Parallel Ray Tracing for Underwater Acoustic Predictions

Rogério M. Calazan[1], Orlando C. Rodríguez[1], and Nadia Nedjah[2]

[1] SiPLAB, University of Algarve, LARSyS
Campus de Gambelas, 8005-139 Faro, Portugal
{a53956,orodrig}@ualg.pt
http://www.siplab.fct.ualg.pt
[2] Dep. of Electronics Engineering and Telecommunications,
Faculty of Engineering, State University of Rio de Janeiro
Maracanã, 20550-013, Rio de Janeiro, Brazil
http://www.eng.uerj.br/~nadia

Abstract The acoustic pressure is a fundamental metric provided by underwater acoustic models. It is used to compute the Transmission Loss (TL). Several underwater applications make use of the TL estimations. However, the acoustic pressure model for a real three-dimensional environment is computationally intensive. Aiming at performance improvement, parallel processing can be used so as the computational workload can be split into several tasks, which can be allocated on multiple processors to run concurrently. This paper proposes a parallel implementation of a ray tracing model using Open MPI to estimate the acoustic pressure. The performance of the proposed parallel implementation is evaluated via a tank scale experiment, which is used in order to provide the waveguide parameters and the TL data as well as to compare the obtained results against predictions. We also report and analyze the achieved speedup and efficiency. Furthermore, two other underwater acoustic models are used to compare the runtime and TL results to those obtained by the proposed parallel implementation.

Keywords: Parallel computing · Open MPI · Underwater acoustics · Ray tracing

1 Introduction

Ocean acoustic models are important numerical tools, which provide a detailed description of sound propagation in the ocean waveguide. This is achieved through the computation of the pressure field transmitted by a set of acoustic sources and received on a set of hydrophones. Ocean acoustic models can be classified into different types, depending on the particular analytical approximation of the wave equation that the model implements numerically. Ray tracing models, for instance, are based on geometrical optics and address the solution of the wave equation using a high frequency approximation, which leads to the computation of wavefronts based on ray trajectories. Moreover, several acoustic models include the capacity of handle with *out-of-plane* propagation. That is, these are able to take into account with the environmental variability in range, the depth and azimuth, induced by a three-dimensional bathymetry or/and a sound speed

field [20,7,19]. Furthermore, the acoustic pressure is a fundamental metric provided by these numerical models and is used to get the transmission loss (TL). The TL is a standard metric used in the underwater acoustic field to measure the variation of a signal strength with distance [7]. The TL is used as a key point in the sonar equations [5], underwater noise predictions [18] and source tracking [2], among many others. However, the calculation of the acoustic pressure in a real three-dimensional environment can lead is computationally very intensive and requires a big deal of time to be obtained.

Parallel processing is a strategy used to provide faster solutions to computationally complex problems by splitting the workload into sub-tasks that would be allocated on multiple processors to run concurrently. In this sense, the distributed memory architecture is widely employed to achieve high performance computing. This architecture takes advantage of a Message Passing libraries to perform communication and synchronization, such as Open MPI [11].

This paper proposes a parallelization strategy of a ray tracing model as well as its implementation using Open MPI to compute the acoustic pressure. In order to evaluate the proposed strategy and its implementation, a tank scale experiment is used to provide the waveguide parameters and the TL data to compare against predictions. The speedup and efficiency achieved are reported and discussed. Furthermore, two other underwater acoustic models are used to compare the requirements in terms of execution time and precision of the TL results to those imposed by the proposed implementation. The results show that a great deal of improvement is achieved by parallel implementation without degrading the result precision and accuracy.

The remainder of this paper is organized into six section. First, in Section 2, we comment on some related works. After that, in Section 3, we briefly describe the TRACEO3D model; Then, in Section 4, we explain the proposed strategy and its parallel implementation; Thereafter, in Section 5, we provide a description of the tank scale experiment; Subsequently, in Section 6, we provide a thorough discussion of the obtained results and compare them to those provided by two existing related models; Finally, in Section 7, we present the conclusions and point out some directions for future works.

2 Related works

In [1], the authors present a parallel implementation using MPI libraries of a parabolic equation based algorithm to investigate 3D acoustical effects. The results demonstrate that the parallel implementation reduces drastically the execution time, solving an idealized data set. A realistic case of underwater acoustic propagation is also addressed. The MPI approach is similar to the one used in our implementation. However, in our work the objective is to parallelize a ray tracing based model, which works more efficiently in the case of high frequencies.

