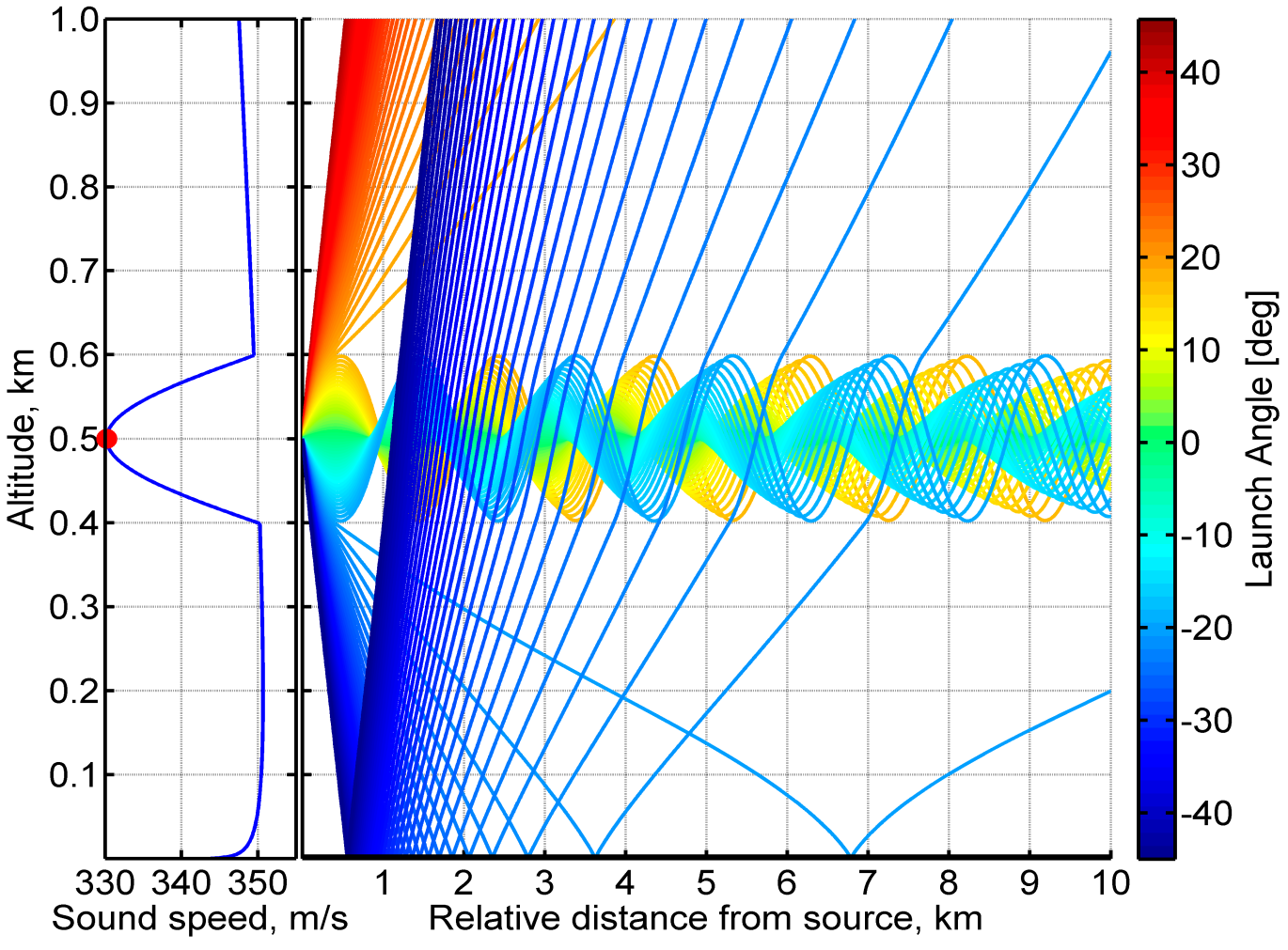
Acoustic ray tracing parallelization

Customer

National Aerospace Laboratory NLR

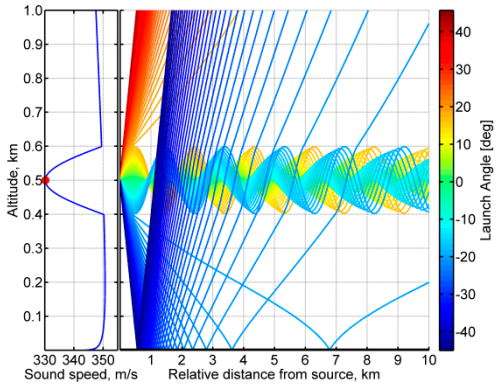
NLR-TP-2015-281 - July 2015



National Aerospace Laboratory NLR
Anthony Fokkerweg 2
1059 CM Amsterdam
The Netherlands
Tel +31 (0)88 511 3113
www.nlr.nl

