GDAŃSK UNIVERSITY OF TECHNOLOGY
Faculty of Electronics, Telecommunications, and Informatics

The author of the PhD dissertation: **Adam Kurowski**
Scientific discipline: **Information and communication technology**

DOCTORAL DISSERTATION

Title of the PhD dissertation: **Designing acoustic scattering elements using machine learning methods**

Title of the PhD dissertation (in Polish): **Projektowanie rozpraszających ustrojów akustycznych metodami uczenia maszynowego**

Supervisor	Second supervisor
signature	*signature*
prof. dr hab. inż. Bożena Kostek	
Auxiliary supervisor	Cosupervisor
signature	*signature*

Gdańsk, 2021

Designing acoustic scattering elements using machine learning methods

Doctoral Dissertation

Author: Adam Kurowski

Supervisor: prof. dr hab. inż. Bożena Kostek

Gdańsk, 2021

GDAŃSK UNIVERSITY OF TECHNOLOGY
Faculty of Electronics, Telecommunications, and Informatics

STATEMENT

The author of the PhD dissertation: Adam Kurowski

I, the undersigned, agree/~~do not agree~~* that my PhD dissertation entitled:
Automatic design of acoustic scattering devices employing machine learning algorithms may be used for scientific or didactic purposes.[1]

Gdańsk, 19.02.2021 .....

signature of the PhD student

Aware of criminal liability for violations of the Act of 4th February 1994 on Copyright and Related Rights (Journal of Laws 2006, No. 90, item 631) and disciplinary actions set out in the Law on Higher Education (Journal of Laws 2012, item 572 with later amendments),[2] as well as civil liability, I declare, that the submitted PhD dissertation is my own work.

I declare, that the submitted PhD dissertation is my own work performed under and in cooperation with the supervision of prof. dr hab. inż Bożena Kostek, ~~the second supervision, the auxiliary supervision, the cosupervision~~*.

This submitted PhD dissertation has never before been the basis of an official procedure associated with the awarding of a PhD degree.

All the information contained in the above thesis which is derived from written and electronic sources is documented in a list of relevant literature in accordance with art. 34 of the Copyright and Related Rights Act.

I confirm that this PhD dissertation is identical to the attached electronic version.

Gdańsk, 19.02.2021 .....

signature of the PhD student

I, the undersigned, agree/~~do not agree~~* to include an electronic version of the above PhD dissertation in the open, institutional, digital repository of Gdańsk University of Technology, Pomeranian Digital Library, and for it to be submitted to the processes of verification and protection against misappropriation of authorship.

Gdańsk, 19.02.2021 .....

signature of the PhD student

*) delete where appropriate.

1 Decree of Rector of Gdansk University of Technology No. 34/2009 of 9th November 2009, TUG archive instruction addendum No. 8.

2 Act of 27th July 2005, Law on Higher Education: Chapter 7, Criminal responsibility of PhD students, Article 226.

DESCRIPTION OF DOCTORAL DISSERTATION

The Author of the PhD dissertation: **mgr inż. Adam Kurowski**

Title of the PhD dissertation: **Designing acoustic scattering elements using machine learning methods**

Title of the PhD dissertation in Polish: **Projektowanie rozpraszających ustrojów akustycznych metodami uczenia maszynowego**

Language of the PhD dissertation: English

Supervision: prof. dr hab. inż. Bożena Kostek

~~**Second supervision***:~~
~~**Auxiliary supervision***:~~
~~**Cosupervision***:~~

Date of doctoral defense:

Keywords of the PhD dissertation in Polish: adaptacja akustyczna, ustroje akustyczne, dyfuzor, uczenie maszynowe, uczenie wzmacniane, FDTD (ang. *finite difference – time difference*)

Keywords of the PhD dissertation in English: acoustic room treatment, acoustic devices, diffuser, machine learning, reinforcement learning, FDTD (*finite difference – time difference*)

Summary of the PhD dissertation in Polish:

Streszczenie:

W procesie projektowania i korekcji akustyki wnętrz często zachodzi konieczność doboru odpowiedniego rodzaju ustrojów akustycznych oraz podjęcia decyzji dotyczących ich wielkości, geometrii oraz usytuowania w danym wnętrzu. Celem rozprawy doktorskiej jest opracowanie i walidacja modelu matematycznego pozwalającego przewidzieć efekty aplikacji ustroju rozpraszającego w wybranych punktach pomieszczenia i wykorzystanie tego modelu w komputerowej optymalizacji parametrów projektowanej adaptacji. Środkiem do tego celu są algorytmy uczenia maszynowego ze szczególnym wskazaniem na uczenie głębokie (ang. *deep learning*) i uczenie wzmacniane (ang. *reinforcement learning*). Podejście takie pozwala na uzyskiwanie projektów dyfuzorów akustycznych o szczególnie pożądanych własnościach. Jest to paradygmat przydatny zwłaszcza wówczas, gdy przeznaczeniem takiego dyfuzora jest przestrzeń o nietypowych charakterystykach, dla której nie sprawdza się standardowe rozwiązania projektowane z myślą o bardziej ogólnym sposobie zastosowania. Z tego właśnie powodu w niniejszej rozprawie zaproponowany został wątek automatycznej optymalizacji ustrojów akustycznych za pomocą programu komputerowego. Podejście to ma tę zaletę, że symulacja komputerowa pozwala na bieżąco śledzić przewidywane efekty zmian wprowadzanych do projektu dyfuzora. Zmiany te mogą być monitorowane dla przypadku ogólnego zastosowania dyfuzora – poprzez symulację warunków bezechowych, ale potencjalnie podejście to pozwala także na uwzględnienie w symulacji miejsca aplikacji i projektowanie dyfuzora akustycznego bezpośrednio do zastosowania w wybranym miejscu aplikacji.

Summary of PhD dissertation in English:

In the process of the design and correction of room acoustic properties, it is often necessary to select the appropriate type of acoustic treatment devices and make decisions regarding their size, geometry, and location of the devices inside the room under the treatment process. The goal of this doctoral dissertation is to develop and validate a mathematical model that allows predicting the effects of the application of the scattering system in selected points of the room. Further, it is aimed to use this model in the process of computer optimization of the room sound treatment process. The means for achieving these goals are machine learning algorithms with a particular focus on deep learning and reinforcement learning. Deep machine learning models are trained by using computer simulation employing the finite difference method (FDTD), which is used as a source of the so-called reward signal. The simulation model is based on the modified difference equations derived by the author of this thesis that allows simulating the behavior of acoustic diffusers in anechoic conditions. In order to reproduce this type of conditions, the results obtained by the reinforcement learning algorithms, in particular - by the deep policy gradient algorithm, were compared with the results obtained with the classical methods of designing acoustic diffusers and those obtained using another optimization method, i.e., genetic algorithms, where the numerical simulation of the behavior of the acoustic diffuser serves to calculate the value of the fitness function, which plays a role analogous to the reward function. The optimized property of acoustic diffusers is the autocorrelation diffusion coefficient. Numerical experiments have shown that optimization algorithms can be used to maximize the metrics computed by numerical simulation. The best results were obtained with the reinforcement learning algorithms. To validate the calculation results, the measurement of acoustic diffuser prototypes was also performed in the anechoic chamber. As a result of the measurements, the thesis confirmed that the algorithms resulting from computer optimization are characterized by more desirable parameters - the broadband autocorrelation diffusion coefficient and the band diffusion coefficients calculated for the bands with central frequencies with the values of 250 Hz, 500 Hz, 1 kHz, 2 kHz, and 4 kHz. The proposed algorithm is a new approach to the intelligent design of acoustic systems to improve room acoustic properties.

~~**Summary of PhD dissertation in language, in which it was written****:~~

~~**Keywords of PhD dissertation in language, in which it was written****:~~

*) delete where appropriate.

**) applies to doctoral dissertations written in languages other than Polish or English.

Streszczenie rozszerzone w j. polskim:

W procesie projektowania i korekcji akustyki wnętrz często zachodzi konieczność doboru odpowiedniego rodzaju ustrojów akustycznych oraz podjęcia decyzji dotyczących ich wielkości, geometrii oraz usytuowania w danym wnętrzu. Celem rozprawy doktorskiej jest opracowanie i walidacja modelu matematycznego pozwalającego przewidzieć efekty aplikacji ustroju rozpraszającego w wybranych punktach pomieszczenia i wykorzystanie tego modelu w komputerowej optymalizacji parametrów projektowanej adaptacji. Środkiem do tego celu są algorytmy uczenia maszynowego ze szczególnym wskazaniem na uczenie głębokie (ang. *deep learning*) i uczenie wzmacniane (ang. *reinforcement learning*). Podejście takie pozwala na uzyskiwanie projektów dyfuzorów akustycznych o szczególnie pożądanych własnościach. Jest to paradygmat przydatny zwłaszcza wówczas, gdy przeznaczeniem takiego dyfuzora jest przestrzeń o nietypowych charakterystykach, dla której nie sprawdza się standardowe rozwiązania projektowane z myślą o bardziej ogólnym sposobie zastosowania. Z tego właśnie powodu w niniejszej rozprawie zaproponowany został wątek automatycznej optymalizacji ustrojów akustycznych za pomocą programu komputerowego. Podejście to ma tę zaletę, że symulacja komputerowa pozwala na bieżąco śledzić przewidywane efekty zmian wprowadzanych do projektu dyfuzora. Zmiany te mogą być monitorowane dla przypadku ogólnego zastosowania dyfuzora – poprzez symulację warunków bezechowych, ale potencjalnie podejście to pozwala także na uwzględnienie w symulacji miejsca aplikacji i projektowanie dyfuzora akustycznego bezpośrednio do zastosowania w wybranym miejscu aplikacji.

Celem pracy jest zaproponowanie algorytmów, które na podstawie wyznaczonych celów dla optymalizowanych metryk opisujących dyfuzory akustyczne są w stanie zaproponować projekty dyfuzorów akustycznych Schroedera, które mają potencjał do ich wykorzystania w praktycznym przypadku aranżacji akustycznej wnętrza. Aby sformalizować ten cel, zdefiniowane zostały następujące trzy tezy, które następnie zostały udowodnione w dalszej części rozprawy:

1. **Możliwe jest zastosowanie symulacji numerycznej jako metody predykcji współczynnika dopasowania (*ang. fitness*) lub nagrody (*ang. reward*), wykorzystywanych przez algorytmy sztucznej inteligencji do optymalizacji konstrukcji dyfuzora Schroedera.**

2. **Możliwe jest wykorzystanie metod uczenia maszynowego, takich jak algorytmy genetyczne lub głębokie uczenie wzmacniane (*ang. deep reinforcement learning*), do optymalizacji geometrii dyfuzora Schroedera w celu uzyskania projektu o pożądanych właściwościach akustycznych współczynnika autokorelacji dyfuzji, gdy pomiar jest wykonywany w warunkach bezechowych.**

3. **Przygotowane geometrie akustycznych elementów rozpraszających uzyskane w wyniku optymalizacji komputerowej są możliwe do wykorzystania w praktycznej realizacji i mogą służyć do modyfikacji i poprawy akustyki pomieszczeń.**

W celu uproszczenia późniejszego etapu przygotowania fizycznych prototypów pomiarowych założono, że badane będą dyfuzory Schroedera bez przegród pomiędzy jego wgłębieniami. Trening modeli głębokiego uczenia maszynowego przedstawiony w rozprawie odbywa się poprzez wykorzystanie symulacji komputerowej metodą różnic skończonych (FDTD, *finite-difference time-domain*), która wykorzystywana jest jako źródło tzw. funkcji dopasowania i sygnału nagrody. Symulacje metodą FDTD są szeroko wykorzystywane w przypadku predykcji rozchodzenia się fal elektromagnetycznych, ale z powodzeniem są one także stosowane w symulacjach przewidujących propagację fal akustycznych. Źródłem metryk optymalizowanych przez zaproponowane w rozprawie algorytmy jest wynik symulacji komputerowej. Sposób wyliczania metryk na podstawie efektów symulacji może się różnić w zależności od tego, jaki dokładnie typ symulacji jest przeprowadzany. W badaniach prowadzonych na potrzeby niniejszej rozprawy miały miejsce dwa eksperymenty i

wykorzystywały one dwa różniące się między sobą sposoby wyliczania funkcji dopasowania i sygnału nagrody.

Podstawą pierwszego eksperymentu jest symulacja w prostym pomieszczeniu o prostokątnej geometrii. Wymiary pomieszczenia wynoszą odpowiednio: 3 m długość i 2 m szerokość oraz wysokość. W praktyce może się to odnosić do pomieszczenia, które pełni funkcję reżyserki w studiu nagraniowym. Podejście polegające na symulowaniu zamkniętego obszaru o tak prostej geometrii ma istotną zaletę, jaką jest wykorzystanie do obliczeń zdyskretyzowanych równań różniczkowych sformułowanych bezpośrednio w takiej postaci, w jakiej przedstawiane są one w szerokiej literaturze opisującej problematykę symulacji propagacji fal akustycznych metodą FDTD [Botts2014, Cox2017, Hamilton2017, Webb2011]. Miarą wykorzystaną do oceny jakości projektów dyfuzora Schroedera w tym scenariuszu eksperymentalnym była jednorodność charakterystyki częstotliwościowej dyfuzora w wybranym docelowym miejscu odsłuchu muzyki. Jako dane wzorcowe, względem których oceniane były efekty uzyskane przez algorytmy optymalizacji, wykorzystane zostały dyfuzory Schroedera zaprojektowane metodami opartymi na sekwencjach liczb pseudolosowych. Wykorzystano dyfuzory bazujące na residuum kwadratowym (*ang. quadratic residue diffuser, QRD*) i na pierwiastku pierwotnym (*ang. primitive root diffuser, PRD*). Jako algorytmy optymalizacji wykorzystane zostały wspomniane już wcześniej algorytmy genetyczne oraz dwa algorytmy uczenia wzmacnianego – głębokie, dwudzielne Q-sieci (*ang. deep duelling Q-networks, DDQNs*) i algorytm głębokiego gradientu strategii (*ang. deep policy gradient*). Efektem przeprowadzonego eksperymentu badawczego jest wniosek, że uzyskane mediany współczynnika równomierności odpowiedzi impulsowej badanego pomieszczenia były najkorzystniejsze dla dyfuzorów PRD oraz dyfuzorów optymalizowanych. Nie było istotnych statystycznie różnic między medianami dyfuzorów PRD i dyfuzorów będących efektem optymalizacji. Istotnym faktem było jednak to, że dla każdego przypadku optymalizacji można było wyróżnić najlepszy projekt dyfuzora, który był lepszy od dyfuzora uzyskanego algorytmem PRD. **Wniosek pozwala na udowodnienie dwóch pierwszych tez rozprawy. Z sukcesem zaimplementowany został algorytm, który wykorzystuje symulację komputerową metodą FDTD do optymalizacji kształtu dyfuzorów Schroedera, co udowadnia tezę nr 1 postawioną w niniejszej rozprawie doktorskiej. Dodatkowo, optymalizacja ta możliwa była do przeprowadzenia za pomocą algorytmów genetycznych oraz algorytmów wykorzystujących głębokie uczenie wzmacniane, co z kolei dowodzi prawdziwości tezy nr 2.**

Drugi eksperyment bazuje na symulacji propagacji fal akustycznych w warunkach bezechowych. Aby przygotować tego typu algorytm symulacji, konieczne było zmodyfikowanie równań, które były podstawą symulacji propagacji fal akustycznych w eksperymencie pierwszym. Zastosowana została autorska modyfikacja równań Webba i Bilbao [Webb2011], które były podstawą symulacji w eksperymencie pierwszym. Umożliwiło to posłużenie się warstwami idealnie dopasowanymi Berengera (ang. *perfectly matched layers*, PML) do redukcji odbić od granic domeny obliczeniowej, co z kolei pozwoliło na symulację pomiaru autokorelacyjnego współczynnika dyfuzji projektowanych dyfuzorów. Ten właśnie współczynnik optymalizowany był przez algorytmy uczenia wzmacnianego, które okazały się skuteczne w eksperymencie 1. Na podstawie wyników tego eksperymentu wybrane zostały najlepsze dyfuzory zaprojektowane przez algorytm genetyczny i jeden algorytm uczenia wzmacnianego – algorytm głębokiego gradientu strategii. Ze względu na ograniczenia technologiczne wynikające z rozmiaru przestrzeni decyzyjnej w eksperymencie 2., konieczna była rezygnacja z algorytmu głębokich, dwudzielnych Q-sieci. Efekty działania algorytmów optymalizacji porównane zostały z wynikami losowania kształtu dyfuzora i wyboru najlepszego wyniku losowania. Algorytmem, który był w stanie osiągnąć najwyższą wartość autokorelacyjnego współczynnika dyfuzji był algorytm głębokiego gradientu strategii. Na podstawie najlepszych dyfuzorów zaprojektowanych przez algorytm genetyczny, głębokiego gradientu strategii i poprzez wybór najlepszego wyniku losowania przygotowane zostały fizyczne prototypy dyfuzorów. Prototypy

te wykonane zostały ze styroduru i następnie zmierzone w komorze bezechowej. Wyniki pomiarów wykazały, że prototypy dyfuzorów charakteryzowały się współczynnikiem dyfuzji o wartościach zaniżonych względem tych przewidywanych o wartość z przedziału ufności od 0.0234 do 0.123. **Przy przewidywanych wartościach współczynnika dyfuzji wynoszących około 0,95 taka dokładność pozwala na udowodnienie trzeciej tezy postawionej w rozprawie doktorskiej. Mówi ona o tym, że projekty dyfuzorów akustycznych wygenerowane przez zaproponowane algorytmy optymalizacji nadają się do praktycznej implementacji w procesie faktycznej adaptacji akustycznej pomieszczeń.** Dodatkowo, w wyniku pomiarów zaobserwowano, że algorytmy będące efektem optymalizacji komputerowej charakteryzują się bardziej pożądanymi parametrami – szerokopasmowym autokorelacyjnym współczynnikiem dyfuzji oraz pasmowymi współczynnikami dyfuzji wyliczonymi dla pasm o częstotliwościach centralnych 250 Hz, 500 Hz, 1 kHz, 2 kHz i 4 kHz.

Zaproponowany algorytm stanowi nowe podejście do inteligentnego projektowania ustrojów akustycznych służących do poprawy własności akustycznych pomieszczeń. Na podstawie pomiarów w komorze bezechowej udało się pokazać, że dyfuzory zaprojektowane zaproponowaną metodą mogą stanowić przydatną alternatywę dla dyfuzorów projektowanych technikami tradycyjnymi. Jest to główny efekt badań przeprowadzonych w ramach rozprawy doktorskiej, jednak należy także wspomnieć o innych autorskich osiągnięciach, które pozwoliły na udowodnienie głównych tez rozprawy. Są to następujące osiągnięcia:

- modyfikacja równania różnicowego FDTD (*finite-difference time-domain*, tj. metodą różnic skończonych), pozwalająca na wykorzystanie warstw idealnie dopasowanych PML (ang. *perfectly matched layers*) Berengera w symulacjach FDTD, do którego punktem wyjściowym była praca Webba i Bilbao [Webb2011];
- propozycja modelu matematycznego wykorzystującego metodę FDTD, który jest w stanie obsłużyć warstwy PML i uwzględnić takie właściwości ośrodka propagacji, jak temperatura powietrza, ciśnienie atmosferyczne i wilgotność względna;
- propozycja metod uczenia maszynowego wykorzystujących symulację FDTD jako źródło sygnału nagrody. W ten sposób uzyskuje się możliwość projektowania dyfuzorów akustycznych z maksymalizacją wartości współczynnika dyfuzji akustycznej w kontekście zadanego środowiska symulacji;
- autorska implementacja proponowanej zmodyfikowanej wersji metody symulacji FDTD w języku Python;
- autorska implementacja symulacji opartej na FDTD jako estymatora funkcji przystosowania algorytmu genetycznego do maksymalizacji pożądanych właściwości dyfuzora akustycznego;
- przeprowadzona została ocena prototypów dyfuzorów akustycznych tworzonych przez algorytmy uczenia maszynowego (tj. uczenia wzmacnianego – głębokie, dwudzielne Q-sieci (*ang. deep duelling Q-networks, DDQNs*) i algorytm głębokiego gradientu strategii (*ang. deep policy gradient*) w warunkach rzeczywistych z wykorzystaniem prototypów fizycznych, których właściwości akustyczne mierzono w komorze bezechowej;
- autorska metoda oceny prototypów dyfuzorów akustycznych tworzonych przez algorytmy uczenia maszynowego (tj. dwa algorytmy uczenia wzmacnianego – głębokie, dwudzielne Q-sieci (ang. *deep duelling Q-networks*, DDQNs) i algorytm głębokiego gradientu strategii (ang. *deep policy gradient*) w warunkach rzeczywistych z wykorzystaniem prototypów fizycznych, których właściwości akustyczne mierzono w komorze bezechowej.

Wymienione osiągnięcia mogą stanowić podstawę dla dalszych badań, które w większym stopni obejmą przedstawienie rzeczywistego pomieszczenia w symulacji komputerowej, co pozwoliłoby na projektowanie dyfuzorów dopasowanych do konkretnego wnętrza.