A GPU-based parallel implementation of split-step Fourier parabolic equation is presented in [6]. The authors show that the GPU version achieves a 10x faster than a multi-core version using OpenMP. Several idealized test cases are performed to evaluate the implementations. They concluded that the multi-core version did not inspire confidence regarding timing measurements. In [21] the authors report a parallel implementation of seeking 3D eigenrays with a irregular seabed using OpenMP. The obtained

results are related to an idealized test case and the implementation is 3.76x faster than the sequential implementation, yet keeping the same accuracy. Nonetheless, the performance evaluation is done only using a reduced number of cores and the solution scalability is limited to one processor. Unlike these existing implementations, we put effort to develop an efficient multi-core implementation. Moreover, we evaluate its performance in a real experimental case, regarding its efficiency and accuracy.

3 The TRACEO3D Ray Tracing Model

The TRACEO3D model is a recent three-dimensional version of the TRACEO ray model [17,16]. It is under current development at the Signal Processing Laboratory (SiPLAB) of the University of Algarve. The model is able to predict acoustic pressure fields and particle velocity in environments of elaborate boundaries. The model produces ray, eigenray, amplitude, travel time information and it is able to take in account the *out-of-plane* propagation.

Overall, TRACEO3D produces a prediction of the acoustic field via two steps: first, the set of Eikonal equation is solved in order to provide the ray trajectories; second, the ray trajectories are considered as the central axes of Gaussian beams, and the acoustic field is computed as a coherent superposition of beam influences. After that, according to the position of a given hydrophone, the acoustic pressure can be obtained.

Algorithm 1 summarizes the main steps of acoustic pressure calculation. It is important to note that the total number of rays that will be required to produce the prediction depend not only of the set of elevation angles but also of the set of the azimuth angles. The computational time is proportional to the number of rays, given by n, and the hydrophone array size, given by h. Furthermore, n is a problem dependent parameter because it is related to the amount of rays needed to sweep the 3D waveguide, based on θ and ϕ grazing angles.

Algorithm 1 Sequential TRACEO3D version

```
1:  load initial values
2:  let φ = set of azimuth angles
3:  let θ = set of elevation angles
4:  let h = number of hydrophones
5:  consider n = φ × θ number of rays
6:  for j := 1 → n do
7:     launch ray_j with φ_j and θ_j
8:     solve the Eikonal equations
9:     compute the dynamic equations
10:    for l := 1 → h do
11:       compute the acoustic pressure for hydrophone_l
12:    end for;
13: end for;
14: return the acoustic pressure computed for all rays
```

4 TRACEO3D on Distributed Memory Multi-core Processors

The Distributed Memory architecture is an efficient way to achieve high performance computing with multiprocessors. In this multiprocessor organization, each processor has his own private physical address space [12]. The processors are interconnected via a high-speed network used for message exchange as illustrated in Fig 1.

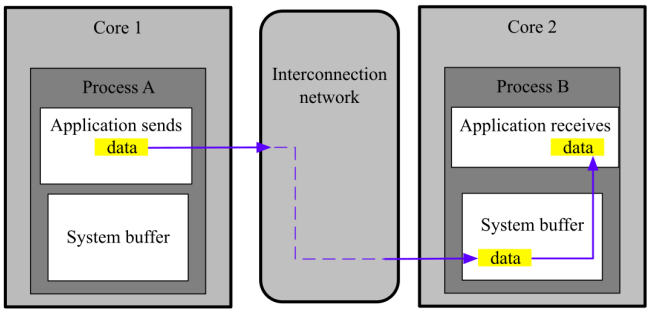


Figure 1: Communication by message in a distributed memory multi-core architecture

4.1 MPI basic concepts

The Message Passing Interface (MPI) is a specification for a standard message library, which was defined in the MPI Forum [10]. The Open MPI [11] is a open source MPI implementation of the MPI specification, designed to be portable and efficient for high-performance architectures. A parallel process using MPI has his own private address space. When a process, say A needs to communicate with another process, say B, then process *A* sends some data stored in its address space to the system buffer of process *B*, which can reside in distinct processor, as show in Fig. 1. The system buffer is a memory space reserved by the system to store incoming messages. This operation is cooperative and occurs only when process *A* performs a *send* operation and process *B* performs a *receive* operation.