EXECUTIVE SUMMARY

Acoustic ray tracing parallelization



Problem area

Application of non-homogeneous atmospheric effects in virtual acoustic simulation applications is rare. Earlier research showed that usage of atmospheric induced curved acoustic rays in such cases is possible, but that the computational expense is too large for real-time application. The current study partially solves this issue.

Description of work

To reduce the computational expense, use is made of the Graphical Processing Unit (GPU) of the computer. This allows the calculation of individual bundles of rays in parallel on several cores. Difficulties arise when particular acoustic rays have a longer travel time than others. Such conditions have an adverse impact on the computational time. To that end, an innovative workload balancing solution is proposed and applied.

Results and conclusions

The resulting calculations were executed using a dedicated framework, designed in collaboration with the TU Delft, which was called Glinda. The Glinda framework was applied to a flyover trajectory and calculating the atmospheric propagation loss at a

Report no.
NLR-TP-2015-281

Author(s)
M. Arntzen
D.G. Simons
J. Shen
A.L. Varbanescu
H. Sips

Report classification
UNCLASSIFIED

Date
July 2015

Knowledge area(s)
Aircraft Noise
Aeroacoustic and Experimental Aerodynamics Research

Descriptor(s)
Acoustic ray tracing
Parallel computing
Virtual community noise simulator

This report is based on a presentation held at the Noisecon conference, Denver (CO), USA, 26-28 August 2013.

discrete interval. By comparing the parallel and non-parallel implementation it could be concluded that a speed up of a factor 10 is feasible. Consequently, the total transmission loss of the acoustic signal could be calculated within roughly 40 milliseconds.

Applicability

NLR's virtual acoustic simulator, the 'Virtual Community Noise Simulator' (VCNS), runs at an update interval of 6 milliseconds. The ray tracing results should ideally be available at that update interval. The current research brought the computational time down from a maximum of half a second to 40 milliseconds. Hence, the algorithm is still not directly applicable. Strategies are proposed to update the algorithm, which cuts down another 20 milliseconds of the computational time. Furthermore, an advanced interpolation scheme could be used to feed the VCNS solutions that come available every 20 milliseconds. Hence, the application of curved rays in real-time virtual acoustic simulation is within grasp.

Acoustic ray tracing parallelization

M. Arntzen, D.G. Simons[1], J. Shen[1], A.L. Varbanescu[1] and H. Sips[1]

[1] TU Delft

Customer

National Aerospace Laboratory NLR

July 2015

This report is based on a presentation held at the Noisecon conference, Denver (CO), USA, 26-28 August 2013.

Customer	National Aerospace Laboratory NLR
Contract number	5003103
Owner	NLR + partner(s)
Division NLR	Air Transport
Distribution	Unlimited
Classification of title	Unclassified
Date	July 2015

Approved by:

Author	Reviewer	Managing department
M. Arntzen	D. Bergmans	F.J.J.M. Hermans i.o.
Date 31/07/15	Date 31/07/15	Date 03/08/15

Content

Abstract 5

1. Introduction 5

2. Ray tracing 6

3. Glinda framework 8

4. Application & results 9

5. Conclusions 12

References 12

This page is intentionally left blank.

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

Acoustic Ray Tracing Parallelization

Michael Arntzen*
Dick G. Simons
TU Delft, Aerospace Engineering
Air Transport & Operations
2600 GB, Delft, the Netherlands
michael.arntzen@nlr.nl

Jie Shen
Ana Lucia Varbanescu
Henk Sips
TU Delft, Software & Computer Technology
Parallel & Distributed Systems
2600 GA, Delft, the Netherlands

ABSTRACT

Aircraft flyover noise synthesis is usually executed with a straight propagation path approach. This is due to its simplicity and low computation demand. A straight path assumption eradicates atmospheric wind and temperature variations that occur in real-life situations. Taking such situations into account requires the use of more advanced modeling like ray tracing. To keep a low computation expense and to meet the near real-time requirements encountered in virtual acoustic simulation, these calculations have been parallelized. Ray tracing inherits an imbalanced workload that will dampen the efficiency of the computation. A general framework is developed to accelerate the computation. An application of the algorithm is used, for a typical aircraft flyover study, to compare the speed-up with respect to a sequential implementation of the algorithm. As a result of the fast computation, quasi real-time virtual acoustic simulations based on ray tracing are believed to be possible in the near future.