Innym interesującym kierunkiem dalszych badań może być projektowanie układów zmiennych w czasie, jako że symulacja metodą FDTD jest w stanie również przewidywać efekty takich układów. Mogłoby to być bardzo istotnym przyczynkiem do badań na przykład nad aktywnymi dyfuzorami akustycznymi zrealizowanymi w postaci macierzy głośników, których impedancja akustyczna jest zmienna w czasie. Interesującym tematem przyszłych badań może też być modyfikacja zaproponowanych metod w celu zastosowania ich do projektowania innych typów dyfuzorów.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Prof. Bożena Kostek, for all the substantive guidance, support, and pieces of advice, which made this work possible.

I want to thank Prof. Andrzej Czyżewski for all the professional and personal growth opportunities I encountered during the period of preparation of my doctoral thesis.

I want to thank all my colleagues at Gdańsk University of Technology, who supported me through the journey consisted in carrying out experiments and preparation of this thesis. This goes especially to Szymon Zaporowski - for all the pieces of advice; thanks to whom I was always sure that my long-lasting computations are being carried out on the deep learning workstations, even during the "dark time" of the COVID-19 pandemic.

Finally, I want to thank my family for their support and unconditional trust in me. Especially, I want to thank:

- *my Mom, Małgorzata, for all her support and a word of encouragement even in moments of failure,*
- *my Brother, Mariusz, for all the discussions and pieces of advice on all topics related to computer science,*
- *and my Uncle Zenon, whose pieces of advice regarding the practical side of my research, namely – the preparation of diffuser prototypes, were priceless and always on-point.*

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

a	single action belonging to the action space A
A	action space of the DDQN (dueling deep Q-network), and DPG (deep-policy gradient) algorithms
A_s	amplitude of an excitation Gaussian impulse signal
ADAM	adaptive movement estimation optimizer
ANN	artificial neural network
API	application programming interface
BEM	boundary element method
$\mathbf{BK}$	matrix containing pre-computed values of boundary conditions taking into account a Courant number, and specific admittances of surfaces
c	speed of sound in an acoustic medium (air)
C_T	clarity acoustic room parameter
CAD	computer-aided design
CNN	convolutional neural network
CUDA	compute unified device architecture
d_ψ	correlation diffusion coefficient of an acoustic diffuser
$d_{\psi,n}$	normalized diffusion coefficient
$d_{\psi,r}$	reference measurement of a flat reflective plate
DDQN	dueling deep Q-network
DL	deep learning
DPG, PG	deep-policy gradient, policy gradient
DQN	deep Q-network
$\mathrm{E}(r)$	expected reward of a reinforcement learning agent
EDT	early decay time
f_0	design frequency of a Schroeder diffuser

f_c	maximum frequency of a band-limited excitation Gaussian impulse
f_{ub}	upper band limit of an acoustic diffuser
FDTD	finite-difference time-domain method
FEM	finite element method
FFT	fast Fourier transformation
GPU	graphics processing unit
GRAM	geometrical room acoustic model
$\mathrm{h}(t)$	impulse response of the room
$\boldsymbol{K}$	matrix containing a definition of all boundary conditions in a computational domain
L	acoustic pressure level
LEDE	light-end, dead-end acoustic treatment principle
M_a	molar mass of dry air
m_a	amount of moles of air present in the propagation medium
M_v	molar mass of water vapor
m_v	amount of moles of water vapor present in the propagation medium
MLS	maximum-length sequence
P	atmospheric air pressure
p	acoustic pressure
p_{sat}	saturation pressure of water vapor
p_a	partial pressure of air
p_v	partial pressure of water vapor
PML	perfectly matched layer
PRD	primitive root diffuser
PReLU	parameterized rectified linear unit activation function

Q	Q function used by the DDQN reinforcement learning algorithm
QRD	quadratic residue diffuser
R	molar gas constant
$\mathrm{R}(\theta, \phi)$	acoustic reflection coefficient
R_a	specific gas constant of air
r_t	reward achieved by the agent in the step of the interaction with the environment denoted by t
R_v	specific gas constant of water vapor
RFZ	reflection-free zone
RH	relative humidity
RL	reinforcement learning
RT	reverberation time
s	state of an environment with which the reinforcement learning agent is interacting
S_n	surface area of the *n*-th room boundary
SGD	stochastic gradient descent
T	temporal sampling period
T_a	air temperature
T_i	integration time
$\boldsymbol{u}$	acoustic velocity vector
USP	ultimate sound probe
$\mathrm{V}_{\mathrm{room}}$	volume of the room
w	weights of a neural network
$W_d(n)$	*the n*-th value from a pseudorandom sequence (i.e. PRD, QRD, etc.)
X	spatial sampling period

x_a	mole fraction of air in the propagation medium
x_v	mole fraction of water vapor
XPS	extruded polystyrene
Y_k^{DDQN}	value of Q function for the *k*-th action undertaken by the DDQN algorithm,
Z_{air}	specific acoustic impedance of air
Z_w	specific acoustic impedance of a surface
α	acoustic compressibility-related attenuation factor
α^*	attenuation factor related to a so-called “mass-proportional” damping
$\alpha_{room,n}$	acoustic damping factor of n-th surface in a room, used in Sabine equation
β	specific acoustic admittance of the surface
δ_c	correlation scattering coefficient
ϵ	probability of taking a random choice of action in an ϵ-greedy reinforcement learning algorithm
κ	adiabatic ratio of specific heats of air
λ_{min}	the shortest wavelength for which no lateral propagation modes arise in the diffuser well (limiting the upper frequency of usable diffuser bandwidth)
λ_0	wavelength of a diffusers design frequency
∇	nabla operator
$\pi_w(s,a)$	policy function
ρ	density of the air

σ	variance of a Gaussian impulse
$\sigma_{f\in<f_{li};f_{ui})}$	measured power level of the frequency band from f_{li} to f_{ui}
σ_D	frequency response uniformity reward coefficient
γ	discount factor of a reinforcement learning agent
κ	compressibility factor of the air
λ	Courant number
ϕ	vertical wave incidence angle
θ	horizontal wave incidence angle
ξ	prime number used to generate a pseudo-random sequence
ζ	compressibility factor of air

TABLE OF CONTENTS

1. INTRODUCTION

Proper sound treatment of open and enclosed spaces is a broad field of knowledge, which often requires employing a number of methods used to shape frequency responses of both the open and closed spaces used for purposes such as theaters, public speech places, or lecture halls [Gołaś2012]. Similar methods that propose an alteration of sound propagation are also applied to other areas of application, such as noise and vibration control or medical imaging [Adamczyk2011, Leniowska2012]. Often, they are also employing modern, state-of-the-art algorithms which are based on artificial intelligence [Grochowina2020, Kurowski2020]. The use of such methods can be especially effective if new emerging computational techniques are taken into account, such as numerical modeling of acoustic waves propagation, artificial intelligence-based optimization, and control algorithms. Acoustic simulations have a very broad use in acoustics in tasks such as auralization [Hamilton2017], noise prediction [Ibrahim2020], investigation of the acoustics of historic buildings [Berardi2020], a simulation-driven optimization of room acoustics, or optimization of acoustic devices [Li2016, Pilch2020, Su2020, Sun2020]. Since the rise of the popularity of GPU-based computations, acoustic simulations are also easier to be carried out, as the GPU-accelerated calculation can be employed [Cao2020, Webb2011]. Machine learning methods using deep structures are a rapidly growing field of knowledge, with multiple applications in areas of expertise such as EEG signals processing [Fernandez-Blanco2020, Gao2020, Hemanth2020, Kurowski2019a, Kurowski2019b, Luo2020] or even for obtaining results associated with “creative” tasks [Thoma2016]. In the field of acoustics, artificial-intelligence or machine learning algorithms were successfully implemented to perform tasks such as speaker and source localization [Bianco2019], audio rendering [Tang2020], analysis of acoustic signals originating from urban-related sources [Huang2020, Kurowski2019c, Naranjo-Alcazar2020, Shen2020], or acoustic-based terrain classification [Valada2018].

In this thesis, techniques involving computer simulation and deep reinforcement-learning-based optimization are applied to one of the devices commonly used to shape acoustics of rooms undergoing the process of acoustic treatment. This concerns the acoustic diffusers. Acoustic diffusers, also called acoustic scattering devices or devices, are a crucial type of tool used in the sound treatment of rooms. There is a great need to obtain designs of acoustic diffusers with a precise set of acoustical properties that can be used to shape the response of enclosed space under treatment. This applies to a theatre, concert hall, lecture room, recording, radio or television studios, etc. Therefore, there are many approaches for obtaining acoustic diffuser geometries, which may be further used in the acoustic room treatment. Baseline (also called classical or traditional) acoustic diffuser design techniques have a number of advantages that influenced their popularity in everyday practice. These include, among others, simplicity of use or broad applicability with satisfactory results. Based on such devices, the solution generated by the optimization algorithm may allow for complete automation or significant reduction of human interference in the design of devices used to improve the acoustics of rooms. It also makes it possible to tailor the solution to

specific application conditions.

The purpose of the research presented in this dissertation is to propose a solution that creates an acoustic diffuser architecture based on the optimization process. Acoustic diffusers utilize the phenomenon of acoustic wave scattering to increase sound diffusion at the place of their application [Cox2017]. Their design is a crucial stage in preparing the room entire acoustic adaptation, especially when they are the primary method of preserving the reflected sound in the room. This concerns especially the case when the room is divided into two zones – one with a significant amount of acoustic wave attenuation (the so-called “dead”) and one associated with a large amount of reflected acoustic waves (the so-called “live”). This type of methodology is called LEDE; an acronym derived from live-end dead-end [Davies1980, Gervals2011].

On the one hand, the introduction of reflected sound may be beneficial. It can enhance such aspects of audio perception as the feeling of being enveloped by the sound and is often used creatively by recording engineers [Owsinski2009, Senior2015]. On the other hand, the presence of reflections may lead to the occurrence of unwanted phenomena such as flutter echo [Everest2015, Makarewicz2004], or standing waves which cause variance in both the acoustic pressure level of the audio signal and relative levels of frequency band powers of the transmitted acoustic signal. This may result in the so-called “sound coloration” [McCarthy2016, Zolzer2011]. Acoustic diffusers are often applied to preserve reflected energy in a room in a scattered form, which does not introduce the aforementioned interference-related problems [Everest2015]. In the part of the room accountable for reflections, an acoustic diffuser should very often be placed to prevent, for example, harmful interference from the occurrence of acoustic waves, which makes the frequency characteristics of the room non-linear and nonuniform. Appropriate use of acoustic diffusers also allows for equalizing the room reverberation time over a wide frequency range.

For this reason, the possibility of choosing an acoustic diffuser in such a way that its design method takes into account its placement within the room of application is important. It can contribute to improving the results obtained. Such precision is vital for spaces dedicated to audio editing and audio quality control, such as recording and mastering studios or theatre halls [Gołaś2009, Katz2007, Owsinski2017a, Owsinski2017b, Wyner2013]. In such kinds of applications, a carefully planned use of sound absorption is necessary [Cucharero2019, Rivet2012]. For this reason, an interesting perspective is using an optimization algorithm to design the diffuser geometry matched to the rooms where they are to be implemented.

Another critical problem in the design of acoustic diffuser geometry matched to the geometry of the rooms is the modeling of acoustic phenomena occurring in the place of their application. This problem can be solved using numerical simulation techniques, which are widely adopted within acoustics and related areas such as noise control or vibroacoustic. Examples of such acoustic-related problems are the simulation of musical instruments with boundary element method (BEM) [Becache2005] or time-difference method [Derveaux2003], simulation and elimination

of noise sources [Oberst2010, Yuksel2012], calculation of head-related transfer functions for binaural audio [Fiala2010, Kahana2007], modal analysis of, i.e., loudspeaker enclosures and vibration reduction [Hansen2018, Jee2000] or prediction of acoustic transmission in closed spaces [Hsiao2011a, Hsiao2011b]. Such numerical methods may also be utilized to simulate the influence of given acoustic diffuser designs on the acoustics of a room and then calculate the geometry of diffusers tailored to improve the acoustic properties of such a place. One of the popular simulation techniques employed for such purpose is the finite-difference time-difference (FDTD) method, which is also commonly used for simulation of electromagnetic waves behavior [Inan2011]. It is also widely used in research related to the propagation and other phenomena related to acoustic waves [Cox2017, Drake2014]. It allows the prediction of the performance of the diffuser at the place of its application. It also provides an approximate assessment of the behavior of individual diffuser geometries and the selection of the most promising designs. On this basis, it is possible to develop physical prototypes of sound scattering devices, which then may be measured in the anechoic chamber and in the target room where they are to be placed. Computer simulation allows the number of prototypes to be tested without the need for physical prototype construction. This, in turn, allows reducing the cost of prototype manufacturing because only the most promising designs are evaluated in an anechoic chamber or in-situ physical measurement.

A variety of techniques are employed to design diffusers, from classical approaches based on pseudo-random number sequences to methods using fractal geometry [Everest2015]. They can also use as a principle of their operation not only the phenomenon of reflection, deflection, and scattering of the wave, but also resonance phenomena [Cox2017, Jimenez2017, Pogson2010]. This means that to design the acoustic diffuser, the environment in which it is to be installed should be taken into account. It is necessary to carefully select a methodology that will consider a whole range of factors that affect the behavior of the diffuser. In addition to the device geometry itself, such factors are of importance as, for example, the material from which such a diffuser is made, the place where the diffuser is to be installed, the geometry of the room in which it is located, or the place where the person listening to the sound in a given room will be. Therefore, optimization algorithms in the form of, e.g., genetic or reinforcement learning algorithms are an attractive way to solve the problem of finding the acoustic diffuser geometry that optimizes listening parameters in a given context, which consists of the previously mentioned factors.

There are many examples of the application of the genetic algorithm in acoustics as it is a method of optimizing structures affecting the structure of the acoustic field both inside the rooms (acoustic diffusers) and in the open space (noise barriers) [Baulac2008, Patraquim2017, Redondo2019, Toledo2015]. Coupling computer simulation with the method of designing acoustic systems is a scheme that has already appeared in the literature and has been used, for example, to assess the diffusion coefficient of acoustic diffusers [Patraquim2017, Redondo2007]. As a simulation method, various methods are used; for example, the edge element method

[Redondo2007] or the finite difference method [Patraquim2017]. This doctoral dissertation shows how to design an acoustic diffuser that equalizes the frequency response of a cuboid shape room using a computer simulation assessing the genetic algorithm associated with it.

Another optimization approach may be the use of machine learning algorithms. In recent years, machine learning was applied to a variety of acoustic-related problems such as real-time speech synthesis [Arik2017], sound synthesis [Blaauw2017, Engel2017], audio chord recognition [Boulanger-Lewandowski2013], music transcription [Sturm2016], or sound source separation [Huang2015]. Among many machine learning algorithms, one class is particularly useful in sequential optimization problems, i.e., reinforcement learning and its modification employing deep neural networks – deep reinforcement learning [Buduma2017, Sutton1998]. This class of algorithms was successfully applied to complex problems involving decision-making and multi-step optimization such as the design of industrial infrastructure [Kobayashi2004], control of modular robots [Varshavskaya2008], lane-keeping assists [Sallab2016], playing card games [Yeh2018], or real-time computer strategy and survival games [Romac2019, Usunier2016, Vinyals2017]. Accordingly, for an acoustic diffuser structure optimization, which is a sequential optimization problem, deep reinforcement learning is to be applied.

Therefore, the purpose of the research presented in this doctoral thesis is to investigate if it is possible to employ a reinforcement learning algorithm to design a Schroeder diffuser. A Schroeder diffuser may be described by a matrix of integer values. So, according to the above-given definition, the shape of a diffuser stored in such a matrix is referred to as a diffuser geometry in successive parts of this doctoral thesis. Moreover, it is relatively easy to prepare a physical prototype of such a diffuser, especially if the skyline type of diffusers is considered. For this reason, in further experiments, the skyline type of Schroeder is chosen for simulations and measurements. Outcomes obtained from the reinforcement learning algorithm designing a Schroeder diffuser can be validated by a physical measurement performed on a real prototype.

Three theses proposed, which are going to be proved in this dissertation, are as follows:

1. **It is possible to employ a numerical simulation as a fitness or reward estimator used by an artificial intelligence algorithm to optimize a Schroeder diffuser design.**
2. **It is possible to employ machine learning methods such as genetic algorithms or deep reinforcement learning for optimization of the Schroeder skyline diffuser geometry to achieve a design having desired acoustic properties of autocorrelation diffusion coefficient when the measurement is performed in an anechoic condition.**

3. **Prepared geometries of acoustic scattering elements acquired in the computer-based optimization are feasible for practical implementation and can be used as the means of acoustic room treatment.**

The experiments presented in successive parts within this thesis are performed to check if assumptions shown in the theses proposed above are correct. Moreover, it should be checked whether the reinforcement learning-based algorithm applied to design acoustic diffusers can outperform other state-of-the-art solutions. To that end, two benchmark solutions were selected for comparison. The first relies on classical random sequence theories, and the second one is optimization employing a genetic algorithm. The block diagram presenting the structure of this thesis is shown in Fig. 1.1.

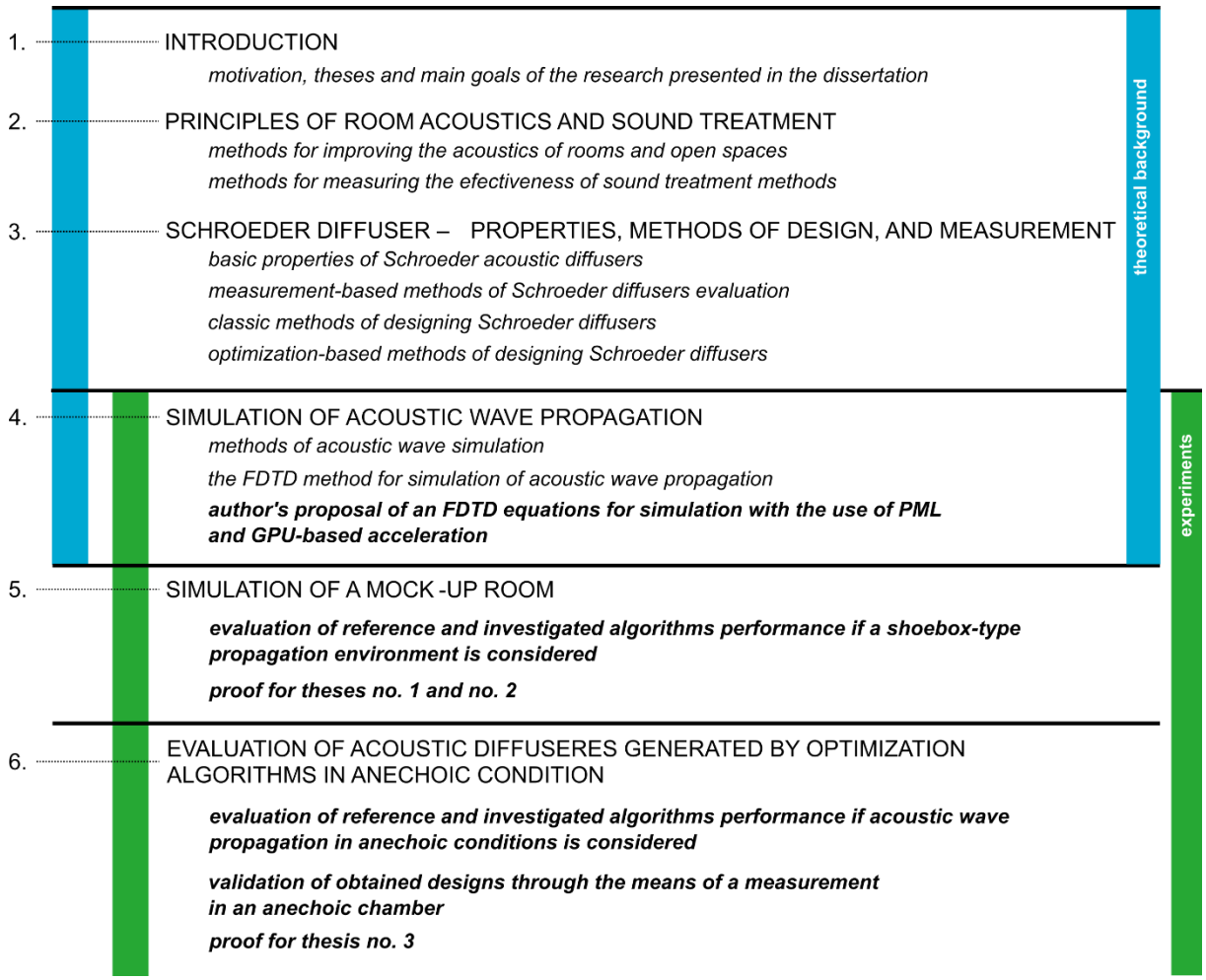


Fig. 1.1 Block diagram of this doctoral dissertation.

An introduction presents the motivation, theses, and main goals of the research presented in this thesis.

Section 2 introduces the associated acoustic treatment of open and closed spaces. The main focus is on the acoustic treatment of closed spaces, as this is the purpose of using acoustic diffusers. This Section also presents problems associated with the simulation of acoustic wave propagation and the use of such simulation to design

acoustic treatment solutions. Also, it provides a rationale for why an FDTD simulation technique was used as a simulation technique in experiments presented in this doctoral dissertation.