There are blocking and no-blocking primitives for sending/receiving messages. In the case of blocking primitives, the processes involved in the operation would stop the program execution until the send/receive operation is completed. Otherwise, the program execution continues immediately after scheduling the communication. The MPI lists the communicating processes in groups, wherein the processes are identified by their classification within that group. This classification within the group is called ranking. Thus, a process in MPI is identified by a group number followed by its rank within this group. As a process may be part of more than one group, it may have different ranks. A group uses a specific communicator that describes the universe of communication between processes. The MPI_COMM_WORLD is the default communicator. It includes all processes, defined by the user, in an MPI application.

Algorithm 2 displays some of the main MPI primitives using FORTRAN. The command at line 5 defines and starts the environment required to run the MPI. The statement at line 6 identifies the process within a group of parallel processes. The function at line 7 returns the number of processes within a group. From that point on, each process runs in parallel as specified in the parallel region. Running rocesses may cooperate with each other via message passing. The routine at line 9 ends MPI processes.

Algorithm 2 Examples of MPI primitives implemented in Open MPI using FORTRAN

```
 1: program example
 2: include mpif.h
 3:       integer rank, size
 4:       /* Sequential region */
 5:       call MPI_INIT()
 6:       call MPI_COMM_RANK(MPI_COMM_WORLD,rank)
 7:       call MPI_COMM_SIZE(MPI_COMM_WORLD,size)
 8:            /* parallel region */
 9:       call MPI_FINALIZE()
10:       /* Sequential region */
11: return
```

4.2 Parallel Implementation

The approach for the parallel TRACEO3D Open MPI extension (TRACEO3Dompi) is described in Algorithm 3. First, in line 2, a parallel environment is initialized for p distinct processes. After that, each process loads the initial conditions and then, the algorithm proceeds with the division of the workload at ray level to balance the load of existing processes, as shown in line 8. However, not only the number of rays can influence the workload, the difference among the rays can also do so. When a given ray is launched, it follows his own path and has his own propagation time and because of that it has his own execution time. This characteristic could lead to a unbalance within the process's workload, so a process may require more computational effort to trace the ray than another one, thus increasing the overall execution time.

Nevertheless, rays with adjacent launch angles have usually very similar path and consequently have similar execution times. Thus, to avoid that the same process gets all the rays that might be more time consuming, adjacent rays are gathered in different processes. Thus, a division optimization is performed to balance the workload. This process is executed in line 11. In line 12, each process starts to compute his respective set of rays. For each ray, the set of Eikonal equations is solved. After that, the computation of the dynamic equations is performed in line 15. Then, the contribution of the ray for the acoustic pressure in each hydrophone is calculated in line 17. When this is done for k rays, each process sends a message to MASTER to compute all the contributions and return the final result. Note that, there are only two moments when interprocess

communication occurs: at the beginning, as shown in line 2, and at the end, as shown in lines 20 – 24. It is noteworthy to point out that, as in the case of ray tracing algorithms, this requires more computations than communications.

Algorithm 3 Parallel TRACEO3D Open MPI Extension

1: **let** p = number of processes
2: **MPI_Init()**
3: **generate** $rank$ for each process
4: **load** initial values
5: **let** ϕ = set of azimuth angles
6: **let** θ = set of elevation angles
7: **let** h = number of hydrophones
8: **consider** $n = (\phi \times \theta)\,/p$ number of rays per process
9: **do** $i = n \times rank$
10: **do** $k = i + n$
11: **select** the launch angles $\phi_{i..k}$ and $\theta_{i..k}$ according to rank
12: **for** $j := i \rightarrow k$ **do**
13: **launch** ray_j with ϕ_j and θ_j
14: **solve** the Eikonal Equations
15: **compute** the Dynamic Equations
16: **for** $l := 1 \rightarrow h$ **do**
17: **compute** the acoustic pressure for hydrophone$_l$
18: **end for**;
19: **end for**;
20: **if** $rank = Master$ **then**
21: **receive** the computed acoustic pressure for $rank$
22: **else**
23: **send** the computed acoustic pressure to $Master$
24: **end if**;
25: **MPI_Finalize()**
26: **return** the acoustic pressure computed for all set of processes