1. INTRODUCTION

A frequently encountered assumption in the propagation of sound, as used in noise synthesis, is the spherical spreading of a sound source [1, 2]. Consequently, the spherical spreading losses are evaluated as a function of the straight-line ray path distance between the source and observer. Such propagation effects are encountered in free-field conditions if the speed of sound is constant. However, atmospheric effects may render the spherical spreading law invalid. Variations in temperature and wind cause a curved acoustic ray path instead of straight, thereby modifying the spreading losses. To simulate this propagation effect, models based on the Helmholtz-equation or simpler frequency independent models exist [3]. The latter are computationally more efficient and ray tracing is the most prominent member of this family.

Recently, ray tracing was used to simulate the effects of atmospheric variations on the synthesis of aircraft flyover noise [4]. Under shallow propagation angles the effects of the curved path analysis proved to be important. If the aircraft is in the overhead position, these effects diminished and the straight path proved to be valid. The use of straight ray paths allows executing the calculation of propagation effects on the fly, which makes it applicable for virtual environments utilizing acoustic simulations. Including curved rays proved to be not possible due to the large computation time. The possibility of including curved rays in such a simulation environment is thus only realistic if faster algorithms exist.

* Also affiliated with the NLR (Dutch National Aerospace Laboratory).

Ongoing advances in Graphics Processing Unit (GPU) technology allow parallelizing the algorithms to accelerate the computation. Therefore it was studied if acoustic ray tracing could benefit from this technology [5]. One of the drawbacks, from a GPU stand point of view, is that ray tracing algorithms usually employ a variable time-step. Such an approach, combined with particular atmospheric effects, causes particular rays to have a long calculation time compared to other rays. This results in a workload imbalance, which cannot be efficiently evaluated if the algorithm is "embarrassingly parallel" executed on a GPU. To that end, a solution was recently developed by utilizing heterogeneous platforms with the multi-core CPU* (Central Processing Unit) and the GPU. The resulting framework that was created is called Glinda [5]. Applications were tested for single source positions and initial tests showed promising speed-up of the ray tracing algorithm.

In the current study, Glinda is used to provide input for a typical aircraft flyover noise synthesis study. An entire aircraft trajectory is simulated with Glinda rather than a single source position. Consequently, it is evaluated if the performance provided by Glinda suffices for the potential use in the aircraft noise synthesis.

2. RAY TRACING

Ray tracing is a well-established method within the acoustic community to evaluate propagation characteristics [3]. Different solution methods exist ranging from systems of coupled differential equations to simple algebraic implementations and even semi-empirical methods. The current implementation uses an algebraic implementation to evaluate Snell's law of refraction. This method stems from the optics world and was essentially optimized to do ray tracing with minimal computation operations [6].

To start the ray tracing, a layered atmosphere is assumed where each layer contains a constant temperature and wind speed. At a layer boundary these characteristics change discretely. Acoustic refraction, i.e. a change in ray angle, will occur at such a boundary due to the change of medium characteristics. This is shown in Figure 1.

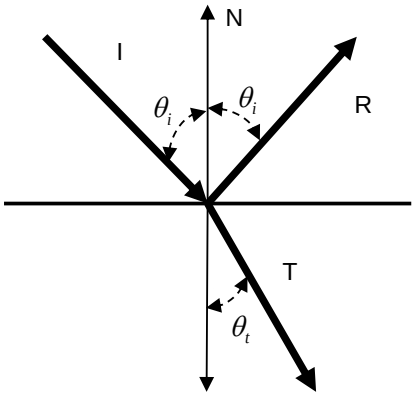


Figure 1: An incident ray segment (I) exhibits change in direction when transmitted (T) in a medium with different (atmospheric) characteristics. Reflection (R) occurs for ground or internal reflection. N is the normal vector, θ_i is the incidence angle and θ_t is the transmitted angle.

* Throughout the paper, the multi-core CPU is denoted as CPU.