Next, Section 3 describes methods of designing acoustic diffusers. Its main purpose is to present necessary methods for the evaluation of Schroeder diffuser designs which are necessary for definition optimization goals. For instance, an autocorrelation diffusion coefficient introduced in Section 3.1 is a necessary tool for evaluating the quality of devices designed by algorithms in Sections 3.3 and 3.4. It also contains descriptions of classical methods of Schroeder diffuser design and optimization-based methods, namely:

1. the genetic algorithm-based method,
2. a dueling deep Q-network, which is the first of deep reinforcement learning algorithms evaluated in this dissertation,
3. and a deep policy gradient algorithm, which is the second of deep reinforcement learning algorithms evaluated in this dissertation,

Section 4 contains the mathematical foundation of the implementation of the FDTD method used in this study. It describes the original method proposed by [Webb2011]. Moreover, it also shows the author's original contribution in the form of a modification of the FDTD model mentioned above, which allows to carry out simulations with GPU-based acceleration and the use of Berenger perfectly matched layers to simulate anechoic acoustic wave propagation. This Section also contains a description of how the temperature, atmospheric pressure, and relative humidity of air could be taken into account while carrying out simulations.

Section 5 contains a description of the first experiment conducted to prove theses of this doctoral dissertation. The investigation presented in this Section is based on calculations employed to design acoustic diffusers intended to be used in a shoebox-type mock-up room. A measure optimized in this case is the uniformity of the frequency response of a simulated room. The outcome of this experiment allows proving theses no. 1 and no. 2 proposed in this dissertation.

Section 6 contains a description of the second experiment, which involved simulation of acoustic diffuser behavior in anechoic conditions. A measure used to evaluate designs in this experiment is the autocorrelation diffusion coefficient. Designs obtained from this optimization process were further used to prepare physical prototypes, which were measured in an anechoic chamber. Outcomes from these experiments allow proving theses no. 1-3 proposed in this doctoral dissertation. Section 7 shows concluding remarks, main achievements, and directions of future research. This Section is followed by the literature sources. Finally, examples of code prepared by the author and detailed results are contained in Appendices.

2. PRINCIPLES OF ROOM ACOUSTICS AND SOUND TREATMENT

Room acoustics is an area that contains many concepts and notions. Generally speaking, it describes how sound behaves in an enclosed space. In this Section, only selected issues related to the practical part of the dissertation are to be presented. Mainly, the focus is on the process of acoustic room treatment. However, acoustic parameters that are important when checking room acoustics before and after it is treated acoustically are also referred to. Besides, room designing methods in the context of a listening space are shortly recalled.

Sound treatment is a process that usually splits into three stages. First, the analysis of the acoustic space in which acoustic properties have to be improved is performed. Secondly, identification of critical problems, design, and choice of tools that can be used to reduce the impact of those problems on the acoustic of the room is to be discussed. Finally, implementation of the prepared solution and verification of its effectiveness should take place.

There is a group of typical defects and problems related to room acoustics that every designer should consider when implementing acoustic adaptation. In the case of the, i.e., a mixing room, sound reflections are an essential aspect. During work, the sound engineer hears the sound coming directly from the monitors as well as the sound reflected in the room. Along with the size of the interior, the time difference between the direct and reflected signals increases. However, in most cases, the work of engineers takes place in medium or relatively small interiors, so reflections are usually a big problem [Gervals2011]. Therefore, the task of the designer is to absorb or scatter these reflections to such an extent as to minimize their impact on listening from the speakers. For this purpose, acoustic panels (absorbers) and diffusers are used.

Another issue is the room resonances, also called “room modes.” They are particularly troublesome in small interiors. They manifest themselves when the acoustic wave travels between two reflecting surfaces and the distance between them (one of the dimensions of the room) is equal to a multiple of half the length of that wave [Cox2017]. Such sound bounces off these surfaces and goes the same way, but in the opposite direction, encountering another wave on its way. In this situation, the sound energy is concentrated or attenuated locally, and thus the sound field is nonuniform. This results in an unnatural, pronounced sound of selected tones in the one place in the room or their complete absence elsewhere. It is particularly noticeable in cubic or rectangular rooms. In this case, resonance will often occur between all pairs of parallel reflecting surfaces.

Reproduction of audio signals is also a complicated process that is performed by a chain of many elements. First, it is a sound source, an acoustic source such as a musical instrument or a loudspeaker that converts analog signals into acoustic signals. Analog signals also may be a reproduction generated from digital sources of audio signals. The type of data storage, therefore, also has its impact on the quality of audio

reproduction in the room of choice. The quality of listeners' experience is influenced by the quality of the source material recorded on the medium used for playback of signals or on the quality of performance of musicians or other sources of musical signals. In the case of electroacoustic reproduction of audio signals, the quality of devices such as amplifiers, loudspeakers, or players also plays an important factor in the quality of the listener's experience. The room is the second element of this chain of factors and has its influence on the overall perception of audio signals and is as important as good performance of musicians or good quality of source material stored on the medium and devices used for the sound reproduction [Owsinski2017a, Owsinski2017b, Wyner2013]. The environment influences the sound in a number of ways in a particular area where the listener is located. First of all, the room is a closed space; therefore, it is associated with the so-called room modes. Such modes are resonant frequencies of a room cavity and lead to the phenomenon of standing waves and may impact the frequency response of the room. It may also alter transients, as resonances of the room may cause accumulation of energy in frequency bands associated with modes of the listening room. Specular reflections may cause confusion when stereo imaging is considered. Due to the precedence effect [Brown2015], a strong reflection may alter the perception of a sound direction of arrival and, therefore, change the stereo imaging in a particular place of the room. A reflection-free-zone (RFZ) is a concept developed to mitigate this unwanted phenomenon and is often used by acousticians to ensure that for a given place, there will be no strong reflections capable of skewing the stereo imaging of acoustic signals played.

One of the popular methods of designing rooms used for purposes related to audio reproduction or audio editing is a live-end dead-end (LEDE) approach [Davies1980, Gervals2011]. As already mentioned, it is based upon a division of the room space into two sub-areas. The first one is treated with acoustic absorbers. This part of a room is called a dead end. Due to such treatment, from this end of the room, only a direct sound is produced. The other part of the room has reflective surfaces, which provide reflections necessary to give a listener sense of being enveloped by a sound. This is called a live end. However, the sound at the live end should be scattered in addition to being simply reflected. This difference in scattering and the so-called specular reflection may define if the room provides a satisfactory frequency response. Specular reflections may cause constructive and destructive interferences. They may alter the frequency response of the room due to the occurrence of standing waves. They may modify both the frequency content of acoustic waves arriving at the given point of the room. Still, they can also alter phase, which additionally to degrading the timbre of perceived sound may also change a stereo image of a played recording.

Even if all precautions and methodology were applied to the design of the room, it should be validated by measurement of metrics. Such metrics permit assessment of its quality with respect to factors such as subjective evaluation of sound clarity, its warmth, or separability of stereo images of musical instruments. The basic and one of the most important signals measured in a room is its impulse response [Cox2017, Zolzer2011]. It is a signal measured in a given place of a room, being the response to

the excitation, an impulse approximating a Dirac delta. In practice, the excitation is often achieved by firing a starting pistol or by measurement with a signal with special autocorrelation properties such as the maximum length sequence (MLS) or a sine wave sweep [Guidorzi2015, Müller2008, Rothbucher2013]. The derivative characteristic of the room, which is calculated from the gathered impulse response, is the frequency response of the room. It is a measurement of linear subtractive or additive changes introduced to a given excitation signal by the room. It is usually calculated from an impulse response. But sometimes, it may be measured directly, for instance, with the use of white or pink noise and a related linear or octave-band filter bank. Impulse response may also be used for the calculation of other parameters. They are often used to assess the quality of the room acoustics in an objective manner. Examples of such parameters derived from the impulse response [Beranek1996, Everest2015] are provided in Tab. 2.1.

Tab. 2.1 Examples of parameters used for the description of the room acoustics that can be derived from the impulse response.

Parameter name	**Parameter definition**
Clarity	It is a measure allowing for assessment of the influence of early (useful) and late (useless) reflections on the intelligibility of speech and music signals. The metric is calculated with the following formula: $$C_T = 10\log\frac{\int_0^{T_i} h^2(t)dt}{\int_{T_i}^{\infty} h^2(t)dt},$$ Where T_i means integration time and is set to 50 ms to obtain clarity C(50) and 80 ms to obtain C(80), $h(t)$ denotes impulse response of the room.
Reverb time RT(x)	The RT(x) coefficient is a duration of time needed for an acoustic pressure level in a given room to decrease by x dB. Typical values of x are 20, 30, and 60 dB. The value of RT(60) is often approximated by measurement of RT(30) and calculation of RT(60) by multiplying the value of RT(30) by 2.
Early Decay Time (EDT)	Reverb time for the decay of reverb curve from 0 dB to -10 dB, therefore it formally can also be denoted as RT(10). It often can be employed as a good measure of speech clarity.

Another approach for validations of results is an assessment of the subjective opinion of experts, who assess the quality of the acoustics of a particular room. In such a case, it is also useful to define a set of parameters, making it easier to standardize answers from the experts taking part in a subjective listening test. Such parameters are subjective in their nature and have a descriptive definition. However, it should be noted that many literature sources correlate subjective evaluation with measurable parameters [Astolfi2008, Beranek1996, Carvalho1996, Fastl2007, Giménez2014, Zera1997], even though conclusions may be drawn as to the quality of room acoustics only after a thorough statistical analysis of responses given by the experts. Examples of parameters assessed in a subjective manner with the use of subjective tests are provided in Tab. 2.2 [Everest2015].

Tab. 2.2 Examples of parameters used for the subjective description of the room acoustics.

Parameter name	**Parameter definition**
Clarity	A measure of the ability to perceive details of an auditory stimulus and distinguish details such as musical notes or speech articulation.
a feeling of being enveloped by the sound	A measure of the subjective sensation of being immersed by the sound.
brightness	A subjective parameter correlated to the content of high acoustic wave frequencies.
warmth	A subjective parameter correlated to the content of low acoustic wave frequencies.
quality of spatial localization	A subjective measure of how easy it is to distinguish from which direction one perceives sounds in a given room. It may degrade if too many specular reflections are present.

As already mentioned, in the case of the subjective assessment of a room, it is necessary to take into account the fact that a very rigorous statistical analysis of the results of the experiment has to be taken to obtain reliable conclusions. Otherwise, a variance in responses, which may result from, i.e., individual differences in perception of acoustic sensations, may influence findings drawn from the collected answers. Another important factor is a choice of the right question asked to the participants of the experiment. It has to be formulated in such a way that will not influence the answers of participants taking part in the survey and should be connected to only one aspect of

the room acoustics, which is being assessed. The experiment also has to be designed so that listeners will not be forced to concentrate over a very long period. If this condition is not met, it may influence results obtained for parts of an experiment conducted near the end of the listening session because the participants will not focus on a task. It should be noted that subjective tests are standardized, so all the relevant assumptions should be considered to the minute when designing them [ITURBS1116, ITURBS1284, ITURBS1534, ITURBS2132].

2.1. Analysis methods used in the acoustic CAD and the acoustic optimization software

Computer modeling of the sound field finds its application in the design of rooms, their construction, or modernization. In designed models, it is possible to estimate the room acoustic properties with high accuracy when measurements are not possible or the room is still at the design stage. This significantly reduces the cost and time of construction and allows skipping the additional step of designing acoustic adaptation. At the same time, simulation of acoustic wave propagation can also be successfully employed for the estimation of properties of objects such as Schroeder diffusers.

The main goal of experiments shown in this dissertation is the use of acoustic simulation for automated optimization. Although it can be tempting to try to use an acoustic CAD for the evaluation of diffuser designs generated by optimization algorithms, there are some practical difficulties, which are the reason why it may not be the best way of evaluating Schroeder diffusers generated in such a way. Additionally, most CAD software does not provide an application programming interface (API) that would be suitable for fast testing of automatically generated measurement scenarios, which would be compatible with typical software solutions from the realm of machine learning such as TensorFlow and Keras libraries [Abadi2015, Chollet2015]. The aforementioned machine learning libraries are important components of authors software implementing the deep reinforcement-learning-based approach, which is described in this thesis.

On the one hand, the workflow which is typical for acoustic CAD is similar to the automated pipeline of operation needed in automated optimization tasks. Both processes can be divided into the stage of preparation, a geometry of the room and scattering obstacles, and then – a stage of simulation of acoustic wave propagation. The latter task also can be achieved in many ways in acoustic CADs. On the other hand, acoustic CADs are used mainly to simulate large spaces and rooms, which is not always the most desired scenario if automatic optimization of passive acoustic scattering devices such as Schroeder diffusers is considered. One of the main purposes of acoustic CADs is auralization, which is most frequently achieved through the means of geometrical methods [Everest2015]. This leads to the fact that specific methods are used to perform auralization of 3D modeled spaces. Namely – the geometrical room acoustic models (GRAMs) [Cox2017]. Examples of methods employed in GRAMs are as follows:

- A ray-tracing method, which is based upon an assumption that propagation

of acoustic waves can be approximated in a similar manner as in geometrical optics. Limitations of this method are also similar to ones associated with electromagnetic waves. Results obtained with this method are valid for high-frequency signals, which in the case of acoustic signals are sound waves with frequencies over 125 Hz. For signals fulfilling this condition, the ray-tracing method can be used to obtain impulse responses useful for auralization of designed spaces.

- A virtual source method, operating by calculating mirror images of primary sources, leads to creating the second, third, and higher-order sources. The signal obtained in the point for which the echogram is calculated as an effect of simultaneous propagation of an acoustic wave from both the real and virtual acoustic wave sources. As this method also assumes geometrical rules of acoustic wave propagation, it is also not suitable for the simulation of phenomena occurring for the lowest acoustic frequencies.

As seen from the above description, methods employed in many acoustic CADs are often based on the assumption that all of the acoustic wave propagation in modeled 3D spaces follows rules of geometric, ray-like propagation [Everest2015]. There are other types of acoustic wave propagation modeling that do not make this assumption – for instance, models based on solving the acoustic wave equation and methods based on statistical methods. To choose the correct modeling method, the sound field structure must be determined. Tab. 2.3 shows the dependence of the modeling method selection on the sound field structure.

Tab. 2.3 Dependence of modeling method selection on the sound field structure

Acoustic field structure	**Numerical modeling method**
Room with small dimensions in comparison to the wavelength	Wave-based method
Room with large dimensions in comparison to the wavelength The ordered structure of wave fronts	Geometric method
Room with large dimensions in comparison to the wavelength Unordered structure of wave fronts	Statistical method

Wave-based methods are based on solving the wave equation and are not limited by the assumption of geometric propagation of acoustic wave rays. Therefore such models are capable of simulation phenomena occurring in the immediate vicinity of a scattering object or propagation of acoustic waves in a situation of a temperature gradient, which is affecting, i.e. the density of the air and thus curving the propagation waves of acoustic waves, which is an example of the situation directly violating

assumptions made by geometrical models.

An alternative to both the wave-based methods and geometrical ones is the statistical method which assumes, that the sound field in the given space is a diffuser. Therefore a statistical equation can be employed to estimate parameters such as reverberation time. An example of such a formula is Sabine's equation for the calculation of the T_{60} parameter [Long2014].

$$\mathrm{RT}(60) = 0.161 \frac{\mathrm{V_{room}}}{\mathrm{A_t}}, \tag{2.1}$$

where

$\mathrm{V_{room}}$ denotes the volume of the room in meters,

and $\mathrm{A_t}$ is a total area of absorption in a room calculated according to the formula given below:

$$\mathrm{A_t} = \sum_{n=1}^{N} S_n \cdot \alpha_{room,n}, \tag{2.2}$$

where:

N is a number of boundaries (i.e. walls) in a room,

$\alpha_{room,n}$ is a mean attenuation coefficient characterizing *n*-th boundary in a room,

S_n is a surface area (in meters) of *n*-th boundary in a room.

The acoustic CAD software is a compelling and effective method for the simulation of room acoustics. Especially in situations that permit usage of geometrical and statistical methods, however, some models implemented in acoustic CADs also performed some kind of the wave-based method to improve the accuracy of predictions [Everest2015]. Often, they also allow the automatization of simulation. The main tasks of simulation programs include the detection of possible anomalies in strategic locations of the room, such as areas under balconies, alcoves, or corners of the rooms at the design stage of the room. However, for the purpose of simulations needed for automated optimization of acoustic diffusers presented in this thesis it still is more compelling to utilize a simulation library employing a method that is based upon a direct solving of an acoustic wave equation. This permits a more accurate simulation of phenomena associated with the lowest acoustic frequencies and physical phenomena taking place near objects interacting with incident acoustic waves. They are especially taking into the fact that if very precise outcomes are needed, then ray-based geometrical methods tend to require the application of additional correction factors, which improve the accuracy of their predictions [Bergman2018]. Such method choice also ensures that all such wave phenomena are taken into account for higher frequencies which is important if the space simulated is small.

An example of such simulation is a modeling of a measurement made in an

anechoic chamber, where the largest distance between objects is 2 m. Selected examples of such methods are:

- boundary element method (BEM),
- finite element method (FEM),
- finite-difference time-domain method (FDTD).

Algorithms such as BEM and FEM are capable of obtaining very accurate estimates of acoustic wave field distribution [Bergman2018]. Both of those methods are based on frequency-domain calculations, and due to this fact, their output does not allow to obtain impulse responses of a room in a direct way. An alternative to those two methods is the FDTD method. The use of a method based on acoustic wave equation solving also has other advantages. As FDTD is relatively easy-to-implement in terms of the length of code needed to obtain meaningful results, it easily integrates this simulation algorithm with the program performing optimization of an acoustic Schroeder diffuser. This, for instance, would allow the implementation of direct communication between simulation and design algorithms through the RAM memory of the computer. Also, it is desired to employ a method that directly permits the estimation of impulse responses in both room-related and anechoic conditions. This means that the best-suited methods are finite-element methods which tend to be slow. One variant of the finite element method, which is relatively simple to implement and can be sped up by using GPU-based computing, is an FDTD simulation, and this kind of simulation will be used for the optimization of acoustic diffuser designs in the dissertation [Cox2017, Webb2011].

3. SCHROEDER DIFFUSER – PROPERTIES, METHODS OF DESIGN, AND MEASUREMENT

This Section provides a theoretical background of acoustic diffuser properties, methods of design as well as principles of measuring such devices under anechoic conditions. This is crucial knowledge as it allows measurement-based verification of acoustic diffuser prototypes obtained by both the classical and optimization-based methods.

First, in Subsection 3.1, a definition of a Schroeder diffuser will be introduced together with a description of the basic methods for designing them with approaches based on QRD and PRD pseudo-random sequences. Next, methods for measurement-based verification of acoustic diffuser properties will be introduced in Subsection 3.2. An important measure will be introduced in the aforementioned subsection, namely the autocorrelation scattering coefficient. It is a measure that can be used both as a measure of prototype quality but can also be implemented as a parameter for optimization in a numerical simulation. This makes an autocorrelation scattering coefficient a well-suited candidate for a metric to be used for optimization in an anechoic simulation scenario. In Subsection 3.3, a method for designing acoustic diffusers with the use of genetic algorithms is proposed, and in Subsection 3.4, analogous methods employing deep reinforcement learning are presented.

3.1. Schroeder diffuser properties

Acoustic diffusion may be obtained by employing a number of operating principles – for instance - they can be passive devices relying on the phenomenon of reflection and diffraction (passive diffusers) [Arvidsson 2020], but also they can provide diffusion by using digital signal processing techniques and impedance modulation (active diffusers) [Avis2005, Collet2009, Cox2006, Lissek2013, Xiao2005]. The latter ones are also able to perform both as an active diffuser and an active absorber, thus becoming a hybrid device. This hybrid nature of active acoustic sound treatment devices may be a desirable feature if the room in which it is intended to be implemented is not large, and there are only a few places where an acoustic panel or a diffuser may be placed [Avis2004]. This advantage of active acoustic sound treatment devices makes them useful for complicated tasks such as active noise reduction and sound waves attenuation [Matten2017, Moreau2009, Rivet2017], or decreasing influence of unwanted acoustic modes in a room under the sound treatment process [Lissek2009, Karkar2014]. Unfortunately, active acoustic devices usually are expensive, complicated, and challenging to design. Passive acoustic devices can also have a very complex structure, especially if they employ metamaterials [Bongard2011, Schwan2017]. However, there are groups of passive acoustic treatment devices that, in some particular cases, may especially be easy to design and manufacture. An example of such a device is a Schroeder passive acoustic diffuser.

A Schroeder diffuser is an acoustic treatment device that can be defined by a matrix of integers greater than or equal to zero and a set of several scalar parameters with

real values. Such a system consists of a one-dimensional series or a two-dimensional well grid. In the first case, such a diffuser affects the sound field only in one plane. In the second - in two planes perpendicular to each other. The diffuser wells have different depths and are separated from each other by thin walls [Cox2017, Everest2015]. The proportions of the depth of the wells can therefore be presented either as a series of values if the diffuser interacts with the sound field only in one plane or a matrix of values if it interacts with it in two planes. The wells are a system of resonators that affect the wave front falling on the diffuser in such a way as to reduce its ability to interfere after reflection from such a diffuser. Examples of one-dimensional and two-dimensional geometry of the acoustic diffuser are shown in Fig. 3.1. The scalar parameters, which also belong to the description of the acoustic diffuser, include a factor for converting the ratio of the depth of the wells to their target depths and the physical dimensions of the wells, which at the same time define the dimensions of the entire acoustic diffuser.