5 The Scale Tank Experiment

An indoor shallow-water tank of the LMA-CNRS laboratory in Marseille was used to acquired the experimental data, which is described in details in [19,8]. Therefore, only a concise description is presented here. The inner tank dimensions are 10m long, 3m wide and 1m depth. The source and the receiver are both aligned along the across-slope direction, as shown in Fig. 2. The bottom was filled with sand and a rake was used to produce a slope angle $\alpha \approx 4.5^{\circ}$; sound speed in the water was considered constant and corresponded to 1488.2m/s. The bottom parameters correspond to $c_p = 1655$m/s, $\rho = 1.99$g/cm^3 and $\alpha_p = 0.5$dB/λ. The source was located at 8.3m depth, at a range of

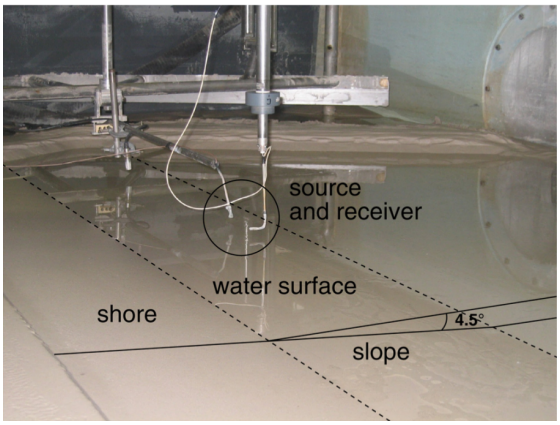


Figure 2: Indoor shallow-water tank of the LMA-CNRS laboratory of Marseille [19]

44.4m from the origin. Thus, for modeling purposes, the geometry of propagation corresponds to the waveguide shown in Fig. 3, wherein the cross-slope range corresponds to 5km. The ASP-H (for horizontal measurements of the across-slope propagation)

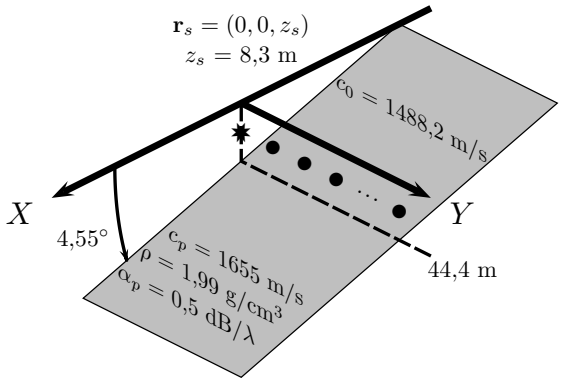


Figure 3: Upslope environment parameterization for modeling prediction computation

data set is composed of time signals recorded at a fixed receiver depth and at several source/receiver distances starting from $r = 0.1$m until $r = 5$m in increments of 0.005m, providing a sufficiently fine representation of the acoustic field in terms of range. Three different receiver depths are considered, namely 10mm, 19mm and 26.9mm, corresponding to the data subsets referenced as ASP-H1, ASP-H2 and ASP-H3, respectively. Acoustic transmissions were performed for a wide range of frequencies. However, comparisons are presented only for data from the ASP-H1 subset with a highest frequency of 180.05 kHz; this is due to the fact that the higher the frequency the better the ray prediction. It is important to note that a scale factor of $1000 : 1$ is required to properly

modify the frequencies and lengths of the experimental configuration. This implies that the following conversion of units is adopted: experimental frequencies in kHz become model frequencies in Hz, and experimental lengths in mm become model lengths in m. For instance, an experimental frequency of 180.05 kHz becomes a model frequency of 180.05 Hz, and an experimental distance of 10mm becomes a model distance of 10m. Sound speed remains unchanged, as well as compressional and shear attenuations.