Snell's law is used to evaluate the change in ray angle. After some mathematical manipulations a simple algebraic form of Snell's law, using the notations of Figure 1, is found. The transmitted ray follows from:

$$\boldsymbol{T} = \eta_{it}\boldsymbol{I} + \left[\eta_{it}cos(\theta_i) - \sqrt{1 + \eta_{it}{}^2(cos(\theta_i)^2 - 1)}\right]\boldsymbol{N}, \quad (1)$$

where, the bold font indicates normalized vectors and η_{it} is the index of refraction that typifies the transition between the two media. The index of refraction is the ratio of the effective sound speeds (combination of temperature and wind) in both media. The algorithm starts by launching a ray segment, with length equal to the product of the time step and the local speed of sound, at an initial angle. This incident segment is refracted into a transmitted ray segment, as calculated by equation 1, which becomes the incident segment on a different atmospheric layer at the next time step. Since the length of a ray segment is dictated by the time step, the time step is proportional to the computation workload. At shallow incidence angles the ray tracing time step is reduced since it is susceptible to refraction. The time step is also reduced in the vicinity of the ground, i.e. in the vicinity of the receiver. This is due to the fact that irregularities in travel time lead to audible artifacts and therefore imposes a high temporal resolution. The variable time step is allowed to vary from 10 ms to 0.5 ms depending on the aforementioned conditions.

The ray sound intensity on the ground follows from the Blokhintzev invariant [7] and is implemented as a focusing factor [3]. Traditional ray tracing limitations, like shadow zones and caustics, are treated based on comparisons with a Fast Field Program (FFP) [8]. It shows that a 10 dB lower loss, as calculated by spherical spreading, approximates the intensity in caustics reasonably well. This result was also found by different research in literature [9]. The intensity in shadow zones was found to be a factor of the distance from the limiting ray into the shadow zone and the sound speed profile. The output of the ray tracing model contains the focusing factor, travel time of the ray, launch angle, incidence angle and the accumulated atmospheric absorption. This provides enough input to calculate the gain, time delay and acoustic filters as used in the simulation of aircraft flyover noise in virtual environments.

Under particular conditions, the atmosphere may contain an acoustic duct where rays are consecutively bent upwards and downwards. Figure 2 shows such an extreme condition.

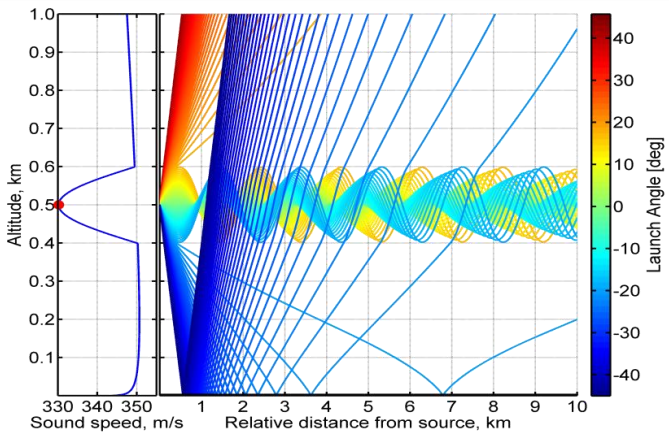


Figure 2: Rays are trapped in an acoustic duct if the source (red dot in the sound speed profile) is in such an area. The acoustic duct region can be distinguished as the indentation in the sound speed profile on the left.

Acoustic ducts can also occur near the ground if the sound speed profile includes an inversion. If a source is at an altitude where these rays occur, the computation time is adversely effected. This forms a limitation for the current implementation in virtual acoustic simulators.

3. GLINDA FRAMEWORK

Glinda [5] is our novel framework to accelerate acoustic ray tracing simulation. It is adaptive to scenarios useful for both balanced and imbalanced workloads, i.e. ray tracing with or without acoustic ducts.

When there is no acoustic duct, the workloads of rays are relatively balanced (see Figure 3-a). All the rays execute the same number of steps. In this situation, Glinda parallelizes the computation on the GPU or on the multi-core CPU, because the whole computation can be evenly distributed on the processing cores of the underlying processor.