The acoustic diffuser can influence the sound field due to the phenomenon of acoustic wave scattering, which can be used to design room acoustics so that they allow the existence of reflected waves, but at the same time, these reflections do not cause undesirable changes of an acoustic signal. An example of such changes may be, for example, the formation of standing waves and the resulting irregularity of the impulse response of the room. Acoustic diffusers in this context are employed for the controlled introduction of diffuse reflections of acoustic waves, which have a reduced ability to interfere, and thus to create standing waves. Sound scattering caused by placing acoustic diffusers in a given place also allows eliminating the phenomenon of so-called flutter echo, which is also an undesirable phenomenon, degrading the quality of listening in a given location [Everest2015].

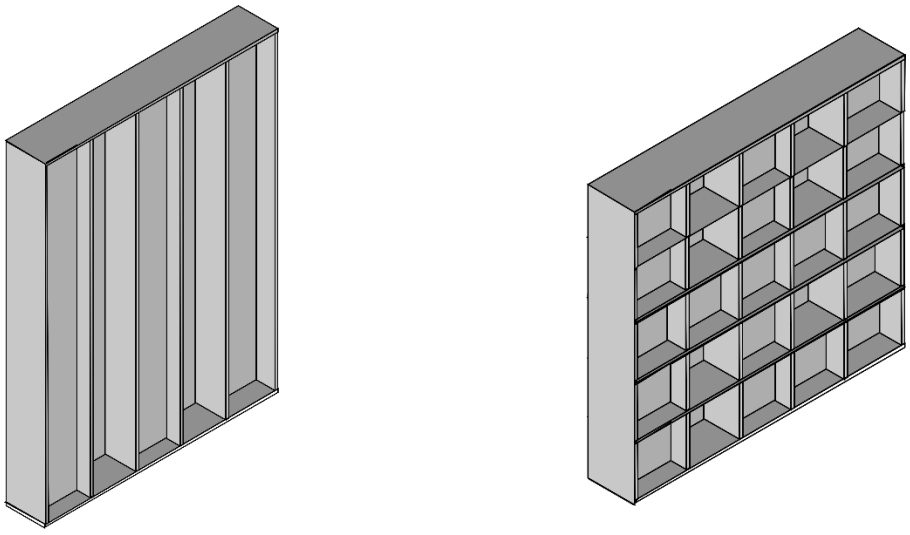


Fig. 3.1. Schroeder diffusers designed for one plane (left side) and two planes (right side).

The method of determining the geometry of Schroeder diffusers based on the proportions of the cavity depth and the external physical dimensions of such devices makes it possible to formulate design methods that are intuitive and allow for quick results in the form of diffuser designs that behave well in real application conditions.

These methods are based on the application of pseudo-random number generation techniques to create sequences of numbers determining the relative proportions of the diffuser cavity depths. This sequence of depth-defining numbers, together with the overall dimensions of the diffuser and the value used to convert pseudo-random numbers into real-world depths of the wells, define Schroeder diffuser geometry.

The properties of the scattering system vary depending on what type of pseudo-random sequence is used to design its geometry. Examples of frequently proposed sequences used for this purpose are MLS (maximum-length sequence), PRD (primitive root diffuser), and QRD (quadratic residue diffuser) sequences [Everest2015]. They come from the theory of generating pseudo-random numbers that are widely used in areas such as cryptography and telecommunications. The pseudo-random sequence is used to generate a random set of proportions for the depth of the diffuser wells. Thus, various diffuser design methods differ from each other, with a mathematical formula used to generate numbers defining the relative depths of the diffuser cavities. In diffuser geometry designed by the process based on QRD sequences, the ratio of the proportional depth of the *n*-th well marked by $W_{d,QRD}$ is given by the formula [Everest2015]:

$$W_{d,QRD}(n) = n^2 \bmod \xi, \tag{3.1}$$

where *n* is consecutive integers greater than or equal to zero, and ξ is the selected prime number.

For PRD diffusers, the ratio of the proportional depth of the *n*-th well is given by the formula:

$$W_{d,PRD}(n) = g^n \bmod \xi, \tag{3.2}$$

where *g* is the smallest primary root of the prime number ξ.

In the case of diffusers operating in two planes, it is assumed that *n* is the sum of the indexes for the row and column, i.e.:

$$n = n_x + n_y \tag{3.3}$$

Finally, the relative well depth factor is converted to the physical depth using the formula:

$$d_n = \frac{W_d(n)\lambda_0}{2\,\xi}, \tag{3.4}$$

where d_n is the physical depth of the diffuser well and λ_0 is the length of the acoustic wave corresponding to the design frequency f_0 of the diffuser. In addition, the acoustic diffuser is characterized by parameters determining the frequency range in which they

are active and effectively scatter acoustic waves. The upper band limit is given by the formula [Cox2017]

$$f_{ub} = \lambda_{\min}/2, \tag{3.5}$$

where w is the wavelength corresponding to the highest frequency band for which the diffuser is effective, and λ_{min} is the shortest wavelength for which no lateral propagation modes arise in the diffuser well, but propagation takes place according to the plane wave model. The second important parameter is the design frequency f_0 and the associated design wavelength λ_0, which are related to the acoustic diffuser. The acoustic diffuser is effective for integer multiples of the associated design frequency.

The type of pseudo-random number sequence used in addition to the algorithm for obtaining subsequent relative depths of individual wells also affects the properties of the diffusers themselves. The use of PRD sequences usually allows the reduction of energy reflected in the direction consistent with the law of reflection from a perfectly reflecting surface, provided the pattern of wells is repeated sufficiently many times [Cox2017]. However, it should be stressed that this situation occurs only for multiples of the design frequency and its integer multiples. If this condition is met, it is possible to reduce the reflection amplitude by $20\log_{10}(p)$.

It is important to note that Schroeder diffusers can be realized in two manners – with or without physical barriers (walls) between wells. Depiction of two such paradigms is provided in Figs. 3.2 and 3.3. Fig. 3.2 depicts a 1D case of such designs, and Fig. 3.3 depicts a 2D case.

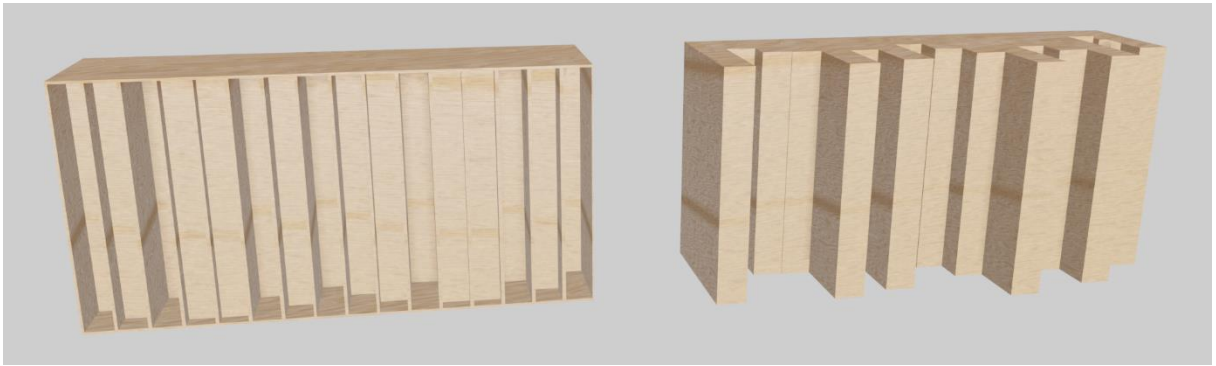


Fig. 3.2 Examples of 1D Schroeder diffusers designed with (left side) or without (right side) walls between the wells.

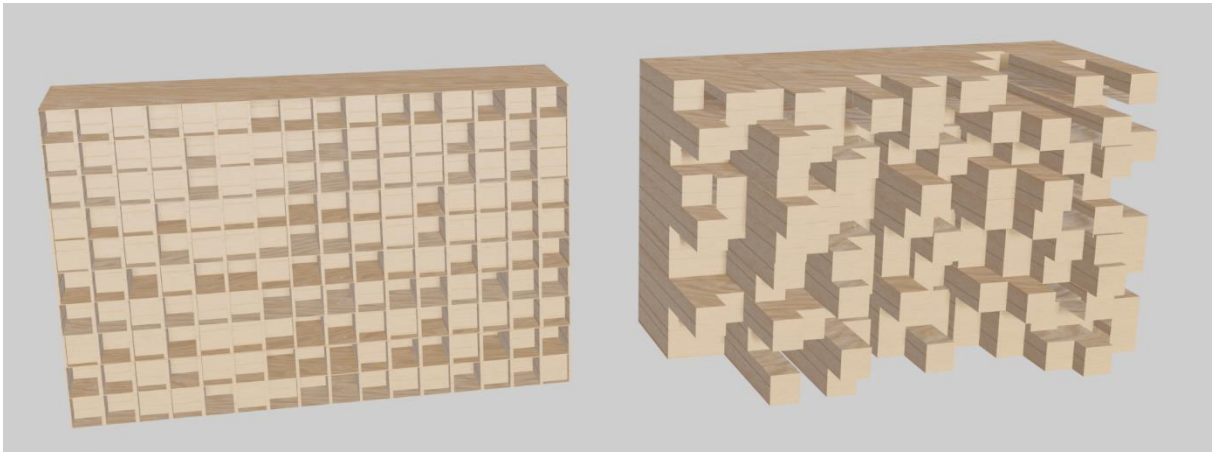


Fig. 3.3 Examples of 2D Schroeder diffusers designed with (left side) or without (right side) walls between the wells.

The best (in the sense of optimization) Schroeder diffusers generated by algorithms presented in this thesis are used to create prototypes, which then are tested in an anechoic chamber. Therefore, it was decided to design 2D diffusers without physical walls between wells (the right side of Fig. 3.3). This decision makes it easier to prepare prototypes for purposes of performing measurement in an anechoic chamber. Designs with walls were also tested, but only by means of computer simulation.

This decision makes it possible to prepare a formal description of a diffuser layout. The following parameters can fully define any 2D diffuser:

- a matrix of integer numbers denoting the height of each Schroeder diffuser well,
- physical dimensions of a single segment making a well (width, length, and height),
- maximum number of segments per well,
- dimensions of diffuser (in elements/wells, i.e. 10 elements by 10 elements),
- the thickness of the back part of the diffuser on which wells are positioned.

Visualization of such description method applied to an example 2D Schroeder borderless diffuser is shown in Fig. 3.4.

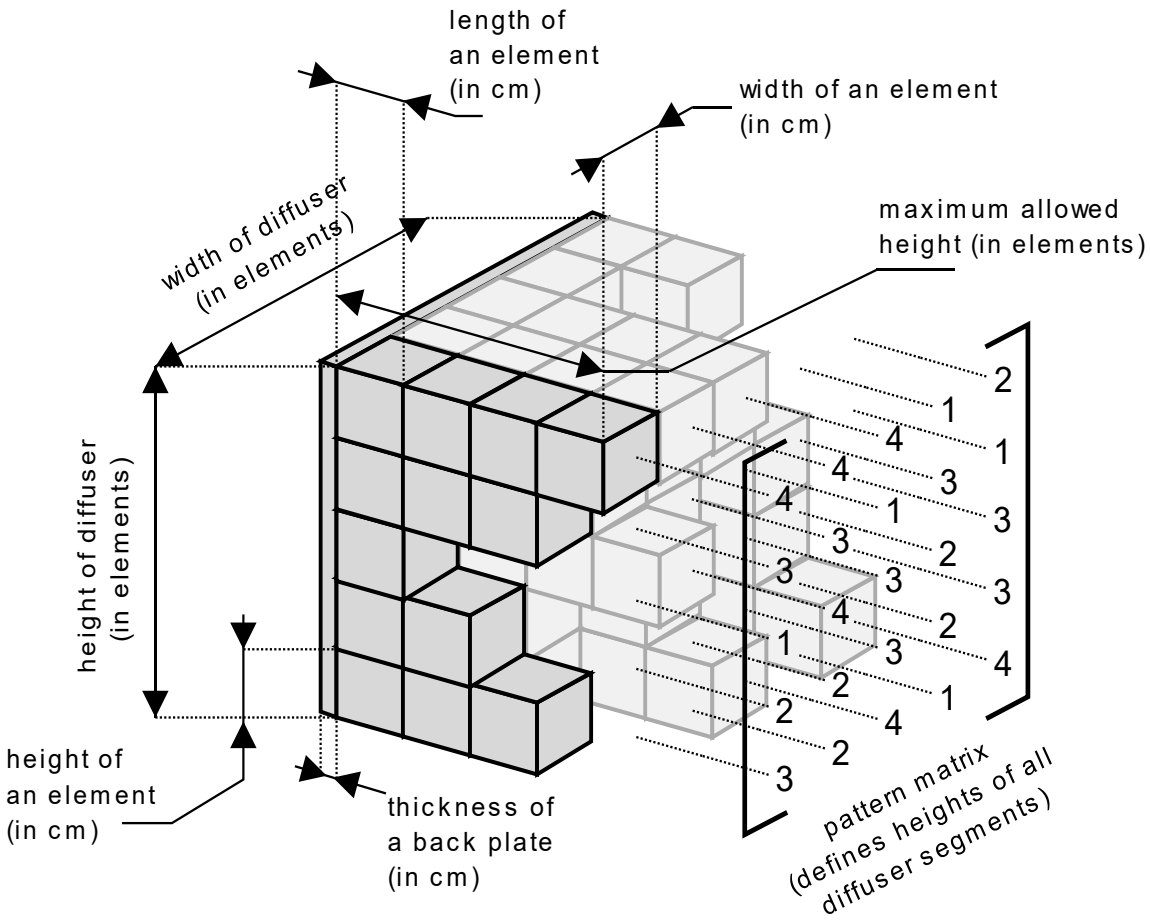


Fig. 3.4 Depiction of all dimensions and other data crucial for an unambiguous definition of a Schroeder-type diffuser simulated in experiments.

Such a description of the diffuser was employed in the course of all numerical and physical experiments. It can be applied to designs with borders and borderless ones. In the case of designs with walls, additional information about the thickness of walls has to be included in the description. The definition of diffuser structure presented in this Section is especially useful for algorithms optimizing diffuser designs, especially if the goal is the maximization of diffusion and scattering coefficients. Due to such choice of design quality metric effects of computational optimization of diffusers shape has a direct influence on matrices obtained from measuring the diffuser prototypes in an anechoic chamber. A detailed method for measuring the aforementioned parameters of acoustic diffusers is presented in the next subsection.

3.2. Methods for measurement of an acoustic field in the proximity of scattering devices

Apart from having parameters that are descriptive of the room acoustics itself, it is also necessary to have a set of parameters that can describe an acoustic diffuser performance in a standardized way. Such measures are calculated for the behavior of the diffuser in anechoic space. This can be calculated on the basis of an outcome of a measurement or a simulation that predicts the result of the measurement. Effectively, this allows for developing simulation models and measurement methods, which interact with each other and allow multiple levels of results correctness control. It is also possible to adopt an iterative approach to simulation and measurement-based testing of solutions developed in such a way – the final result is obtained by iterative stages of simulation-based design of the given acoustic device and the testing of the device by a measurement [Kurowski2016, Kurowski2017, Kurowski2018]. Visualization of such iterative design methodology is presented in Fig. 3.5.

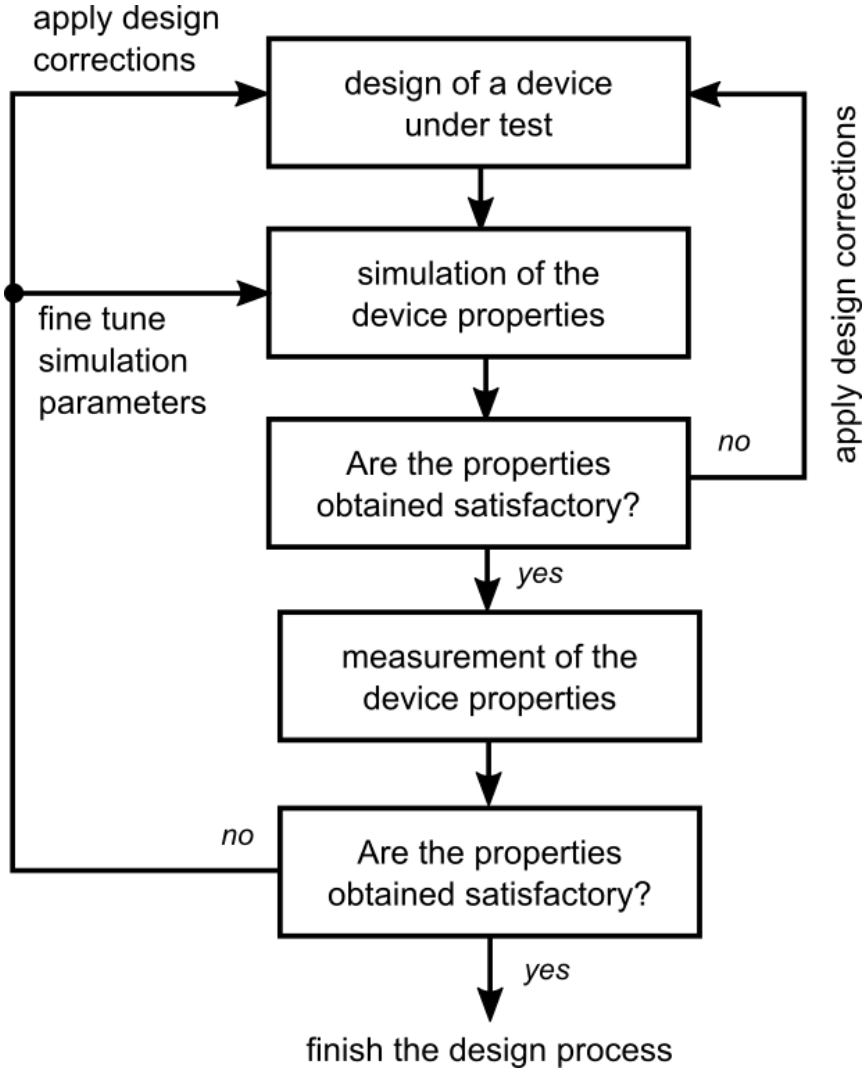


Fig. 3.5. The iterative design methodology employing both the repeated simulation and measurement to track the design goal in the process of preparation of a device. It can be applied to the design of acoustic diffusers, but is not limited to only such cases.

The aforementioned standard parameters, which are possible to both be measured (i.e., in an anechoic chamber) but also can be predicted by a computer simulation, are also useful because they may be employed as metrics optimized by a computer algorithm. Under anechoic conditions, it is possible to define the amount of energy reflected by the particular design in a specular manner. It is also feasible to determine to what extend the scattering pattern provided by the diffuser is uniform or calculate the acoustic pressure in front of the diffuser, which is the result of exciting the design with one single acoustic impulse. The type of metric imposes the way of measuring the sound field, especially the spatial structure of the measurement points. For some metrics such as uniformity of the acoustic pressure distribution, simple grid-like placement of points is more useful as it is desired to have a uniformly sampled map of the scalar value of pressure. In other cases, it is preferable to have a set of equidistant points placed on a semicircle, which makes it easier to calculate metrics related to polar characteristics of the measured diffuser geometry [Cox2017]. An example of such a semicircular pattern is presented in Fig. 3.6.

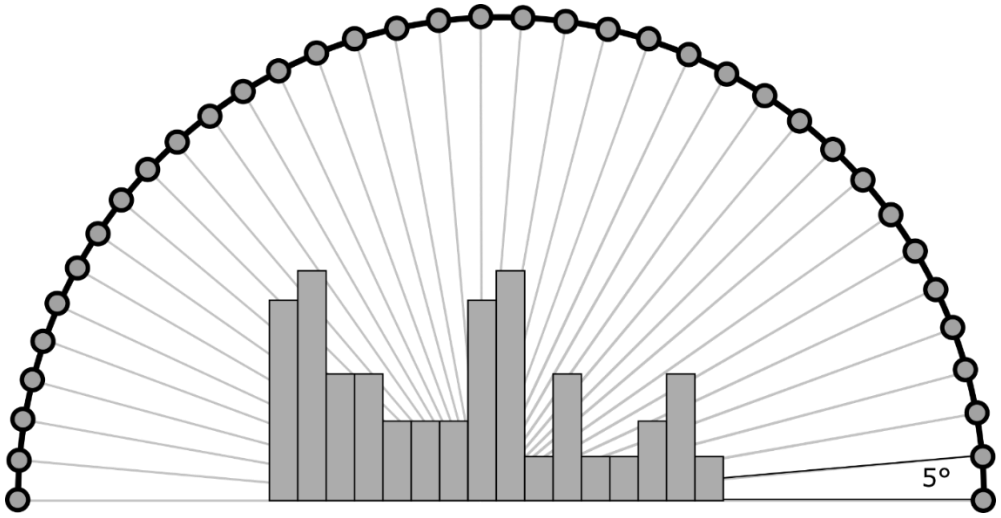


Fig. 3.6. A common pattern of 37 measurement points used for the evaluation of the scattering and reflection coefficient of a given acoustic diffuser device. The resolution of such measurement is equal to 5 degrees.

A typical resolution of such measurement is 5 degrees, and thus in practice, it is necessary to perform 37 measurements to obtain all required information on the diffuser design and calculate derivative coefficients related to the device measured. Data obtained from measurement are impulse responses captured in points positioned in a manner similar to one presented in Fig. 3.6. This allows the calculation of crucial parameters describing the behavior of particular diffuser geometry in the acoustic field.