6 Results and Analysis

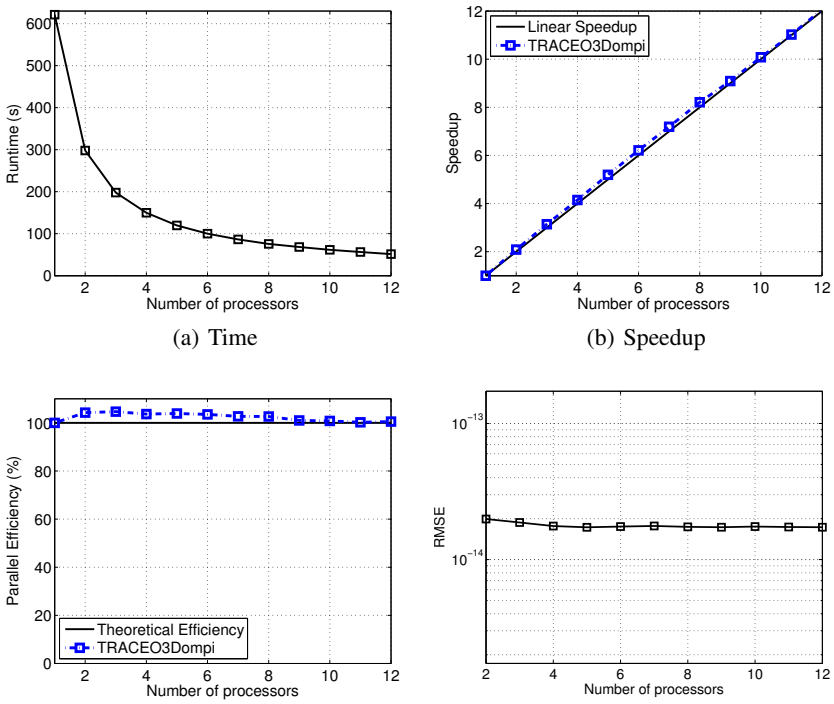
The set of waveguide parameters provided by the tank scale experiment are used as input for the models to compute the predictions. The models execution is performed by a computer server with two CPUs Xeon(R) E5-2420 of 1.90GHz where each CPU has 6 physical cores. In order to analyze the performance and scalability of the parallel implementation, the execution time of the sequential TRACEO3D version is compared to that of the TRACEO3Dompi using different number of processors. Table 1 presents the numbers of processor followed by the achieved execution time and speedup as shown in Fig. 4 (a) and (b) respectively. Each MPI processes is mapped in a processor. In Fig. 4 (c) the efficiency of the parallel implementation is shown, which is described as the speedup divided by the number of processes [3].

Table 1: Execution times and speedups for the sequential *vs.* parallel versions of TRACEO3D

#Processors	1	2	3	4	5	6	7	8	9	10	11	12
Time (s)	621.1	297.9	197.8	149.8	119.5	100.0	86.4	75.6	68.3	61.6	56.3	51.4
Speedup	1	2.1	3.1	4.1	5.1	6.2	7.2	8.2	9.1	10.1	11.0	12.1

The results show that the parallel implementation achieves a linear speedup. Furthermore, when the number of processors exploited is 2 to 8, it can be noted that the efficiency is above 100%. This is believed to be due to the superlinearity effect of caches [3], as the parallel implementation scatters the items of several vectors, which introduces a lower memory use by every parallel process. Moreover, Fig. 4 (d) presents the root mean square error (RMSE) for TL predictions results between the sequential and parallel implementations. We can note that the difference between the values is lower than 10^{-13} for any set of processes. Another issue is that the parallel efficiency is restricted to address each process to one physical core. The parallel processes do not take advantage of virtual cores. It is believed that the intensive computation makes use of the resources present at the physical core disabling the advantage of simultaneous multithreading technology [12].

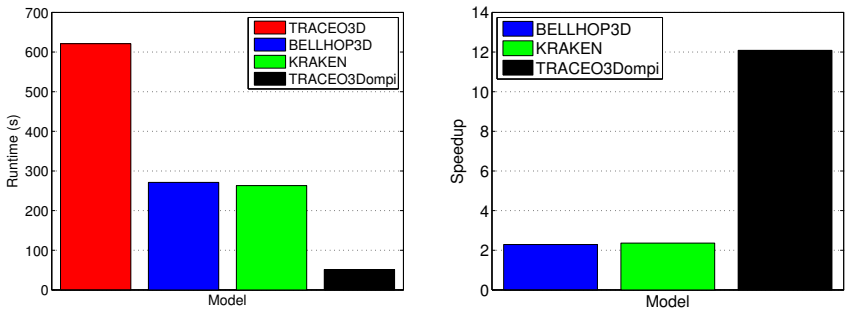
A further analysis regarding performance and accuracy of TRACEO3Dompi is performed using two other acoustic models: BELLHOP3D [15], which is also based in ray theory and is a three-dimensional extension of Bellhop ray model [14] and KRAKEN, which is a model based on normal mode theory [13,9]. Both models are able to produce predictions, which take into account *out-of-plane* propagations. These models are