When an acoustic duct occurs like in the case shown in Figure 2, the workloads of rays are imbalanced (see figure 3-b). Most rays finish their simulation within 1,000 steps, while the rays trapped in the duct area have up to 12,000 steps. These trapped rays pick finer time steps to ensure sufficient simulation accuracy, and finally form a narrow “peak” out of the “bottom” in the workload distribution. In order to efficiently parallelize the computation, Glinda adopts a heterogeneous approach by utilizing both the GPU and the multi-core CPU. As the “bottom” part is wide and relatively flat, a GPU with hundreds or thousands of cores provides the massive parallelism suitable to accelerate this part. The CPU has fewer, yet larger and faster cores, but provides enough parallelism to efficiently process the narrow “peak” part. Glinda cuts the whole computation into the “bottom” task on the GPU and the “peak” task on the CPU, and runs the two tasks again in parallel. By choosing and performing this task-mapping, Glinda provides a balanced execution of the workload. In turn, this leads to significant improvements in performance when compared against the CPU or the GPU used in isolation.

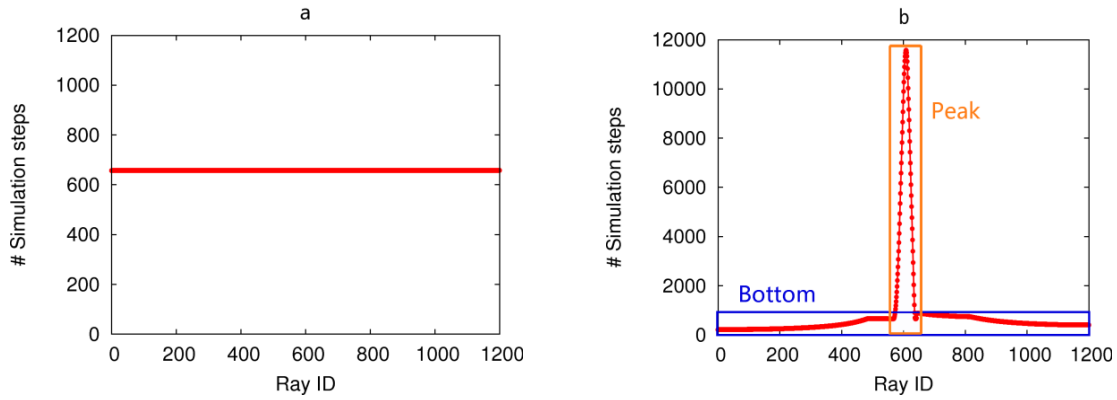


Figure 3: (a) A balanced workload distribution. (b) An imbalanced workload distribution. The x-axis represents the rays launched at different angles, which are identified by ray id starting from 1 to 1200. The y-axis represents the workload of a ray.

As the workload shape (flat or with "peaks") can depend on the atmospheric conditions, and the hardware platform can be altered or upgraded, Glinda is designed to be adaptive to all these changes. It automatically selects the right parallel solution and hardware configuration for the

user. In addition, to ensure the best performance, the optimal execution configuration (e.g., the "cut point" in the imbalanced workload distribution) is obtained through auto-tuning.

Figure 4 shows an overview of Glinda. The "*User interface"* receives the acoustic ray tracing parameters (e.g., the number of rays, atmospheric conditions, etc.), and interacts with the workload probe. The "*Workload probe"* characterizes the workload distribution by sampling, and the "*HW detector"* detects the available hardware resources. According to the outputs from the workload probe and the HW detector, the "*Matchmaker"* proposes the optimal code-platform pair. The code candidates are stored in the "*Code library"*, which has three parallel solutions: all on the CPU, all on the GPU, and the use of both. The "*Auto-tuner"* receives the selected code-platform pair and generates the optimal execution configuration through auto-tuning. As this can be time-consuming, the "*Config-predictor"* analytically detects (using the physics principles) a theoretical "cut point", which can then be used to skip the time-consuming auto-tuner. The "*Execution unit"* performs the real computation. If the results are correct and the computation time meets the user requirement, the "*Check unit"* writes the code-platform mapping pair and the execution configuration into the *"Mapping table"* and the *"Config-predictor"*, respectively, for future uses. When a workload distribution is irregular, we first sort the rays by their number of steps. The sorting result is stored in the *"Indexing table"*, and used by the execution unit to de-sort the output data.