There are two important measures that are used to describe the properties of acoustic diffusers – diffusion and scattering coefficients. Often, estimates of those parameters derived from the polar response are calculated, namely, the correlation diffusion coefficient and correlation scattering coefficient. They are a practical alternative for the canonical version of parameters as they can be obtained from a polar response, which can be measured only once.

The correlation diffusion coefficient d_{ψ} is calculated by the use of the following formula [Cox2017]:

$$d_{\psi} = \frac{\left(\sum_{n=1}^{N} 10^{L_n/10}\right)^2 - \sum_{n=1}^{N}\left(10^{L_n/10}\right)^2}{(N-1)\sum_{n=1}^{N}(10^{L_n/10})^2}, \tag{3.6}$$

where:

L_n is a sound pressure level in an *n*-th measurement point,
N is the number of measurement points,
ψ is the angle of the excitation acoustic wave incidence.

An important assumption is that for this formula to be valid, measurement points must be placed on the semicircle. The diffusion coefficient is a measure of the uniformity of the reflected sound. This coefficient is usually used as an evaluation measure in acoustical diffuser design and for comparison of the performance of various diffuser geometries. Unfortunately, even a flat surface has some diffusive properties according to the given formula. To reduce this effect, often, a reference measurement of a flat reflective plate is taken in addition to the measurement of the diffuser under test. Next, a normalized diffusion coefficient is obtained with the following formula:

$$d_{\psi,n} = \frac{d_{\psi} - d_{\psi,r}}{1 - d_{\psi,r}}, \tag{3.7}$$

where:

d_{ψ} is the correlation diffusion coefficient of the diffuser in question,

$d_{\psi,r}$ is the correlation diffusion coefficient of a flat reflective plane.

Another important parameter derived from such a set of points is a correlation scattering coefficient. It is calculated with the following formula:

$$\delta_c = 1 - \frac{\left|\sum_{i=1}^{n} p_1(\theta_i) \cdot p_0^*(\theta_i)\right|^2}{\sum_{i=1}^{n} |p_1(\theta_i)|^2 \sum_{i=1}^{n} |p_0(\theta_i)|^2}, \tag{3.8}$$

where:

p_1 denotes the pressure scattered from the test surface,

p_0 is the pressure scattered from the flat reference surface,

θ_i indicates the receiver angle of the *i*-th measurement position,

$*$ is a complex conjugate operation,

The scattering coefficient is a measure of the portion of energy that is reflected in a non-specular way from a given surface compared to the whole energy of the acoustic

wave reflected from a surface. It is usually employed for geometrical room simulations.

The use of polar response for the calculation of parameters of acoustic diffuser designs makes it possible to optimize the process of measurement. For instance, there is a possibility of automating the measurement with the use of a Cartesian robot. The anechoic robot is an example of an automatized mechanical framework capable of fast measurement of an acoustic field in a given area. Measured value may vary because an arm of such a robot can handle various sensors [Szczodrak2016, Kotus2015]. For instance, a sensor may be a measurement-grade microphone that captures only an acoustic pressure-related signal. Still, it also can be an acoustic intensity probe, which measures not only scalar characteristics of an acoustic field but also an acoustic velocity vector, which leads to the measurement of acoustic energy flow in a given measurement point [Fahy1995, Weyna2001]. The basic idea of measurement is similar – the probe is positioned in discrete points in the space measured.

The process of such automated measurement is depicted in Figs. 3.7 and 3.8. It may happen that the measurement is taken while the probe is moving. However, such an approach causes a number of additional difficulties. Firstly, a noise of stepper motors or other equipment necessary to move the arm of the robot may be captured by the probe. Another problem is the resolution of the measurement. Also, the influence of vibrations of the robot structure may be eliminated by delaying the measurement until the whole structure of the robot stabilizes.

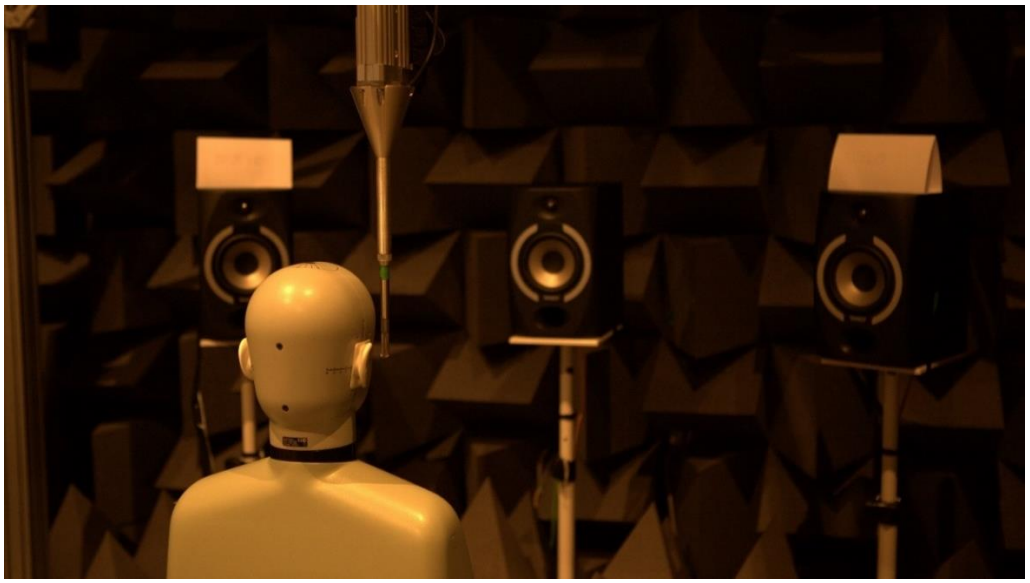


Fig. 3.7. A p-u sound probe during a measurement process. The device is used to measure sound field distribution in the proximity of the head and torso simulator. Positioning the probe for each measurement point is done by a Cartesian robot controlled by a PC computer.

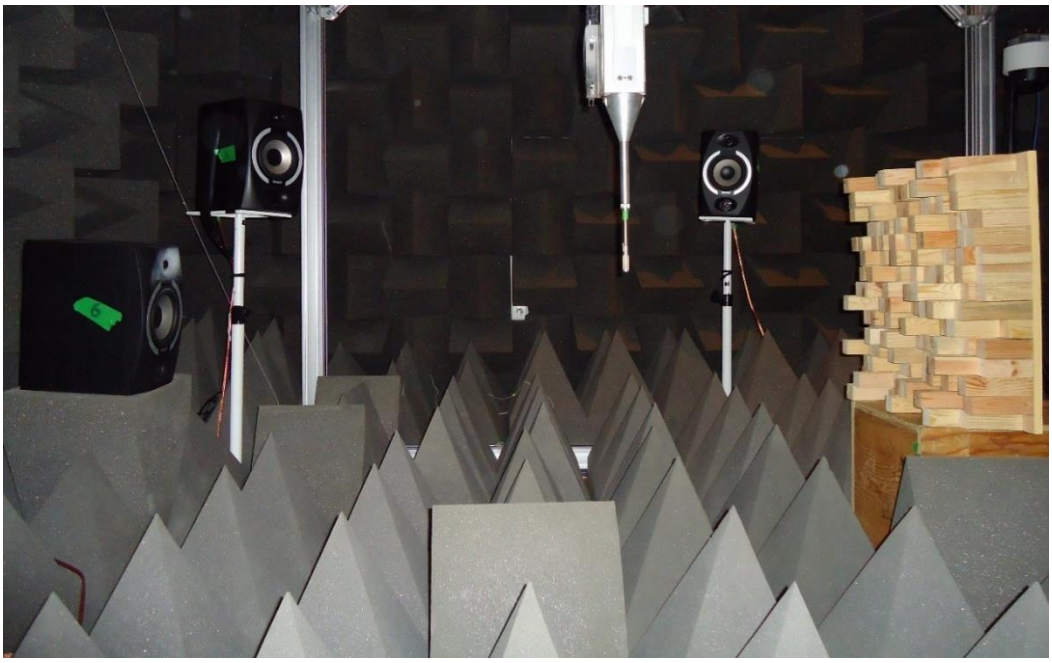


Fig. 3.8. A p-u sound probe during a measurement process of an acoustic diffuser prototype. For calculation of scattering and reflection coefficients, it is often sufficient to measure the level of acoustic pressure for points placed on the semi-circle surrounding the diffuser.

Moving probe captures the averaged data from a distance traveled by the probe. Therefore, it is necessary to perform accurate measurements during a single movement of the probe to ensure that a satisfactory resolution of the final measurement is obtained. On the other hand, this method may be faster than based on performing the measurement in a set of discrete points.

3.3. Design of acoustic diffusers with the use of the evolutionary algorithm

In addition to design methods based on pseudo-random sequences, there are also more advanced methods for designing acoustic diffusers. For some problems, an optimization process may be the most viable way of obtaining satisfactory results in fields of knowledge such as acoustics and fluid mechanics. An example of such a task may be a design of an acoustic horn with specific properties [Schmidt2016], a structure of rotating turbines [Nguyen2017], or even a biomedical problem of designing an aortic valve [Spuhler2018]. Most of those approaches employ some form of computer simulation; many of them are based upon the finite element method, which is used to calculate the optimized solution quality metric. This approach may also be applied to the problem of automatic designing an acoustic diffuser.

The metric is only one part of the optimization-based approach to automated acoustic diffuser design. The second element is an optimization algorithm. An example of a popular optimization algorithm employed for automated design tasks may be a genetic algorithm [Thede2004, Patraquim2017]. When designing acoustic diffusers

using a genetic algorithm, it is necessary to define the so-called genome. For diffusers, it is a matrix containing proportional cavity depth coefficients and a range of values that can be a number representing the acceptable cavity depth coefficient. First, a population of N_i random individuals consistent with the given genome is generated. Individuals may be subjected to crossing-over and mutation operations.

In the algorithm examined in the experiment conducted in the latter parts of the thesis, crossing-over involves randomly swapping columns or rows of two diffuser designs. Whenever the swap involves columns or rows is random, the probability of choice is 0.5. Then the algorithm iterates through rows or columns and converts them between projects with a probability of 0.52. The mutation operation also involves iterating the algorithm randomly through design rows or columns and changing the cavity depth coefficient value by a random, non-zero integer from -2 to 2 with the exclusion of zero. The value of N_i used in the successive parts of this work is assumed to be equal to 30.

This action causes half of the cavities after mutation is changed by one or two quantized differences in height. For high *N* values, this allows for the smooth evolution of designs, which is mitigated by the fact that the probability of mutation is high. A schematic representation of the operations for obtaining new diffuser designs in the initial diffuser population is shown in Fig. 3.9.

The genetic algorithm is an iterative process. Within each iteration, a so-called generation of solutions is then tested. The result of the test is the value of the fitness function, which is a parameter optimized by the genetic algorithm. It is assumed that the value of this parameter is maximized; however, in the case of the minimization task, it is enough to multiply the given fitness function by -1, which causes that the task of minimizing the value of the function will change into the task of maximizing its value. The exact definition of the fitness function depends on the definition of the problem to be solved by the genetic algorithm.

After each iteration, new diffuser designs are evaluated by calculating a fitness function that determines how much they improve the acoustics of the room in which they are applied. Schematically, the algorithm used in this Ph.D. thesis work is presented in Fig. 3.10.

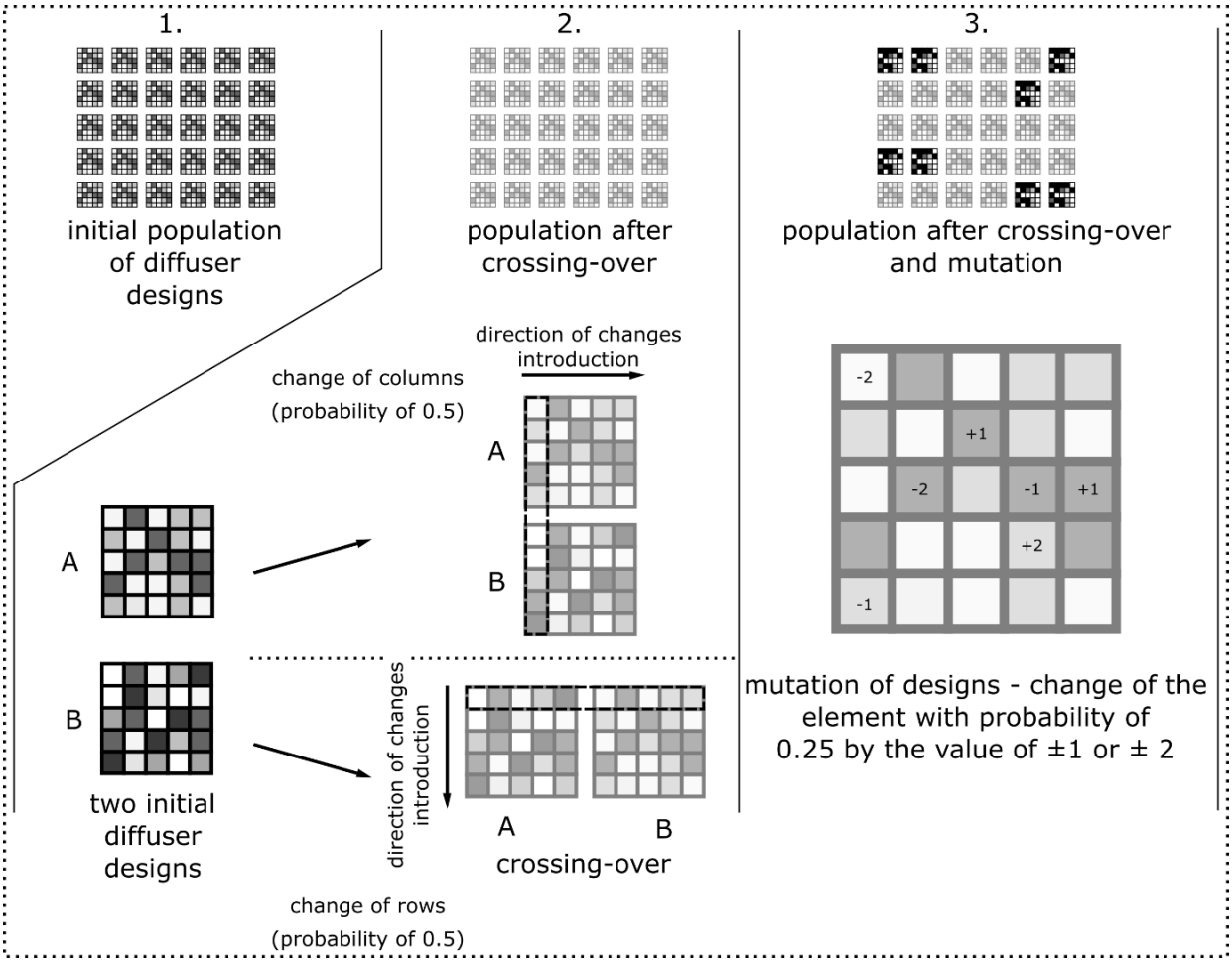


Fig. 3.9. Diagram illustrating the process of crossing-over and mutation used in the genetic algorithm. In the crossing-over operation, randomly exchanged columns or rows of diffuser designs. In the mutation process, the height of the elements is changed by ±1 or ± 2.

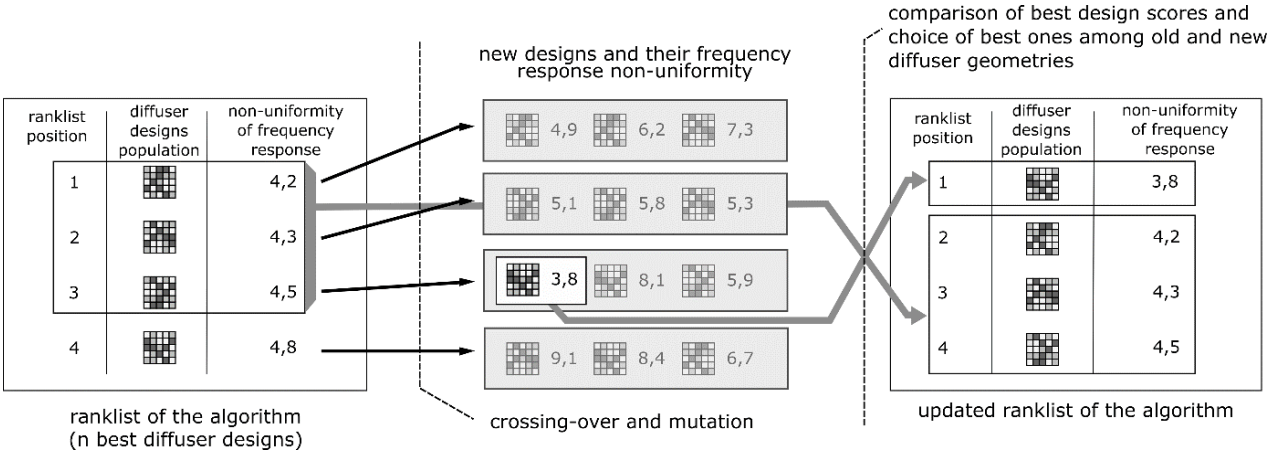


Fig. 3.10. Diagram illustrating the process of selecting the best-fit diffuser designs after mutation and crossing-over.

In the case of acoustic diffusers, the task may relate to maximizing the diffusion coefficient, diffusion coefficient, or minimizing the standard deviation of the frequency

response at a given point of the tested room. After selecting new diffuser designs according to the mechanism in Fig. 3.10, the list of best diffusers is updated based on the final ranking list, and the process of crossing-over, mutation, and calculation of the ranking is performed again. Interruption of this process can occur either automatically, e.g., after reaching the desired value of the optimized coefficient characterizing the diffuser design (e.g., uneven frequency characteristics), or, for example, after a predetermined number of algorithm epochs was carried out.

3.4. Design of acoustic diffusers with the use of reinforcement learning algorithm

Many real-world problems are difficult or even impossible to solve using simple algorithms that may be performed by a computer. Often in such cases, we have to employ heuristic methods to obtain an approximate solution to our problems. One of the recent and very promising fields of knowledge that delivers tools for such heuristic dealing with demanding, real-world problems is machine learning and deep learning. They are successfully addressing such problems as automatic mixing and automatic generation of musical pieces [Jaques2016, Martinez2017], audiovisual emotion recognition [Kim2013], speech recognition and voice conversion [Desai2009, Hinton2012, Mobin2016], sound synthesis [Donahue2018, Oord2016], or data mining tasks such as performing feature engineering to design best feature vectors for a given classification task [Le2012].

Deep learning is currently mostly associated with techniques employing neural networks. However, some methods which can also arguably be classified as connected to deep learning utilize other algorithms such as deep Boltzmann machines [Salakhutdinov2009]. Unfortunately, in the case of a majority of algorithms used in deep learning, especially ones employing neural networks, there is difficulty in analyzing how they exactly operate and how they exactly carry out the reasoning process and come to their decisions. This leads to problems in analyzing their behavior, difficulties in, i.e., formulating theoretical requirements for the structure of neural networks or other, more practical and implementation-related problems. These may refer to, i.e., the possibility of tampering with their performance using specially manipulated input data (so-called adversary attacks) [Kereliuk2015].

Reinforcement learning algorithms are a group of machine learning methods for tackling problems involving interaction with the environment and managing the process of getting knowledge from such an interaction. A reinforcement learning algorithm often takes the form of a so-called agent that interacts with a given, often unknown environment by taking actions. The number of actions possible for being taken can be limited but can also be infinite. Sometimes the action space can even be continuous. The outcomes of actions undertaken by the agent are evaluated, and feedback information about the quality of such action is returned to the agent in the form of a so-called reward. A simple example of the implementation of such a process is an autonomous agent in the form of a robot that learns to navigate a maze, but there are also examples of more complex implementations. For instance, the approach based

on reinforcement learning may be employed for an unsupervised design of neural network structures [Baker2016] or playing computer games [Choi2017, Kempka2016, Mnih2015, Usunier2016].

A reward is the main signal used for training the algorithm to choose the best actions. It is often a numeric value and is maximized by the agent in the process of learning and interacting with the environment. The definition of such a reinforcement learning process is depicted in Fig. 3.11 [Sutton1998].

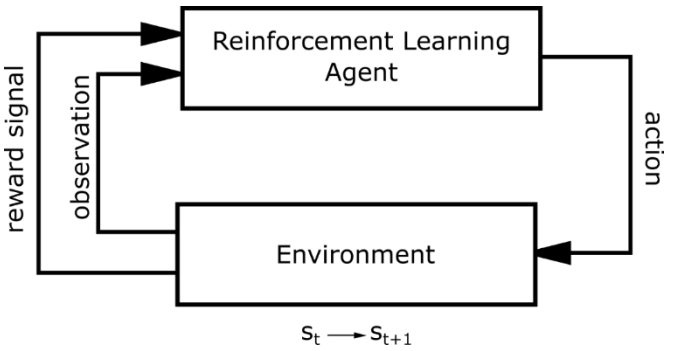


Fig. 3.11. A definition of reinforcement learning problem – the agent interacts with the environment by taking actions. In return, it obtains feedback in the form of the reward signal and observation about the next state after the action.

