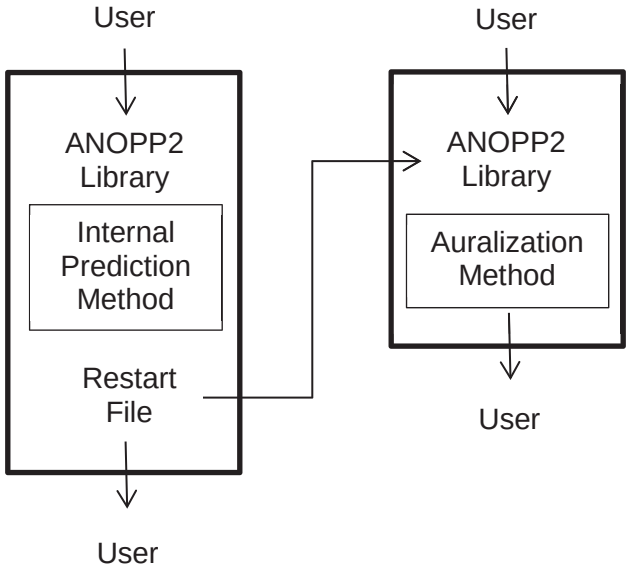

Figure 4: Diagram of the restart architecture.

C. Plugin Option

A third auralization architecture option uses the ANOPP2 plugin system. This system allows users to incorporate external methods into the ANOPP2 architecture to supplement or even replace ANOPP2 functionality without having direct access to ANOPP2 source code. This is accomplished through a dynamic-library-based plugin system where each external method is

embedded in a dynamic library. ANOPP2 couples the dynamic libraries with the system noise prediction architecture.

Figure 5 shows the communication between a user, the ANOPP2 library, and two methods: an internal prediction method and an external auralization method implemented using a dynamic-library. The user provides information to ANOPP2, including the aircraft and its flight mission. The ANOPP2 library then provides that information to both the internal method and to the plugin; the plugin performs the auralization using the aforementioned Auralization API and stores the resulting pressure time histories in an ANOPP2 observer object using the API for that object. As with the inline method, ANOPP2 library functions make that information accessible to the user. The user interfaces with ANOPP2 through a user written code regardless of whether the method is contained within the ANOPP2 library or the dynamic library. It is important to note that new methods, because they are embedded in dynamic libraries, may be added to the system without having to rebuild ANOPP2 because the interfaces between the ANOPP2 library and all plugins remain the same.

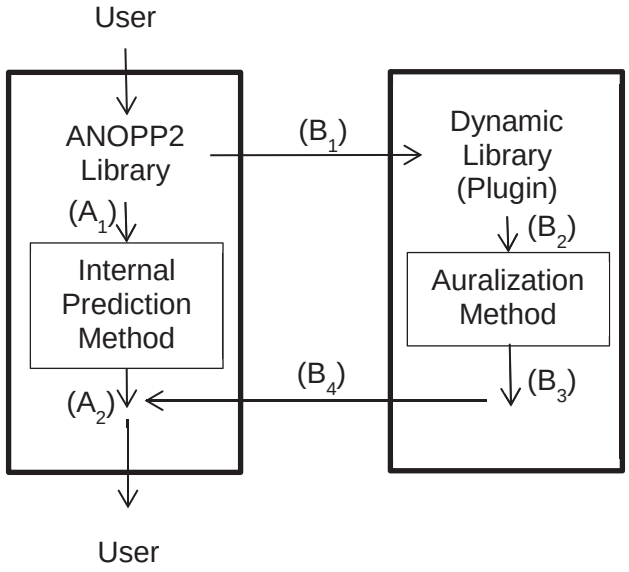

Figure 5: Communication between the user, the ANOPP2 library, and two methods: an internal prediction method and an external auralization method via a plugin. Information flow is denoted by (A) for the internal method and (B) for the external method.