(a) Time (b) Speedup

(c) Efficiency (d) RMSE

Figure 4: The TL achieved results of sequential *vs.* parallel implementations



(a) Execution time (b) Speedup

Figure 5: Execution time for model predictions computation using the tank scale parameters and speedup to TRACEO3D sequential version

available at Ocean Acoustic Library [4]. Table 2 shows the results regarding execution times and speedups based on TRACEO3D.

Table 2: Execution times and speedups for the models of the tank scale predictions

Model	TRACEO3D	KRAKEN3D	BELLHOP3D	TRACEO3Dompi
Runtime (s)	621.1	271.10	263.05	51.41
Speedup	1	2.3	2.4	12.1

Fig. 5 (a) and (b) show the results of execution time and speedup respectively. We can see that the parallel implementation is up to 12x faster than the sequential version and up to 5x faster that the BELLHOP3D and KRAKEN. Regarding the experimental data, and to allow for a visual comparison, the results of TL predictions of KRAKEN are depicted in Fig 6 (a) while those regarding BELLHOP3D and TRACEO3D are depicted in Fig. 6 (b). As it can be clearly observed, KRAKEN prediction is only valid for a small wedge, while TRACEO3D and BELLHOP3D predictions are able to follow accurately the experimental data.

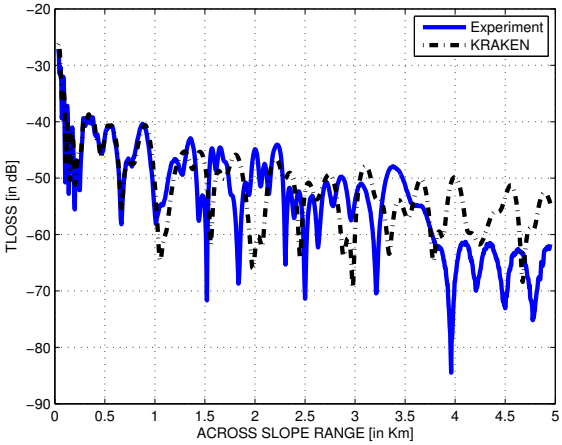
7 Conclusions

This paper presents and investigates the performance of the parallel implementation of TRACEO3D using Open MPI to compute the acoustic pressure. The implementation is evaluated using a tank scale experiment, which provides an ideal reference of a waveguide parameters and TL data as a realistic three-dimensional propagation. Two other well know acoustic models are used to benchmark the parallel version regarding its performance in terms of efficiency and accuracy.

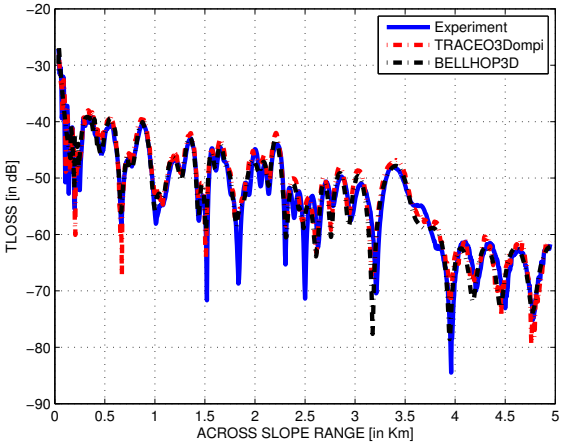
The results show that the parallel implementation offers a linear speedup with respect to the number of processors used as the workload becomes well distribute. Each parallel process addresses only one physical core because the simultaneous multithreading do not improve the performance results. The implementation is up to 12x faster than the sequential version without any loss the accuracy. When compared with other models, the TRACEO3Dompi is 5x faster as well as it is able to predict the experimental curve with high accuracy.

Future works include an investigation of the eigenray search problem and the particle velocity parallelization as part of the parallel extension. Moreover, it is planned to develop a parallel version of the studied model addressed to a graphic processing unit.