In reinforcement learning, a common problem is taking into account the value of future states, which are not certain to happen but may yield a future reward. A common practice is introducing a so-called discount factor to estimate future rewards that may be obtained after taking a certain action. The expected reward in such a case is equal to:

$$E(r) = r_t + \sum_{i=1}^{N} \gamma^i \cdot r_{t+1} \tag{3.9}$$

where:

$E(r)$ is respected reward to be obtained by the agent,

r_t is a reward achieved by the agent in the current step of the interaction with the environment

γ is a discount factor, $\gamma \in \langle 0,1 \rangle$,

r_{t+1} is a reward obtained by the agent in future steps of interaction.

Discount factor allows modifying the behavior of the agent with respect to valuing future rewards. For high values of γ the algorithm values future rewards. For low values of γ, the algorithm values the instant rewards and is less influenced by rewards possible to be obtained in the future. The discount factor is one of the important hyperparameters of reinforcement learning algorithms.

Another hyperparameter of reinforcement learning algorithms, especially ones based on a deterministic way of determining future actions, is the concept of ϵ-greedy policies. In the case of value-based reinforcement learning, the action is chosen by finding an action for which a value of Q is maximum, and thus the expected reward is maximal. The ϵ-greedy policy introduces a chance of taking random action instead of one being an inference with the values of Q functions [Buduma2017]. Thus it enhances the ability of the algorithm to explore the state space instead of exploiting known trajectories. The ϵ parameter determines the probability of taking a random choice of an action. Exploration versus exploitation trade-off is one of the common problems found in the design of reinforcement learning algorithms. There is also another method of introducing exploration to reinforcement algorithms, which is the use of probability distribution generation to define a distribution of actions undertaken by the algorithm and then use of this distribution to determine the action randomly. This kind of solution guarantees a chance of making less probable decisions, which introduces additional trajectories to be explored by the algorithm.

The decision process in reinforcement learning is assumed to have the Markov property. Thus, the state and the value of the reward signal are affected only by the current action, and past actions do not affect outcomes gathered from the environment. Interaction between the agent and the environment has a particular structure. Every single interaction consisting of performing a single action is called a step. A sequence of steps is called a trajectory. The environment may have states which are not possible to be left after entering them. Such states are called terminal states and are often ending states of trajectories taken by the reinforcement learning agent. Each sequence of states ending with the terminal state is often called episodes. The distinction between states and episodes is important because some reinforcement training algorithms introduce changes to the policy every step. Some other of them present them already at the end of the episode. This fact may affect the features of the algorithm, such as the speed of learning. The reinforcement learning agent may be designed in many ways. In the simplest case, it can consist of a matrix of values indicating which action should be taken in the case of a given observation retrieved from the environment. A more and more popular approach to this problem is the use of artificial neural networks, especially deep neural networks, as parts of a reinforcement learning agent. They can both be used for a direct choice of action to be undertaken by the agent or to estimate the future reward expected after taking a particular action. Each of them is the element defining a so-called policy, which is often denoted as $\pi(s, a)$, where s represents a state in which currently is an agent and a denotes action. The goal of training the reinforcement learning agent is to find such a policy that maximizes the expected reward obtained by the agent.

One of the means of obtaining an agent capable of interacting with complex environments is the use of a Q network. This is an example of the so-called value reinforcement learning algorithm. The Q network, or more often, a deep Q network, estimates a quality metric for each action possible to be taken by the algorithm in each step of interaction with the environment. The Q metric denotes a so-called quality of

the action. It is equal to the expected reward obtained by choosing the given action and further following the given agent policy. In real-world scenarios, the Q function is approximated and has to be learned through interaction with the environment. If there are many possible actions, and the state description is complex, the function estimation Q values can be very complicated. Hence, deep neural networks are a common choice as predictors of Q values. A general structure of such a neural network is depicted in Fig 3.12.

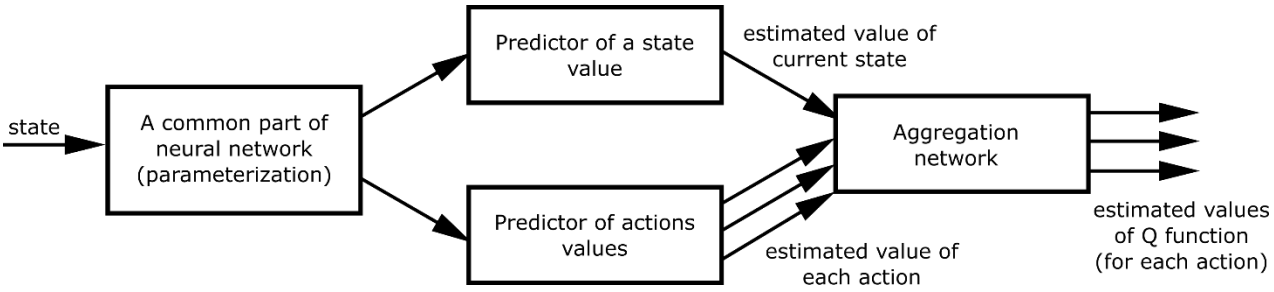


Fig 3.12. The overall architecture of dueling deep Q network.

In practice, the agent often contains two copies of the DDQN network. The first copy is used for the calculation of actions taken in the current state of the environment in which is the agent. A gradient descent algorithm optimizes weights defining this network after every step of interaction with the environment. The second one is identical to the first one and is used for the calculation of Q estimates for a future state after performing the action suggested by the first network. This is necessary to calculate the error for the first network. Weights of target networks are updated after a chosen number of updates are performed on the first neural network. This introduces additional numerical stability as estimates of future Q do not change rapidly due to the constant optimization process, which is applied to the first network. A target value of Q in the case of DDQN neural network has the following form [François-Lavet2018]:

$$Y_k^{DDQN} = r + \gamma Q\left(s', \underset{a\in}{argmax}\, Q(s', a, w_k); w_k^-\right), \tag{3.10}$$

where:

Y_k^{DDQN} is the value of Q function for the *k*-th action undertaken by the algorithm,

a represents actions from the action space A,

w_k is a set of weights defining a neural network used to calculate actions of the agent

w is a set of weights defining a neural network used for the evaluation of the best Q value of the future state.

The first network is trained by pairs of the environment state and corresponding values of Y_k^{DDQN}. The network has one additional feature which allows it a more precise estimation of values of the Q function. The architecture depicted in Fig 3.12 is a so-called deep dueling Q network. Such a type of Q network performs estimation of state

value and action value separately. The action value is also often called an advantage. The Q value is a sum of state value and advantage value. This operation is performed by the aggregation network. Also, the state value predictor and action advantage predictor are often fed parameters from the common part of the neural network used to extract parameters from the state description provided to the deep Q network. One of the advantages of the DDQN algorithm is the fact that changes to the neural network are applied after each step. Therefore they often converge to the desired solution faster than other algorithms such as the policy-gradient method.

One of the prominent problems, when constant on-line interaction with the environment is considered is the fact that consecutive decisions and their consequences may be correlated. This is especially visible in the case of agents utilizing screen images as input states. An example of such a situation is a neural network playing the Atari computer game. In such a case, consecutive frames of the video signal captured and fed on the input of the network are correlated. This can have a detrimental effect on the quality of the neural network training process and may even lead to a lack of convergence of the training process. A better scenario would be if a set of experiences from interaction with the environment could be gathered and then shuffled as in a standard supervised learning approach. This can be achieved by introducing a so-called replay memory [Foerster2017]. It is a structure that contains descriptions of all transitions between environment states together with information about reward obtained and action chosen. When a DDQN network is trained, a replay memory is sampled in a random manner, and a batch of training examples is created from the selected examples of transitions. Such an algorithm prevents the occurrence of strong correlation, which would be present if consecutive actions were taken directly from the memory of past actions as states and actions placed close in the time axis tend to be strongly correlated.

Another approach to reinforcement learning is a deep policy-gradient (DPG) algorithm, which also employs a deep neural network as a part of the agent. In this case, the output of the network is a probability distribution of actions to be taken by the agent. An example of such a neural network architecture is presented in Fig. 3.13.

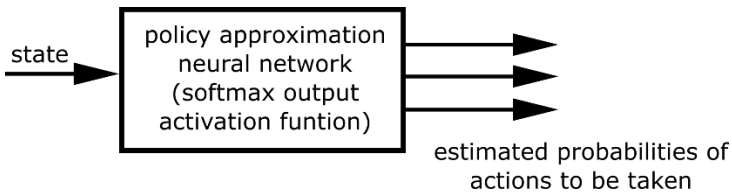


Fig. 3.13 A general architecture of a neural network used for a policy-gradient reinforcement learning algorithm.

In the case of a policy-gradient neural network, the network itself approximates learned policy. The policy-gradient ascent algorithm, which objective is to maximize the reward gained by the policy gradient algorithm, is employed in such a case. Weights of the network (denoted by w) are updated by a gradient ascent algorithm. The gradient

is calculated with the use of the following formula:

$$\nabla_w \pi_w(s,a) = \pi_w(s,a) \nabla_w log\big(\pi_2(s,a)\big), \tag{3.11}$$

This can be implemented with standard libraries allowing the implementation of gradient-descent training libraries. A special loss function, taking into account Eq. 3.11, can be constructed, thus allowing the implementation of a policy-gradient algorithm. A drawback of the policy gradient algorithm is the fact, that update for policy neural network may only be applied after a full episode of interactions with environments. Therefore they often converge with lower speed. On the other hand, policy-gradient methods make it easier to design solutions that rely on a probability distribution of actions to be taken and thus do not require explicit complication such as ϵ-greedy strategies to encourage off-policy exploration of the environment. They also consist of less complex neural networks and do not require additional data structures such as replay memory, which further simplifies their practical implementation.

Both DDQN and DPG algorithms may be applied to the problem of acoustic diffuser design. Also, there are techniques that make use of the hybrid approach based on a mixture of DDQN and DPG approaches. Such a family of methods is called an actor-critic approach [Grondman2012]. In each case, the environment is a numerical simulation of the acoustic diffuser behavior. This can be achieved by using, for instance, a finite-difference algorithm (FDTD). The reward is a change of optimized metric – positive if the change of the metric value is in the desired direction and negative in another case. Actions are used to introduce changes to the design of the acoustic diffuser, which is reevaluated after each change of its design. In such a setting, the machine learning algorithm learns to optimally choose the sequence of actions to optimize the value of the chosen property of an acoustic diffuser.

All measurement and design methods presented in this chapter are especially used if they are coupled with an optimization algorithm. Thanks to such a design, it is possible to create an algorithm, which proposes a set of designs, which scattering properties are evaluated by computer simulations. Next, improvements are made to the best design, and the whole process is repeated until the results are satisfactory. This approach requires the use of simulation, which is capable of predicting input data for Eq. 3.6, namely – the impulse responses. In a measurement setup, they are obtained with the use of a microphone or an acoustic probe. In the realm of the simulation, equivalents of those impulse responses have to be obtained from a computer simulation. Techniques that can be used for such simulations are shown in the next Section.

4. SIMULATION OF ACOUSTIC WAVE PROPAGATION

The purpose of this chapter is to present methods used for acoustic simulations, which are employed in further Sections associated with the automatic, optimization-based design of acoustic diffusers. The simulation is an element of an optimization algorithm responsible for generating feedback for optimization algorithms (such as genetic algorithms or deep policy-gradient networks). This feedback is utilized to select best-fitted designs (in the case of genetic algorithms) or generate a reward signal used by a gradient optimizer in the process of neural network training (in the case of deep policy-gradient network or deep Q network). Methods described in this chapter enable simulation of both enclosed spaces and anechoic conditions. The latter is achieved by the use of Berenger's boundary condition. For the purpose of this thesis, a special version of discretized acoustic wave equation was derived on the basis of the work of Webb and Bilbao [Webb2011]. They proposed a form of the discretized acoustic equation, which is easy to implement on GPU. The modified equation is also easy to be implemented on GPU, but at the same time allows for the simulation of lossy propagation medium and thus, at the same time, allows using Berenger's perfectly matched layers to simulate anechoic conditions. Finally, a method for calculating necessary wave parameters of propagation medium such as acoustic wave propagation speed and the acoustic impedance of propagation method is provided. The input values to the proposed method are atmospheric pressure, temperature, and relative humidity.

In this Section, first, the FDTD method is shown, employed to simulate wave propagation in a given geometry of a diffuser designed. Then, a perfectly matched layer equation is derived to achieve anechoic propagation in a numerical simulation is applied. Following this, a simulation of a mock-up room and methodology for assessing the quality of diffuser designs is presented.

4.1. Simulation of acoustic wave propagation by the FDTD method

In the process of selecting the diffuser geometry, each of the proposed designs must be assessed in terms of its impact on the acoustics of the simulated room. There are many ways to simulate the propagation of acoustic waves that can be used for this purpose. The main problem with the simulation of wave phenomena related to acoustic wave propagation is obtaining a wave equation solution, which is a differential equation. The popular choice to evaluate the shape of an acoustic field in given conditions is obtaining the approximate solution to the given equation, for instance, employing a finite difference approach [Farlow1993, Hamming1986]. However, this approach can be further divided into a few subcategories. In acoustics we can find for instance boundary element method [Brick2008, Citarella2011], finite element method [Logg2010, Phunpeng2015], or the finite-difference time-difference (FDTD) method [Kirkup1998]. Finite-difference methods are applicable to a wide range of problems associated with wave propagation. Therefore it should be noted that they can also be commonly found to be used for simulation of phenomena associated with the

propagation of electromagnetic waves [Chaber2017, Niedermayer2015]. It is worth remembering that in some cases, it may be viable to employ other simulation methods such as acoustic transfer vectors [Gerard2012, Tournour2000] or acoustic ray-tracing [Pompei2009].

Python is one of the most popular programming languages that is used for numerical calculations. There are examples of calculations employing finite-difference methods [Brennan2013, Langtangen2016, Logg2010, Phunpeng2015]. Libraries for the C++ programming language such as BEM++ are also available [Śmigaj2015]. However, the main advantage of the algorithm implementation in Python is the simplicity of using both the finite-difference numerical libraries and other already implemented algorithms such as ones found in deep learning libraries available for the Python programming language. Another advantage of using the Python programming language is an easy interface for implementing acoustic simulation software employing many-core computation architecture using the GPU acceleration, which significantly affects the performance of finite-difference algorithms [Markall2012].

From the point of view of simulating acoustic diffuser behavior, one of the most beneficial simulation methods is the FDTD technique. Its advantage lies in the simplicity of implementation and ease of acceleration of calculations using graphics cards. This is possible because the numerical simulation using the FDTD method is based on calculations performed independently for each point of the computational domain divided into nodes. Thus, it is possible to accelerate the calculations in the graphics card computing unit by parallelizing them for each computing node. These features make the FDTD method popular not only in the case of simulation of acoustic wave propagation but also in the simulation of phenomena associated with electromagnetic waves [Inan2011].

The equation used in a computer simulation is derived directly from the acoustic wave equation, which has the form [Kim2010, Kutruff2009]:

$$\frac{\partial^2 p}{\partial t^2} = c^2 \nabla p, \tag{4.1}$$

where p is the sound pressure function over time and c is the velocity of the sound wave propagation in the air. In the case of simulations in three-dimensional space, it is possible to present the value of ∇p as the sum of three partial derivatives of the pressure signal after successive spatial components:

$$\nabla p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2}, \tag{4.2}$$

Then, to derive the differential equation that is the basis for implementing the algorithm simulating the propagation of acoustic waves in the spaces studied, it is necessary to approximate partial derivatives over time and after spatial components

using finite differences. The type of substitution carried out to fulfill this requirement plays a crucial role in terms of the stability of the FDTD method. In equations used in experiments and presented in this thesis, a leapfrog method is employed, as this calculation method is unconditionally stable [Bergman2018, Inan2011]. This means that for all temporal derivatives, the following discrete approximations have to be used:

$$\frac{\partial \mathrm{p}}{\partial t} \approx \frac{p_{i,j,k}^{n+1} - p_{i,j,k}^{n-1}}{2T} \tag{4.3}$$

$$\frac{\partial^2 \mathrm{p}}{\partial t^2} \approx \frac{p_{i,j,k}^{n+1} - 2p_{i,j,k}^{n} + p_{i,j,k}^{n-1}}{T^2} \tag{4.4}$$

For spatial derivatives, the following substitutions should be used:

$$\frac{\partial \mathrm{p}}{\partial x} \approx \frac{p_{i+1,j,k}^{n} - p_{i-1,j,k}^{n}}{2X} \tag{4.5}$$

$$\frac{\partial^2 \mathrm{p}}{\partial x^2} \approx \frac{p_{i+1,j,k}^{n} - 2p_{i,j,k}^{n} + p_{i-1,j,k}^{n}}{X^2} \tag{4.6}$$

Equations for y and z spatial components are analogous. The only difference is a change of discrete spatial index on the right-hand side of equation. For the y component, equation is calculated for the *j* index and the z component for the *j* component.

Equation 4.2 after such substitution of presented discrete approximation takes the following form:

$$\begin{aligned}\frac{p_{i,j,k}^{n+1} - 2p_{i,j,k}^{n} + p_{i,j,k}^{n-1}}{T^2} = c^2 \Bigg[& \frac{p_{i+1,j,k}^{n} - 2p_{i,j,k}^{n} + p_{i-1,j,k}^{n}}{X^2} \\ & + \frac{p_{i,j+1,k}^{n} - 2p_{i,j,k}^{n} + p_{i,j-1,k}^{n}}{X^2} \\ & + \frac{p_{i,j,k+1}^{n} - 2p_{i,j,k}^{n} + p_{i,j,k-1}^{n}}{X^2} \Bigg],\end{aligned} \tag{4.7}$$

where *n* denotes the discrete index on the time axis, i, j, k denote the indices on the successive spatial components x, y, and z, T means the distance between consecutive discrete simulation points on the time axis, and X means the analogous distance between the points on the digitized spatial components.

To obtain a formula useful for implementation in a computer program, it is necessary to further transform this equation according to the variable $p_{i,j,k}^{n+1}$. The result of such transformation takes the form:

$$p_{i,j,k}^{n+1} = \lambda^2\left[p_{i+1,j,k}^{n} + p_{i-1,j,k}^{n} + p_{i,j+1,k}^{n} + p_{i,j-1,k}^{n} + p_{i,j,k+1}^{n} + p_{i,j,k-1}^{n} - 6p_{i,j,k}^{n}\right] + 2p_{i,j,k}^{n} - p_{i,j,k}^{n-1}, \quad (4.8)$$

where $\lambda = cT/X$ and it is called the Courant number. Its value affects the stability of the numerical method, which is FDTD. In the case of all calculations shown in this thesis, the optimal value of the Courant number is chosen to be used. The aforementioned value of Courant number is equal to $\sqrt{1/3}$ and this is the maximum possible value that still guarantees the stability of the method [Bergman2018, Kowalczyk2009].

Eq. 4.8 can be used together with a Dirichlet boundary condition applied to selected regions of the computational domain. It is also necessary to derive equations implementing the behavior of points on the edges of the computational domain, which also can be achieved by both implementing the Dirichlet or Neumann boundary condition [Kowalczyk2009].

To employ Dirichlet boundary conditions, one can simply force selected nodes of the computational domain to an arbitrary value (for instance, zero pressure for nodes associated with a rigid object). This, however, limits the simulation to only perfectly rigid objects.

It would be useful to introduce boundary conditions associated with objects whose acoustic properties are described by certain values of acoustic impedance (or admittance). This can be achieved by applying Neumann boundary conditions. The exact value of the acoustic impedance (or admittance) may be calculated from the desired reflection coefficient, as it can be calculated by taking into account the following relationship [Kim2010]:

$$R(\theta, \phi) = \frac{Z_w \cdot \cos(\theta) \cdot \cos(\phi) - 1}{Z_w \cdot \cos(\theta) \cdot \cos(\phi) + 1} \quad (4.9)$$

which defines the value of the reflection coefficient $R(\theta, \phi)$ for the incidence angles with the values θ and ϕ. It is, therefore, possible to define in the simulation the values for the perpendicular reflection coefficient $R(0,0)$, which can be converted into material impedance by the relationship [Blauert2009]:

$$Z_w = \frac{1 + R(0,0)}{1 - R(0,0)}. \quad (4.10)$$

The Neumann boundary condition has the following form [Botts2014]:

$$\frac{\partial p}{\partial t} + \frac{c}{\beta}\nabla_n p = 0, \quad (4.11)$$

where:

β is the specific acoustic admittance of the surface,

∇_n denotes a derivative along with a given vector *n,* which is normal to the surface to which the boundary condition is being applied.