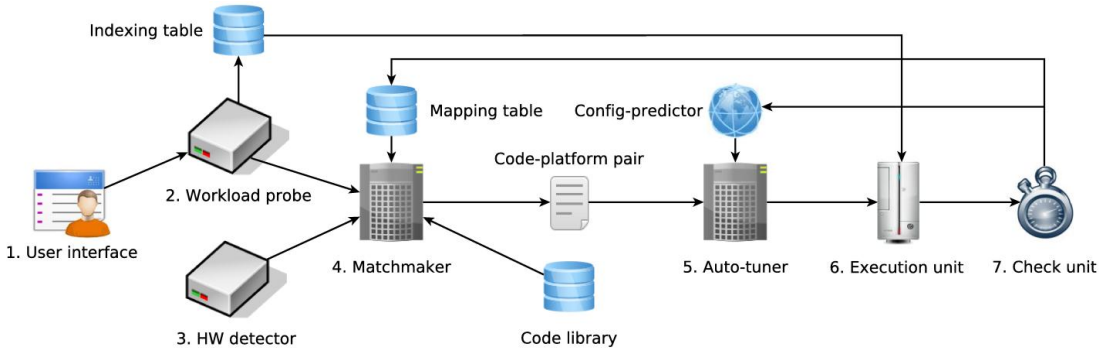


Figure 4: The overview of the Glinda framework.

4. APPLICATION & RESULTS

To put Glinda to the test, a test was executed to stress the ray tracing propagation algorithm. The considered atmosphere and (take-off) trajectories are shown in Figure 5.

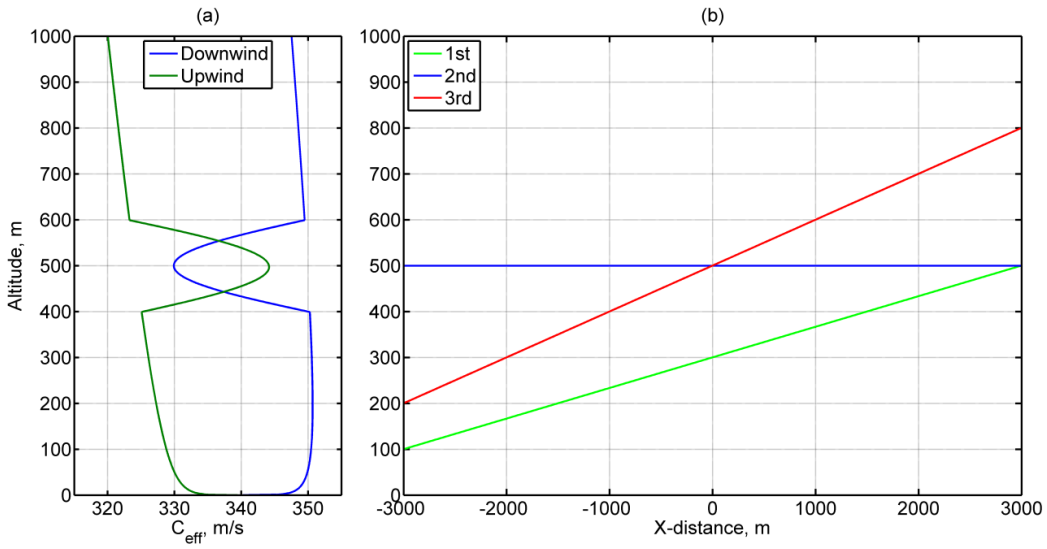


Figure 5: (a) The effective sound speed profile for the downwind and upwind conditions. (b) The 2D trajectories of the 3 flights flying from negative to positive x-distance. Negative x-distances imply upwind propagation conditions whereas positive implies downwind conditions.

The atmospheric effects are accumulated in the effective sound speed profile (Figure 5-a), as used by Snell's law. As the aircraft position relative to the listener changes, the effect of wind on the effective sound speed profile changes. As a result, the forward radiated sound of the aircraft propagates with the upwind profile whereas the aft radiated sound propagates using the downwind profile. Figure 5-b shows the trajectories of the aircraft, here the 1st flight resembles a slow climbing aircraft (~1300 ft/min) and the 3rd flight resembles an aircraft that climbs twice as fast. The 2nd flight is at a constant altitude in the middle of the acoustic duct area. Consequently, if the 2nd flight is past the observer, the downwind conditions with a duct at the source altitude exist.