Acknowledgments This work receives support from the Foreign Courses Program of CNPq and the Brazilian Navy. Thanks are due to the SiPLAB research team, LARSyS, FCT, University of Algarve. The authors are also deeply thankful to LMA-CNRS for allowing the use of the experimental tank data discussed in this work.



(a) KRAKEN



(b) TRACEO3Dompi

Figure 6: Comparisons of the experimental results for LMA CNRS H1 @ 180.05Hz

References

1. Castor, K., Sturm, F.: Investigation of 3d acoustical effects using a multiprocessing parabolic equation based algorithm. Journal of Computational Acoustics 16(02), 137–162 (2008)
2. Felisberto, P., Rodriguez, O., Santos, P., Ey, E., Jesus, S.: Experimental results of underwater cooperative source localization using a single acoustic vector sensor. Sensors 13(7), 8856–8878 (Jul 2013), http://dx.doi.org/10.3390/s130708856

3. Grama, A., Karypis, G., Kumar, V., Gupta, A.: Introduction to parallel computing. Addison Wesley, 2 edn. (2003)
4. HLS Research: Ocean acoustics library. http://oalib.hlsresearch.com/ (Jan. 2017)
5. Hodges, R.P.: Underwater acoustics: Analysis, design and performance of sonar. John Wiley & Sons (2011)
6. Hursky, P., Porter, M.B.: Accelerating underwater acoustic propagation modeling using general purpose graphic processing units. In: OCEANS'11 MTS/IEEE KONA. pp. 1–6 (Sept 2011)
7. Jensen, F.B., Kuperman, W.A., Porter, M.B., Schmidt, H.: Computational Ocean Acoustics. AIP, New York, 2nd edn. (2011)
8. Korakas, A., Sturm, F., Sessarego, J.P., Ferrand, D.: Scaled model experiment of long-range across-slope pulse propagation in a penetrable wedge. The Journal of the Acoustical Society of America 126(1), EL22–EL27 (2009)
9. Kuperman, W.A.: Rapid computation of acoustic fields in three-dimensional ocean environments. The Journal of the Acoustical Society of America 89(1), 125 (1991), http://dx.doi.org/10.1121/1.400518
10. MPI Forum: Message Passing Interface (MPI) Forum Home Page. http://www.mpi-forum.org/ (Jan. 2017)
11. Open MPI: Open MPI: Open Source High Performance Computing Home Page. http://www.open-mpi.org/ (Jan. 2017)
12. Patterson, D.A., Hennessy, J.L.: Computer organization and design: the hardware/software interface. Newnes (2013)
13. Porter, M.B.: A numerical method for ocean-acoustic normal modes. The Journal of the Acoustical Society of America 76(1), 244 (1984), http://dx.doi.org/10.1121/1.391101
14. Porter, M.B.: Gaussian beam tracing for computing ocean acoustic fields. The Journal of the Acoustical Society of America 82(4), 1349 (1987), http://dx.doi.org/10.1121/1.395269
15. Porter, M.B.: Out-of-plane effects in ocean acoustics. Tech. rep., DTIC Document (2013)
16. Rodríguez, O.C., Collis, J.M., Simpson, H.J., Ey, E., Schneiderwind, J., Felisberto, P.: Seismo-acoustic ray model benchmarking against experimental tank data. The Journal of the Acoustical Society of America 132(2), 709–717 (2012)
17. Rodríguez, O.C.: The TRACEO ray tracing program. http://www.siplab.fct.ualg.pt/models/traceo/manual/index.html (date last viewed 15/04/2016)
18. Soares, C., Zabel, F., Jesus, S.M.: A shipping noise prediction tool. OCEANS 2015 - Genova (May 2015), http://dx.doi.org/10.1109/OCEANS-Genova.2015.7271539
19. Sturm, F., Korakas, A.: Comparisons of laboratory scale measurements of three-dimensional acoustic propagation with solutions by a parabolic equation model. The Journal of the Acoustical Society of America 133(1), 108–118 (2013)
20. Tolstoy, A.: 3-d propagation issues and models. Journal of Computational Acoustics 4(03), 243–271 (1996)
21. xi Xing, C., Song, Y., Zhang, W., xin Meng, Q., chun Piao, S.: Parallel computing method of seeking 3d eigen-rays with an irregular seabed. In: 2013 IEEE/OES Acoustics in Underwater Geosciences Symposium. pp. 1–5 (July 2013)