The boundary condition has to be applied to the discretized form of the acoustic wave equation. Therefore, the boundary condition equation also has to be discretized. A specific type of discretization of spatial derivatives has to be employed in this step. Discretization has to be centered on velocity nodes. In the calculations presented, no values of acoustic velocity are calculated, but still, values of pressures obtained through the equations proposed are centered on hypothetical velocity nodes if the grid containing both the pressure and velocity nodes is considered. The velocity-centered approximation of spatial derivative has the following form:

$$\frac{\partial p}{\partial x} = \frac{p^n_{i+1,j,k} - p^n_{i,j,k}}{X}. \tag{4.12}$$

Substitutions for the y and z dimensions have analogous form, and the only change is the application of the formula to the discrete index *j* in the case of the y dimension and discrete index *k* in the case of the z dimension. After substitution, we obtain the following discretized form of a boundary condition:

$$\frac{p^{n+1}_{i,j,k} - p^{n-1}_{i,j,k}}{2T} + \frac{c}{\beta}\frac{p^n_{i+1,j,k} - p^n_{i,j,k}}{X} = 0. \tag{4.13}$$

After rearrangement, the discretized Neumann boundary condition takes the following form:

$$p^{n+1}_{i,j,k} - p^{n-1}_{i,j,k} + \frac{2\lambda}{\beta}\left(p^n_{i+1,j,k} - p^n_{i,j,k}\right) = 0. \tag{4.14}$$

It was shown by Webb and Bilbao [Webb2011] that the boundary condition given by Eq. 4.14 could be applied to the discretized acoustic wave equation in such a manner that the discrete update equation for the pressure matrix can be uniform across all computation domain. This is an important feature if GPU-accelerated computing is considered. It allows employing an identical calculation procedure for each computational unit in the GPU. The first step in this kind of application of the boundary condition is the assumption that for all borders to which the Neumann boundary condition is applied, at least one of the statements below is true:

$$p^n_{i+1,j,k} = 0, \qquad p^n_{i,j+1,k} = 0, \qquad p^n_{i,j,k+1} = 0. \tag{4.15}$$

In practice, this means that nodes on the border of the computational domain, as well as nodes that are inside objects submerged in the computational domain, have to be forced to always contain the value of 0. This makes it possible to modify Eq. 4.14 to the following form:

$$p_{i,j,k}^{n+1} - p_{i,j,k}^{n-1} - \frac{2\lambda}{\beta} p_{i,j,k}^{n} = 0. \tag{4.16}$$

which can be rearranged to:

$$p_{i,j,k}^{n} = \frac{\beta}{2\lambda}\left(p_{i,j,k}^{n+1} - p_{i,j,k}^{n-1}\right) \tag{4.17}$$

It is important to remember that Eq. 4.17 is true only if the assumption about forcing zero values on the border of the computational domain and the inside of scattering objects embedded into this domain is true. On the contrary case, the model will be unstable.

Eq. 4.17 can be used to obtain discretized acoustic wave equation with an identical form for all GPU calculation nodes. It can be substituted into Eq. 4.8 to get the following equation:

$$\begin{aligned} p_{i,j,k}^{n+1} = \lambda^2 \Big[& p_{i+1,j,k}^{n} + p_{i-1,j,k}^{n} + p_{i,j+1,k}^{n} + p_{i,j-1,k}^{n} + p_{i,j,k+1}^{n} + p_{i,j,k-1}^{n} \\ & - \left(6 - I_{i,j,k}\right) p_{i,j,k}^{n} - I_{i,j,k} \frac{\beta}{2\lambda}\left(p_{i,j,k}^{n+1} - p_{i,j,k}^{n-1}\right) \Big] + 2p_{i,j,k}^{n} - p_{i,j,k}^{n-1} \end{aligned} \tag{4.18}$$

Eq. 4.17 can be substituted into Eq. 4.18 multiple (from 1 to 3) times depending on how many planes in the vicinity of a computational node are forcing the boundary conditions. The number of substitutions is denoted by I. If there is only one plane, i.e. in case if the node is a neighbor of a flat plate, only one substitution is performed if the node is positioned on the edge of two walls (or in the corner of a 2D computational domain), then two substitutions are made as there are two planes affecting the propagation of acoustic waves. Finally, if the node is positioned in the 3D corner of the computation domain (i.e., a room corner), then $I = 3$, and three substitutions are performed. Eq. 4.18 can be simplified by introducing the following terms:

$$K_{ijk} = 6 - I_{ijk} \tag{4.19}$$

$$\mathrm{BK}_{i,j,k} = \left(6 - K_{ijk}\right)\lambda\beta/2$$

After such substitutions, Eq. 4.18 takes the following form:

$$p_{i,j,k}^{n+1}\left(1+\mathrm{BK}_{i,j,k}\right) = \left(2-K_{ijk}\lambda^2\right)p_{i,j,k}^{n}+\left(\mathrm{BK}_{i,j,k}-1\right)p_{i,j,k}^{n-1}+\lambda^2\left(p_{i+1,j,k}^{n}+p_{i-1,j,k}^{n}+p_{i,j+1,k}^{n}+p_{i,j-1,k}^{n}+p_{i,j,k+1}^{n}+p_{i,j,k-1}^{n}\right), \tag{4.20}$$

which is a version of the acoustic wave equation proposed by Webb and Bilbao in their work on acoustic wave equation tailored to computations on the GPU [Webb2011].

One important fact in the case of using Eq. 4.20 is a necessity to enforce zero pressure on the boundary of the computational domain and inside of scattering objects placed inside the domain. Another requirement is a specification of K matrix which contains information about all boundary conditions inside the domain. It always has to be generated at the beginning of calculations based on the computational domain shape and geometry of scattering obstacles. An example of such **K** matrix for the 2D domain is depicted in Fig. 4.1.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	0
0	5	6	6	6	5	5	5	5	5	5	6	6	6	6	6	6	5	0
0	5	6	6	5	0	0	0	0	0	0	5	6	6	6	6	6	5	0
0	5	6	6	5	0	3	4	4	4	5	6	6	6	6	6	6	5	0
0	5	6	6	5	0	0	0	0	0	5	6	6	6	6	6	6	5	0
0	5	6	6	5	0	0	4	5	5	5	6	6	6	6	6	6	5	0
0	5	6	6	5	0	0	5	6	6	6	6	6	6	6	6	6	5	0
0	5	6	6	6	5	5	6	6	6	6	6	6	6	6	6	6	5	0
0	5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	5	0
0	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 4.1. Example values contained in the **K** matrix. The computational domain has special 0 values informing the computational algorithm that nodes with that value of K are subject to enforcing the zero value of acoustic pressure. Inside the domain, a scattering obstacle is placed.

The last important aspect which has to be taken into account while preparing an FDTD-based numerical simulation is the injection of excitation signal into the domain. If one would simply set a value of pressure in an arbitrarily chosen point of computational grid, such as in the formula below:

$$p_{i,j,k}^{n+1} = p_{\mathrm{ext.}}^{n+1}, \tag{4.21}$$

then it would also automatically mean enforcement of Dirichlet boundary condition in this point. This would happen due to the fact that a value resulting from the propagation of acoustic waves into a node, which is a source for the excitation signal, is

systematically overwritten by consecutive values of the excitation signal. It forces this component of pressure in the source node to zero, which effectively means the application of the Dirichlet boundary condition. Such a situation leads to the formation of unwanted reflections of the wave from the source node. This kind of wave source is called a “hard source” [Cox2017].

To mitigate this problem, instead of overwriting the value of pressure in the source node, one can add consecutive values of excitation signal to the current value of pressure in the source node according to the following formula:

$$p_{i,j,k}^{n+1} = p_{i,j,k}^{n} + p_{\text{ext.}}^{n+1}. \tag{4.22}$$

This kind of wave source formulation does not impose an additional Dirichlet boundary condition on the node applied and mitigates the problem of reflections generated by the acoustic wave source. It is called a “soft source.” It should be mentioned that despite the advantage of not generating reflections, this kind of source is associated with a roll-off in lower frequencies as a result of interaction between such source equation and wave equation [Sheaffer2014].

Another important phenomenon related to the excitation signal is numerical dispersion. In simulations employing finite-element approximations, excitation of different frequencies does not travel with the same frequency. The higher the frequency, the more prominent this phenomenon is [Cox2017]. Due to this fact, it is important to use a sufficiently high sampling rate compared to traditional sampling rates used in audio processing, such as frequencies of 72 or 90 kHz, which permit accurate simulation of acoustic bandwidth up to 8 kHz. This is a reason why impulses used as excitations in the case of simulations employing FDTD should be bandlimited. An example of such impulse may be a Gaussian impulse given by a formula:

$$f^n = A_s \cdot \exp\left(\frac{T^2[n - n_0]^2}{2\sigma^2}\right), \tag{4.23}$$

where:

f^n is a current sample of the excitation signal,

A_s is the amplitude of the excitation signal,

n_0 is the number of samples by which the impulse is delayed,

σ is variance if the impulse, which is related to its bandwidth and can be used to impose a frequency limit.

A cutoff frequency for such band-limited excitation impulse is given by the formula below:

$$f_c = \frac{\sqrt{2ln2}}{2\pi\sigma} \tag{4.24}$$

Additionally, the excitation signals should be processed by a DC blocking filter, which prevents exciting the computation grid in the near-DC frequencies, which can also cause unwanted artifacts.

An example of acoustic wave propagation simulation in a close room performed using Eq. 4.20 is depicted in Fig. 4.2. As the equation used in this simulation requires that boundaries of the computation domain have Dirichlet or Neumann condition applied to it, reflections from boundaries of the computational domain are unavoidable and are visible in the simulation outcome.

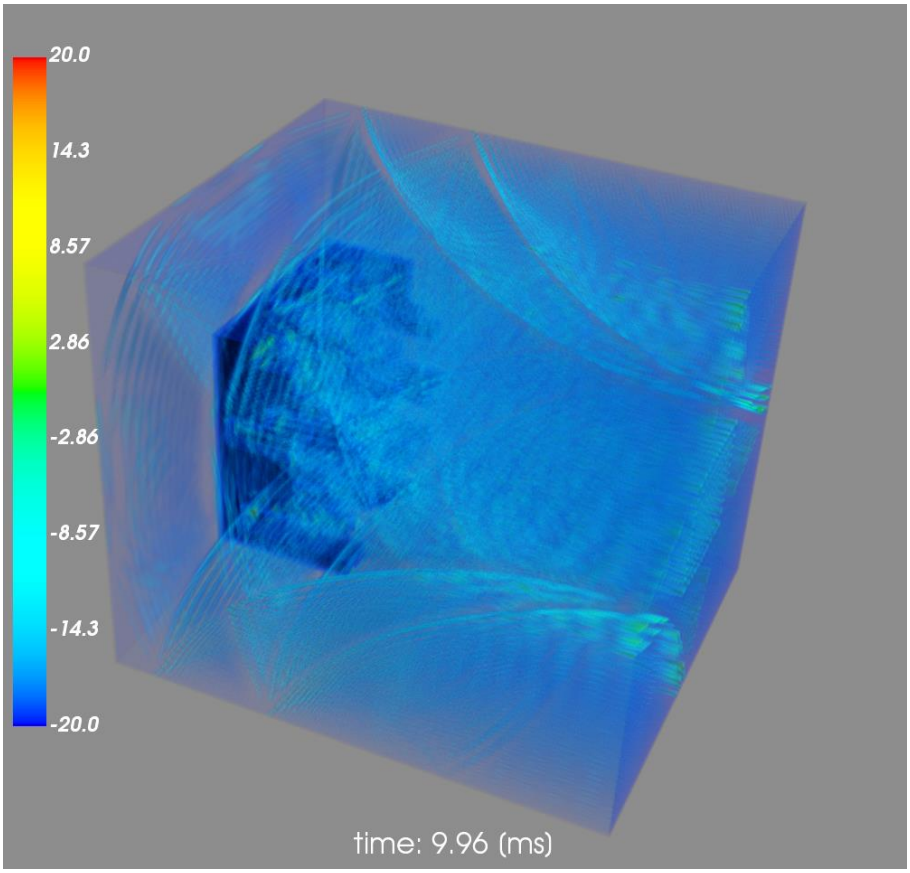


Fig. 4.2. Simulation of acoustic pressure wave propagation in a closed room in which a Schroeder 2D diffuser was placed. As Eq. 4.20 requires the application of Dirichlet or Neumann condition to boundaries of the computation domain, acoustic waves reflect from these boundaries and can be seen in the final simulation result. A script used for the calculation of this simulation was programmed in Python programming language.

4.2. Simulation of anechoic conditions and perfectly matched layers

A special kind of task in numerical simulation is the application of the Sommerfeld condition to achieve anechoic propagation in a numerical simulation. Such a goal may be obtained, for instance, by the use of perfectly matched layers (PML) [Alles2011, Berenger1994, Collino1998, Kim2019]. Utilizing such layers in the simulation of anechoic conditions in a numerical simulation is necessary to avoid unwanted reflections from the boundaries of the computational domain. There are several ways to implement anechoic conditions in a numerical simulation, and each of them is associated with the introduction of a different set of equations used for calculations within the computer simulation [Alles2011, Bermudez2007, Chern2019, Liu1997, Nataf2013, Zampolli2008]. The particular choice often is influenced by the problem to be solved with the use of numerical simulation utilizing PMLs.

One of the simple methods of implementation of PMLs is the modification of equations for the propagation of the acoustic waves by introducing a damping factor. Such change is applied to the area of the computational domain close to the border of the computational area, i.e., walls of the virtual representation of the simulated room or the border of the 2D representation of such a space. A graphic depiction of this principle is presented in Fig. 4.3. The use of PML is necessary for numerical estimation of all parameters of scattering devices requiring assessment of polar response such as reflection or diffusion coefficients of Schroeder diffusers.

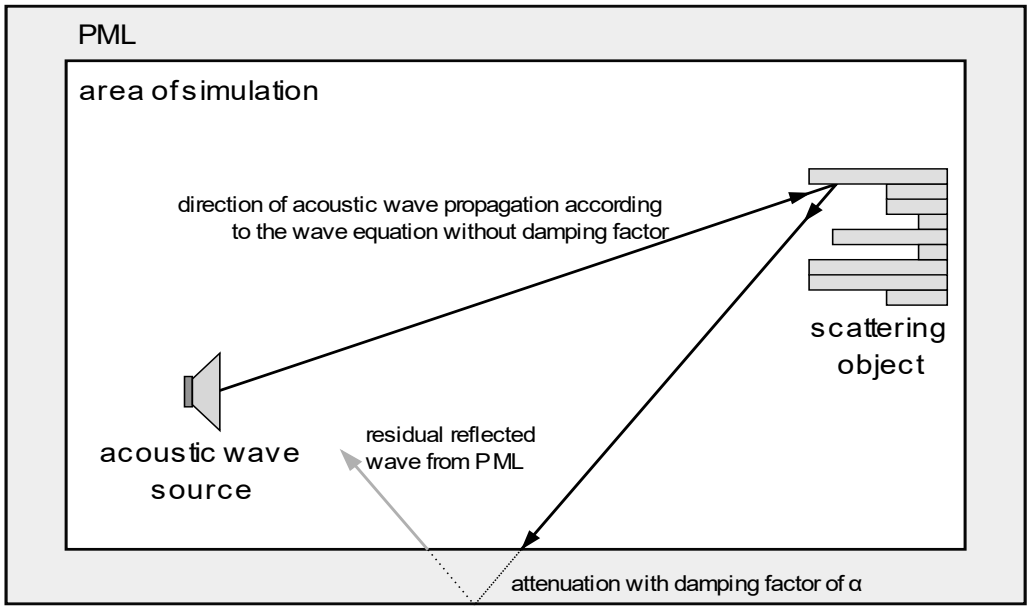


Fig. 4.3. Diagram of the numerical domain employing PMLs to reduce reflections from the boundaries of the computational domain.

The introduction of PMLs into a numerical domain requires modification of equations guiding acoustic wave propagation in the area of PMLs. The first step of the derivation of such layer equations is using a variant of the wave equation, taking into account losses in a propagation medium [Blackstock2000, Vandekerckhove2016]. A

starting point for such a problem are acoustic wave equations for the homogenous lossy medium [Yuan1997]:

$$\begin{cases} \nabla p + \rho \dfrac{\partial \boldsymbol{u}}{\partial t} + \alpha^* \boldsymbol{u} = 0 \\ \nabla \mathbf{u} + \dfrac{1}{\rho c^2} \dfrac{\partial \mathrm{p}}{\partial \mathrm{t}} + \alpha \mathrm{p} = 0 \end{cases} \tag{4.25}$$

where:

$\boldsymbol{u}$ is an acoustic velocity vector,

α is an acoustic compressibility-related attenuation factor, which is typically referred to as an attenuation factor if the propagation of acoustic waves in the air is considered,

α^* is an attenuation factor related to a so-called “mass-proportional” damping, which is typical for sound propagation in solids and is usually zero for propagation in the air.

Such kind of a PML is called the Berenger PML [Berenger1994]. From the numerical point of view, it is useful to employ both α and α^*As the PML relies on high damping factors and is intended to mitigate reflected waves fully. Therefore the use of a non-physical coefficient such as α^* is possible if it makes it easier to attenuate further residual waves reflected from the PML.

It would be desirable to transform Eq. 4.26 to the form of the second-order equation. It would allow optimizing memory usage on a computer performing simulation as only data associated with acoustic pressure should be kept and updated in the course of the simulation. Therefore, a set of transformations can be carried out:

$$\begin{cases} \nabla^2 p + \rho \dfrac{\partial \nabla \boldsymbol{u}}{\partial t} + \alpha^* \nabla \boldsymbol{u} = 0 \\ \dfrac{\partial \nabla \boldsymbol{u}}{\partial t} + \dfrac{1}{\rho c^2} \dfrac{\partial^2 \mathrm{p}}{\partial \mathrm{t}^2} + \alpha \dfrac{\partial \mathrm{p}}{\partial \mathrm{t}} = 0 \end{cases} \tag{4.27}$$

Next, two substitutions can be performed to obtain:

$$\nabla^2 p - \rho \left(\frac{1}{\rho c^2} \frac{\partial^2 \mathrm{p}}{\partial \mathrm{t}^2} + \alpha \frac{\partial \mathrm{p}}{\partial \mathrm{t}} \right) - \alpha^* \left(\frac{1}{\rho c^2} \frac{\partial \mathrm{p}}{\partial \mathrm{t}} + \alpha \mathrm{p} \right) = 0 \tag{4.28}$$

After rearranging, one obtains the following relationship:

$$c^2 \nabla^2 p = \frac{\partial^2 p}{\partial t^2} + \left(\rho c^2 \alpha + \frac{1}{\rho} \alpha^* \right) \frac{\partial p}{\partial \mathrm{t}} + c^2 \alpha \alpha^* p \tag{4.29}$$

To further simplify the equation, it is possible to introduce the following substitutions:

$$\alpha_A = \rho c^2 \alpha + \frac{1}{\rho}\alpha^* \tag{4.30}$$

$$\alpha_B = c^2 \alpha \alpha^* \tag{4.31}$$

This allows the derivation of a final form of Eq. 4.29:

$$c^2 \nabla^2 p = \frac{\partial^2 p}{\partial t^2} + \alpha_A \frac{\partial p}{\partial \mathrm{t}} + \alpha_B p \tag{4.32}$$

Next, the resulting partial differential equation has to be discretized in a similar manner as (4.2). After necessary substitutions, Eq. 4.29 takes the following form:

$$\begin{aligned} c^2 \Bigg(& \frac{p^n_{i+1,j,k} - 2p^n_{i,j,k} + p^n_{i-1,j,k}}{X^2} + \frac{p^n_{i,j+1,k} - 2p^n_{i,j,k} + p^n_{i,j-1,k}}{X^2} \\ & + \frac{p^n_{i,j,k+1} - 2p^n_{i,j,k} + p^n_{i,j,k-1}}{X^2} \Bigg) \\ & = \frac{p^{n+1}_{i,j,k} - 2p^n_{i,j,k} + p^{n-1}_{i,j,k}}{T^2} + \alpha_A \frac{p^{n+1}_{i,j,k} - p^{n-1}_{i,j,k}}{2T} + \alpha_B p^{n+1}_{i,j,k}. \end{aligned} \tag{4.33}$$

After simple transformations and rearranging, one can obtain the following equation:

$$\begin{aligned} p^{n+1}_{i,j,k} = & \left(1 + \frac{\alpha_A}{2}T + \alpha_B T^2\right)^{-1} \Big(\lambda^2 \left[p^n_{i+1,j,k} + p^n_{i-1,j,k} + p^n_{i,j+1,k} + p^n_{i,j-1,k} + p^n_{i,j,k+1}\right. \\ & \left. + p^n_{i,j,k-1} - 6p^n_{i,j,k}\right] + 2p^n_{i,j,k} - \left[1 + \frac{\alpha_A}{2}T\right] p^{n-1}_{i,j,k}\Big), \end{aligned} \tag{4.34}$$

which is an equivalent of Eq. 4.8 for the lossy propagation medium. This form of acoustic wave equation can be a basis for obtaining a version of Eq. 4.20 for the lossy propagation medium. The original formula does not permit to introduce losses, which makes it impossible to use PMLs while also employing a homogenous acoustic wave equation for calculation on GPU. For calculations performed in this work, a more general version of Eq. 4.20, which is capable of both employing the Berenger MPLs and performing calculations on GPU with a homogenous calculation procedure. To achieve this goal, the Neumann boundary condition has to be introduced into Eq. 4.34 in the same manner as it is introduced into an Eq. 4.8. After this substation, one obtains the following equation:

$$p_{i,j,k}^{n+1} = \left(1 + \frac{\alpha_A}{2}T + \alpha_B T^2\right)^{-1} \left(\lambda^2 \left[p_{i+1,j,k}^n + p_{i-1,j,k}^n + p_{i,j+1,k}^n + p_{i,j-1,k}^n + p_{i,j,k+1}^n + p_{i,j,k-1}^n - (6-I)p_{i,j,k}^n - I\frac{\beta'}{2\lambda}\left(p_{i,j,k}^{n+1} - p_{i,j,k}^{n-1}\right)\right] + 2p_{i,j,k}^n - \left[1 + \frac{\alpha_A}{2}T\right]p_{i,j,k}^{n-1}\right). \tag{4.35}$$

This can be further simplified by employing substitutions from Eq. 4.19:

$$p_{i,j,k}^{n+1} = \left(1 + B_K + \frac{\alpha_A}{2}T + \alpha_B T^2\right)^{-1} \left(\lambda^2\left[p_{i+1,j,k}^n + p_{i-1,j,k}^n + p_{i,j+1,k}^n + p_{i,j-1,k}^n + p_{i,j,k+1}^n + p_{i,j,k-1}^n\right] + [2 - K\lambda^2]p_{i,j,k}^n + \left[B_K - 1 - \frac{\alpha_A}{2}\right]p_{i,j,k}^{n-1}\right), \tag{4.36}$$

which is a form of the discretized wave equation for lossy propagation medium, which can be employed for computations employed on GPU and the use of the Berenger PMLs. To further simplify the computation procedure, it is possible to assume that $\alpha = \alpha^*$. This means that $\alpha_A = \alpha(\rho c^2 + 1/\rho)$, and $\alpha_B = c^2\alpha^2$. This allows passing to the GPU processor only one set of damping factor values, which are related to α. This reduces the amount of data needed to be sent during the program initialization phase. This form of the equation is employed in simulations for obtaining results presented in consecutive Sections of this doctoral dissertation. It is worth mentioning that all restrictions regarding the construction of $\mathbf{K}$ matrix and computational domain geometry necessary for Eq. 4.20 are also required for Eq. 4.36 to be stable.