The aircraft is assumed to cover this trajectory in 60 seconds and is discretized every 100 milliseconds (ms.). Accordingly, there are 600 discrete source positions where ray tracing calculations have to be performed. These trajectory points, as will be used for reference in the coming figures, can be easily translated to the x-distance of Figure 5-b, i.e. -2000 is point 100, -1000 is point 200 and etcetera. Next, Glinda is used to calculate the propagation characteristics using a set of 1200 rays. For the current simulations, Glinda utilizes a dual-socket Intel Xeon E5645 six-core CPU (2.4 GHz, 24GB) and an NVIDIA Tesla C2050 GPU with 448 cores (1.15GHz, 3GB). For each case the sequential code and the parallel code (with Glinda) is evaluated. Both the sequential and parallel results coincide, thereby verifying the correct implementation of the parallel algorithm. Timing results of both implementations are plotted in Figure 6.

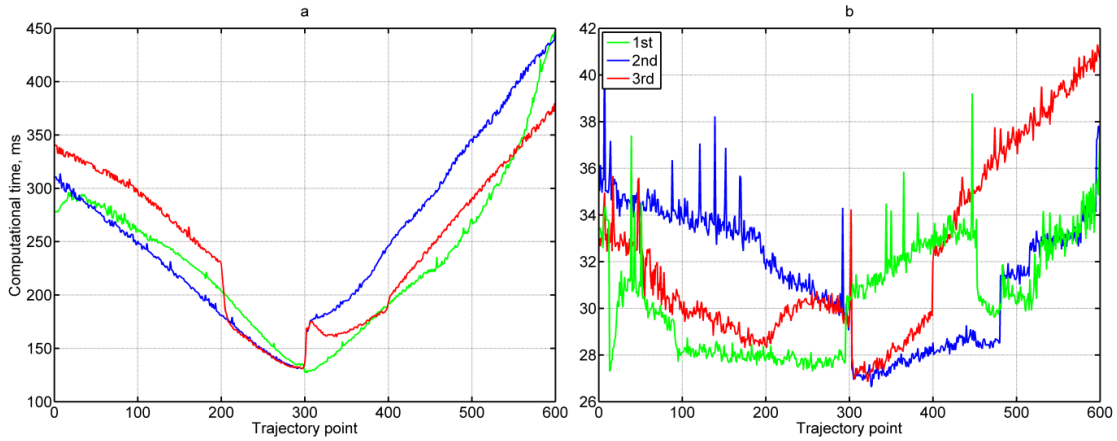


Figure 6: (a) Computation time (ms) for the sequential implementation. (b) Computation time for the parallel implementation. Please note the difference in magnitude (~factor 10) of the y-axis.

From Figure 6, the first observation is that for all trajectory points a significant speed-up of about 10 times is achieved. Since every 100 ms a new trajectory point is simulated, the sequential results are not ready when a new point has to be simulated. This is a show stopper for inclusion in real-time environments. For the parallel case the maximum computation time is approximately 42 ms. In comparison, the traditional straight ray path approach, as currently employed by virtual acoustic simulators, can be based on update intervals of 6 ms using multiple sources at the same time. However, a quasi real-time implementation could be achieved by interpolating between the propagation results for a single source when calculated in parallel using Glinda. Another difference is visible in the variation of the computation time for the trajectory points. Glinda balances the computation workload and, as such, the overall performance is more or less constant (the execution time difference is within 6 ms) compared to the sequential version. This makes it attractive for implementation in virtual acoustic simulation since, like the straight path, the computation time demand remains predictable.

In Figure 6-a, the 3rd trajectory shows a drop in computation time around trajectory point 200. At that position, i.e. 400 meters altitude and upwind conditions, the propagation characteristics change severely. If the aircraft is below that altitude, the inversion caused by the duct in upwind conditions, cause rays to have an imbalanced workload. In contrast, Glinda tuned the parallel implementation and balanced the workloads between GPU and the multi-core CPU at that trajectory point. If the aircraft is above this altitude, fewer rays are captured in the duct and the sequential computation time decreases. The reverse of this phenomenon occurs around trajectory point 300, i.e. direct overhead where conditions change from upwind to downwind, for the 1st and 2nd trajectory and more rays become trapped in the duct.

Figure 6-b shows the performance of the parallel implementation. As a result of the balanced workload, the performance is largely improved. There is a sharp change in workload shape for all the trajectories when switching from upwind to downwind conditions. At these points Glinda adjusts the optimal allocation of rays to be calculated on the CPU or GPU through auto-tuning for the new input conditions. The resulting “cut point” as found by auto-tuning, i.e. the point where the bottom and peak box intersect in Figure 3-b, is saved for future purposes. This saves the auto-tuning computation time although more optimal “cut points” can be found at other

trajectory points. This also causes the discrete changes in performance around point 400 (3rd flight), 450 (1st flight) and 480 (2nd flight). In the future we envisage that the "auto-tuning" can be scheduled according to keeping track of the varying atmospheric conditions and/or the computation time. Further studying should also highlight if the spikes, i.e. 2nd flight from point 100-200, are also caused by this phenomenon.