5. CONCLUDING COMMENTS

In an effort to make the process of auralizing flyover noise a common element of system noise prediction, an auralization capability is being developed for use with NASA's next generation aircraft noise prediction program ANOPP2. The capability is built upon a new Auralization library and its associated Application Programming Interface, which has functions and data structures for path, synthesis, and propagation processing. Three architectures have been identified allowing auralization to be performed using the ANOPP2 API. These are inline, restart, and plugin. The choice of architecture is made by the user to best serve the analysis at hand. Because the Auralization API is meant to be extensible, additional capabilities may be added by the user to fulfill specialized requirements, or to interface to other noise prediction programs such as the Advanced Acoustic Model.[14] Finally, the Auralization library and its API will be more easily maintained than the present set of auralization codes since they will be contained with a single common code base.

ACKNOWLEDGEMENTS

This work was performed with support from the Aeronautical Sciences Project of the NASA Fundamental Aeronautics Program.

RERERENCES

1. Rizzi, S.A., Aumann, A.R., Lopes, L.V., and Burley, C.L., "Auralization of hybrid wing body aircraft flyover noise from system noise predictions," *51st AIAA Aerospace Sciences Meeting*, AIAA-2013-0542, Grapevine, TX, 2013.
2. Arntzen, M., Rizzi, S.A., Visser, H.G., and Simons, D.G., "A framework for simulation of aircraft flyover noise through a non-standard atmosphere," *18th AIAA/CEAS Aeroacoustics Conference*, AIAA-2012-2079, Colorado Springs, CO, 2012.
3. Rizzi, S.A., Sullivan, B.M., and Aumann, A.R., "Recent developments in aircraft flyover noise simulation at NASA Langley Research Center," *NATO Research and Technology Agency AVT-158 "Environmental Noise Issues Associated with Gas Turbine Powered Military Vehicles" Specialists' Meeting*, NATO RTA Applied Vehicle Technology Panel, Paper 17, pp. 14, Montreal, Canada, 2008.
4. "GoldServe, AuSIM3D Gold Series Audio Localizing Server System, User's Guide and Reference, Rev. 1d," AuSIM Inc., Mountain View, CA, October 2001.
5. "NetCDF (Network Common Data Form)," http://www.unidata.ucar.edu/software/netcdf/, Unidata, 2013.
6. Raney, J.P., "Development of a New Computer System for Aircraft Noise Prediction," *AIAA 2nd Aero-Acoustics Conference*, AIAA-75-536, pp. 5, Hampton, VA, March 24-26, 1975.
7. Zorumski, W.E., "Aircraft Noise Prediction Program Theoretical Manual, Parts 1 and 2," National Aeronautics and Space Administration, Langley Research Center, Hampton, VA NASA/TM-83199-PT-1 and PT-2, February 1982.
8. Smith, M., Carrilho, J., Molin, N., Piet, J.-F., and Chow, L., "Modeling Landing Gear Noise With Installation Effects," *13th AIAA/CEAS Aeroacoustics Conference (28th AIAA Aeroacoustics Conference)*, AIAA-2007-3472, pp. 16, Rome, Italy, May 21–23, 2007.
9. "Advanced Sound Propagation in the Atmosphere (ASOPRAT) User's Guide," V 3.0, The University of Mississippi, 1991.
10. Shen, J., Arntzen, M., Varbanescu, A.L., Sips, H., and Simons, D.G., "A Framework for Accelerating Imbalanced Applications on Heterogeneous Platforms," *Proc. Computing Frontiers 2013*, Ischia, Italy, May 14-16, 2013.
11. Rizzi, S.A. and Sullivan, B.M., "Synthesis of virtual environments for aircraft community noise impact studies," *11th AIAA/CEAS Aeroacoustics Conference*, AIAA-2005-2983, Monterey, CA, May, 2005.
12. Allen, M.P., Rizzi, S.A., Burdisso, R., and Okcu, S., "Analysis and synthesis of tonal aircraft noise sources," *18th AIAA/CEAS Aeroacoustics Conference*, AIAA-2012-2078, Colorado Springs, CO, 2012.
13. Rizzi, S.A., Aumann, A.R., Allen, M.P., Burdisso, R., and Faller II, K.J., "Simulation of rotary and fixed wing flyover noise for subjective assessments (Invited)," *161st Meeting of the Acoustical Society of America*, Seattle, WA, May 23-27, 2011.
14. Page, J.A., Wilmer, C., Schultz, T., Plotkin, K.J., and Czech, J., "Advanced Acoustic Model Technical Reference and User Manual," Wyle Laboratories, Inc., SERDP Project WP-1304, May 2009.