The results as delivered by Glinda need to be interpreted before they can be applied in virtual acoustic simulation. This process, i.e. constructing gain and filter coefficients, will add some small overhead, but this is not different than the sequential implementation. In the previous study [3], eigenrays were iteratively searched to find the propagation characteristics since less rays could be used. The current parallelization allows to use many rays and, dependent on the GPU memory size, could house up to 6000 rays in our experiment. However, calculating more rays will increase the overall computation time again. Making use of 1200 rays already eliminated the use of eigenray finding since the ray density on the grid is sufficient. The results from Glinda can thus directly be interpolated upon using an "Interpreter" for application in the simulator. By combining the ray tracing with the "Interpreter" directly in the simulator, the grid of the ray tracing results does not need to be stored in temporary files for offline access. This saves another quarter of the parallel computation time. In addition, as the results on the GPU and the CPU have to be transferred and gathered on the host (the CPU) for the latter "Interpreter", more computation time can be saved if only the results near the receiver is transferred. Although care is necessary to treat acoustic multiple paths and shadow zones since multiple rays or no rays can be present near a receiver.

5. CONCLUSIONS

Ray tracing offers promising results to be incorporated in the next generations of virtual acoustic simulators. Combining the power of all the computer's computation units allows a speed-up of the performance. Future generations of GPUs and CPUs are likely to further reduce the computation time. Final integration with noise synthesis algorithms needs one final interpretive step to extract the relevant acoustic factors. Given the computation efficiency demonstrated by Glinda this is believed to be the final step necessary for successful implementation in future virtual acoustic simulations.

REFERENCES

1. S.A. Rizzi and B.M. Sullivan, "Synthesis of virtual environments for aircraft community noise impact studies," *Proc. 11th AIAA/CEAS AeroAcoustics conference*, 2005, AIAA-2005-2983.
2. A. Sahai, E. Anton, E. Stumpf, F.Wefers and M. Vorlaender, "Interdisciplinary auralization of take-off and landing procedures for subjective assessment in virtual reality environments," *Proc. 18th AIAA/CEAS AeroAcoustics conference*, 2012, AIAA-2012-2077.
3. E.M. Salomons, "*Computational Atmospheric Acoustics"*, Kluwer Academic Publishers, London, 2001, 1st edition.
4. M. Arntzen, S.A. Rizzi, H.G. Visser and D.G. Simons, "A framework for simulation of aircraft flyover noise through a non-standard atmosphere," *Proc. 18th AIAA/CEAS AeroAcoustics conference*, AIAA-2012-2079.
5. J. Shen, M. Arntzen, A.L. Varbanescu, H. Sips and D.G. Simons, "A framework for accelerating imbalanced applications on heterogeneous platforms," accepted for: *Computing Frontiers*, 14-16 May, Ischia, Italy, 2013.
6. A.S. Glassner (editor), "*An introduction to ray tracing"*, Academic press limited, 1990, 3rd edition.
7. D.I. Blokhintzev, *Acoustics of non-homgeneneous moving medium*, 1956, NACA TM-1399.
8. Anon., *Prediction of sound attenuation in a refracting turbulent atmosphere with a Fast Field Programm",* ESDU 04011, May 2004.
9. C.I. Chessel, "Meteorological and ground effects on the propagation of aircraft noise close to the earth surface," *Journal of Sound and Vibration, **60(2)***, 251-266, 1978.

WHAT IS NLR?

The NLR is a Dutch organisation that identifies, develops and applies high-tech knowledge in the aerospace sector. The NLR's activities are socially relevant, market-orientated, and conducted not-for-profit. In this, the NLR serves to bolster the government's innovative capabilities, while also promoting the innovative and competitive capacities of its partner companies.

The NLR, renowned for its leading expertise, professional approach and independent consultancy, is staffed by client-orientated personnel who are not only highly skilled and educated, but also continuously strive to develop and improve their competencies. The NLR moreover possesses an impressive array of high quality research facilities.

***NLR** – Dedicated to innovation in aerospace*

www.nlr.nl