In practice, the introduction of a lossy term into a wave equation means the need for an additional matrix used in the computation process. The α parameter characterizes a local loss factor, and if it is subject to rapid change, for instance, from 0 in areas that are not lossy to the value of 0.15 in the area of PML, it will cause a reflection. Due to this fact, it is necessary to introduce a matrix of α coefficient values, which gradually increase starting from the beginning of the PML layer up to a maximum level at the boundary of the domain. An example profile used for this purpose can have the following form [Cox2017]:

$$\alpha = \alpha_{max}\left|\frac{x - x_0}{x_{max} - x_0}\right|^n, \tag{4.37}$$

where:

x_0 is an initial point in the PML,

x_{max} is the last point in the PML,

α_{max} is a maximum value of damping eventually reached in point x_{max},

n is a number between 2 and 3.

An example of simulation analogous to one depicted in Fig. 4.2 but employing the Berenger PML to eliminate reflections from boundaries of the computational domain is shown in Fig. 4.4. This type of simulation is useful for evaluating such properties of acoustic diffusers as reflection and diffusion coefficient.

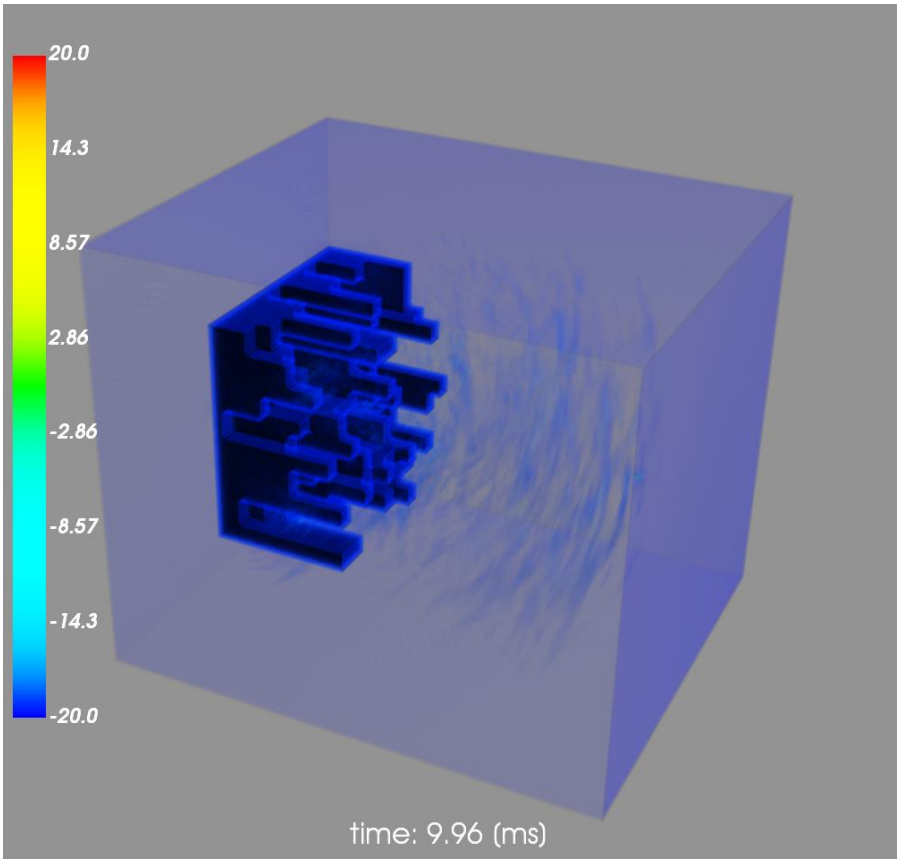


Fig. 4.4. Simulation of acoustic pressure wave propagation in a closed room in which PML is used to attenuate reflections from boundaries of the computational domain. A script used for the calculation of this simulation was prepared in the Python programming language

4.3. Calculation of basic propagation medium parameters

The behavior of acoustic waves governed by Eqs. 4.20 and 4.36 requires knowledge of constants describing the state of the propagation medium. Examples of such constants are specific acoustic impedance of air Z_{air}, or speed of acoustic wave propagation in air c. For most cases an averaged values of those parameters can be assumed, but in some applications, it may be useful to introduce exact values of parameters that are associated with properties of air such as air pressure, temperature, and relative humidity. This enables to relate given propagation conditions more intuitively to real-life conditions but also makes it easier to, e.g., design a simulation of sound propagation on a cold winter day. This may be even more visible in temperature gradient occurrence, which is an especially interesting phenomenon, as it can affect paths of sound wave propagation prominently.

The absolute value of specific acoustic air impedance is given by the following formula:

$$Z_{air} = \rho c, \tag{4.38}$$

where ρ is the density of air, and c is the speed of sound wave propagation in the medium. To calculate the air density, it is necessary to know the current atmospheric pressure, temperature, and relative humidity. The density of air is given by a formula below [Picard2008]:

$$\rho = \frac{PM_a}{\zeta RT_a}\left(1 - x_v\left(1 - \frac{M_v}{M_a}\right)\right), \tag{4.39}$$

where:

P is atmospheric air pressure,

M_a is the molar mass of dry air,

M_v is molar mass of water vapor,

x_v is the mole fraction of water vapor,

ζ is the compressibility factor of air,

R is the molar gas constant,

T_a is the air temperature (in Kelvins).

Value of x_v can also be expressed by the following fraction:

$$x_v = \frac{m_v}{m_v + m_a}, \tag{4.40}$$

where:

m_a is the amount of moles of air present in the propagation medium,

and m_v is the amount of moles of water vapor present in the propagation medium,

As in atmospheric conditions $\zeta \approx 1$, taking this into account Eq. 4.40 the following approximation of can be derived:

$$\rho = \frac{PM_a}{RT_a}\left(1 - \frac{m_v}{m_v + m_a}\left(1 - \frac{M_v}{M_a}\right)\right) = \frac{PM_a}{RT_a}\left(1 - \left(\frac{M_a m_v - M_v m_v}{M_a m_v + M_a m_a}\right)\right) \tag{4.41}$$

This expression can be further simplified to obtain the following result:

$$\rho = \frac{P}{RT_a}\frac{M_a m_a + M_v m_v}{m_v + m_a} = \frac{P(x_a M_a + x_v M_v)}{RT_a} = \frac{p_a}{R_a T_a} + \frac{p_v}{R_v T_a}, \tag{4.42}$$

where:

x_a is the mole fraction of air in the propagation medium,

p_a denotes partial pressure of air,

p_v denotes partial pressure of water vapor,

R_a is the specific gas constant of air, which is equal to 287.058 $\frac{J}{kg \cdot K}$,

R_v is the specific gas constant of water vapor, which is equal to 461.495 $\frac{J}{kg \cdot K}$,

The partial pressure of water vapor can be calculated from the following relation:

$$p_v = \mathrm{RH} \cdot p_{sat}, \tag{4.43}$$

where p_{sat} is saturation pressure of water vapor, which can be obtained from air temperature by employing the approximate Teten formula [Xu2012]:

$$P_{sat} = 610.78 \times 10^{\frac{7.5T_a}{T_a + 237.3}}, \tag{4.44}$$

After the value of air density is obtained, the value of sound wave propagation speed can be derived by employing the following formula [Feynman1965]:

$$c = \sqrt{\kappa \cdot \frac{P}{\rho}}, \tag{4.45}$$

where κ is the adiabatic ratio of specific heats of air equals 1.402.

An illustration of the influence of selected temperatures, pressures, and relative humidity values on acoustic properties of acoustic propagation medium is depicted in Tab. 2.1. It can be seen that if an extraordinary level of precision is needed, factors such as temperature, pressure, or humidity of air have to be taken into account, as their changes can significantly influence values of both the speed of acoustic waves propagation and specific impedance of air. The latter one can, e.g., have an influence on the reflection coefficient of objects placed in a computational domain in which FDTD simulation is carried out.

Tab. 4.1 Selected values of acoustic waves propagation speed and specific acoustic impedance for selected values of air temperature, atmospheric pressure, and relative air humidity.

T_a [℃]	P [hPa]	RH [-]	c [m/s]	Z_{air} [rayls]
-20	990	0.3	319.21	434.82
		0.5	319.22	434.80
		0.7	319.24	434.78
	1020	0.3	319.21	448.00
		0.5	319.22	447.98
		0.7	319.24	447.96
0	990	0.3	331.67	418.48
		0.5	331.75	418.39
		0.7	331.82	418.29
	1020	0.3	331.67	431.17
		0.5	331.74	431.07
		0.7	331.82	430.97
20	990	0.3	343.94	403.56
		0.5	344.25	403.19
		0.7	344.56	402.83
	1020	0.3	343.92	415.80
		0.5	344.22	415.44
		0.7	344.52	415.08

Results shown in this Section show that simulation can be capable of predicting the scattering properties of an acoustic treatment device. Equations presented in this Section were used to implement simulation procedures. The code of the aforementioned numerical procedures is shown in Appendices A and B. This can be achieved by a time-domain-based simulation of a sound field evolution in a given space. This space can be limited by computational domain boundaries – this would be a case of a room simulation, but thanks to the use of a modified version of equations proposed by the author of this thesis, it can also be possible to simulate the propagation of acoustic waves in anechoic conditions. Those two types of simulations were carried out in experiments used to prove theses of this dissertation, and they are described in Sections 5 and 6.

5. SIMULATION OF A MOCK-UP ROOM

In this Section, an example of optimization of an acoustic diffuser for use in enclosed spaces is shown. Calculations presented in the chapter were performed with the simpler version of an acoustic simulation model employing only equations from Section 4.1, and thus – without employing Berenger PML layers to mitigate reflections of excitation Gaussian impulse from the wall. Therefore, it will allow using a numerical model validated in the literature to evaluate how baseline Schroeder diffuser designs compare to designs obtained by means of optimization processes. This is also a reason why Berenger PML layers are not used, as the literature mainly covers the non-anechoic type of equations such as ones presented in Section 4.1, or ones modified to employ methods of computing FDTD other than a leapfrog one [Botts2014, Cox2017, Hamilton2017, Webb2011]. First, the definition of the problem is presented. It involves the dimension of a shoe-box type room, which is a real-live equivalent of a cuboid computational domain used in the simulation and optimized metric calculated for implementation of a diffuser in a room, which is the uniformity of a frequency response in a selected spot in the room. In the case of the experiment presented in this Section, the optimization metric also has to be other than the correlation diffusion coefficient proposed in Section 3.2. This is because the measurement of it requires anechoic conditions. Next, results obtained with the use of a classical, baseline approach based upon random sequences are shown. They are followed by the presentation of the results obtained by genetic algorithms and the ones derived from the reinforcement learning methods.

5.1. Definition of a mock-up acoustic room optimization problem and a baseline sound-treatment approach

In an experiment involving baseline techniques and optimization algorithms for diffuser design, a room mock-up has the following dimensions: 3 m in length, 2 m in width, and 2 m in height. It simulates the case of designing acoustic adaptation for an enclosed space to be used as a small control room or a listening room. A schematic representation of the room and the location of the essential simulation points are shown in Fig. 5.1. To provide a possible baseline solution for treating the proposed space, an acoustic diffuser design is to be proposed. To obtain the pattern of segments making a diffuser, the best design chosen from PRD, QRD, and MLS approaches are selected.

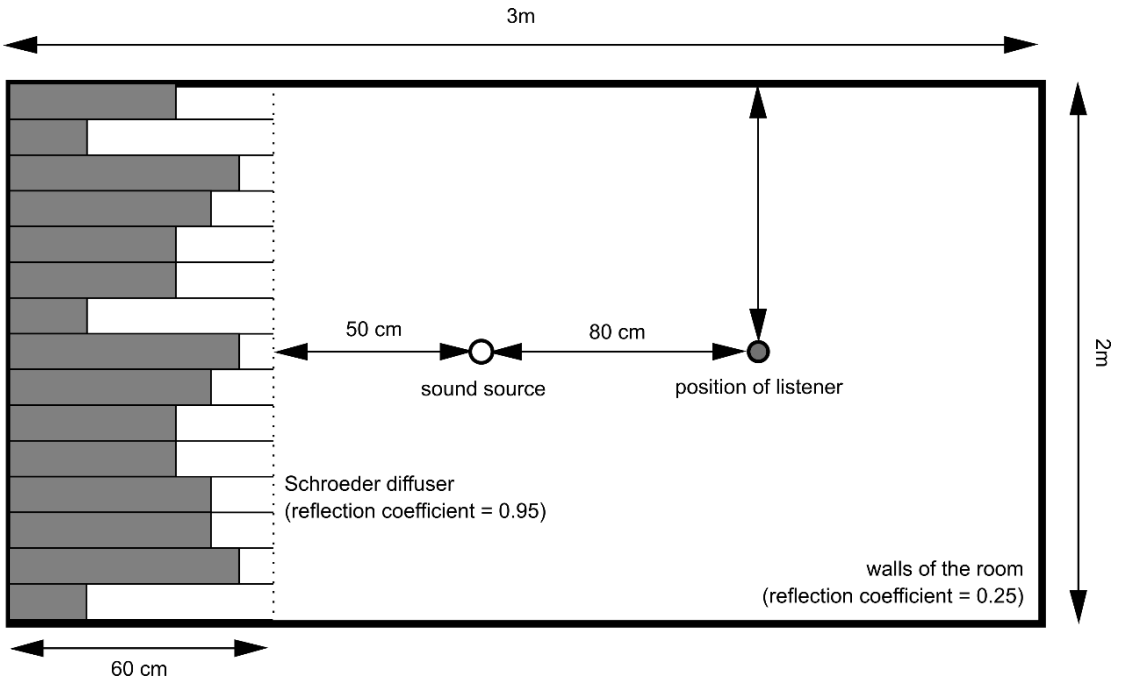


Fig. 5.1. Dimensions of a mock-up room and the presentation of the experiment simulation. The height of the room is equal to 2m.

It is assumed that the reflection coefficients $R(0,0)$ of the materials the walls are made of are 0.25. This corresponds to the situation indicating that a considerable amount of sound waves are mostly absorbed. Such amount of reflections does not allow simulation of anechoic conditions but allows considering the simulated space as a room that was acoustically treated to be used, i.e. as a control room or a place to watch multimedia. Absorbing properties of walls permit controlling the level of reflections propagating inside the simulated space as most of them originate from a diffuser whose material has a reflection coefficient of 0.95. Therefore, the simulated situation corresponds to a room adapted acoustically under the principle of room design so that reflection can be formed at one end and the other end is attenuated or suppressed (live-end, dead-end, LEDE). This approach makes it easier to control the amount and type of acoustic wave reflections in a room since most of them come from the “live” edge of the room where the reflecting surfaces are located. To avoid the risk of undesirable distortions in the frequency response, an acoustic diffuser is placed at the reflective end of the room that scatters the acoustic waves, thereby reducing the risk of standing waves that could distort the frequency room characteristics.

Other assumptions concern the maximum depth of the diffuser wells, which is 60 cm. The diffuser width varies depending on the generated well system. Moreover, the depth of the well patterns generated by the QRD and PRD design techniques are looped by the maximum possible number of times, so the diffuser occupies the largest possible area of the front wall of the analyzed room. The listening point is placed 130 cm from the diffuser and is located equidistant from the side walls of the room. The room is excited by a Gaussian pulse with a band-limited to 8 kHz. The sampling rate of signals obtained from the simulation is 48 kSa/s. Therefore, with the Courant number

equals $\sqrt{1/3}$, the spatial resolution of the simulation is 1.23 cm. The effect of the simulation is the impulse response of the room at the listening position. To assess the quality of diffusers, it is also possible to simulate a room response without an acoustic diffuser, which makes it possible to determine the impact of the diffuser on the acoustics of the simulated room.

The optimization of the acoustic adaptation consists in the appropriate selection of the depth system of the diffuser cavities located on the front wall of the room so that the frequency response of the room at the listening point is as flat as possible. Therefore, in the design of QRD, PRD diffusers, and diffusers optimized by an optimization algorithm, it is necessary to formulate a criterion function that will allow mathematically assessing how flat the frequency response of a room is for each of the tested patterns. The quality assessment of individual diffuser designs is carried out by the measure based on the calculation of the standard deviation in successive sub-bands of the frequency characteristics of the tested room at the listening point. This measure is calculated using Eq. 5.1:

$$\sigma_{f\in<f_{li};f_{ui})} = \frac{\mathrm{std}_{f\in<f_{li};f_{ui})}\mathrm{FFT}\big(p(n)\big)}{\mathrm{std}_{f\in<f_{li};f_{ui})}\{\mathrm{FFT}(p_r(n))\}}, \tag{5.1}$$

where:

$\sigma_{f\in<f_{li};f_{ui})}$ is the measured power level of the frequency band from f_{li} to f_{ui},

$\mathrm{std}_{f\in<f_{li};f_{ui})}$ means the operation of calculating the standard deviation for the values associated with this band,

$p[n]$ denotes the impulse response of a room with a diffuser,

$p_r[n]$ means the impulse response of a room without a diffuser.

The second impulse response is a reference for the values obtained with the diffuser located in the simulated room. The smaller the value of the measure, the better the diffuser design is tested. The simulation is carried out in eight frequency bands: 100Hz-250Hz, 250Hz-500Hz, 500Hz-1kHz, 1kHz-2kHz, 2kHz-3kHz, 3kHz-4kHz, 4kHz-5kHz, 5kHz-6kHz. The overall rating of each prototype, taking into account each of the analyzed sub-bands, is in turn calculated according to Eq. 5.2 and is denoted by the symbol σ_D:

$$\sigma_D = -\sum_i \sigma_{f\in<f_{li};f_{ui})} \tag{5.2}$$

Calculations were made using a program prepared in the Python programming language described in Section 4.1. Calculations were accelerated using a graphics card using CUDA architecture. The room geometry was mapped using a point matrix in three-dimensional space. An example of a cross-section through a computational

domain that maps a room with the diffuser inside is shown in Fig. 5.2.

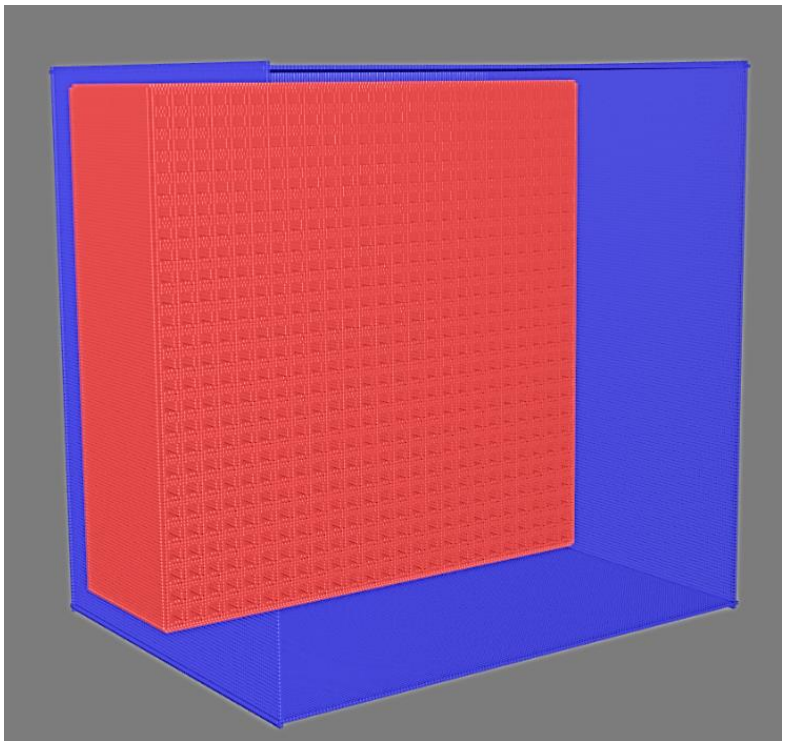


Fig. 5.2. Illustration of the grids of computational nodes (defined by the $\boldsymbol{K}$ matrix) employed in the simulation, nodes forming the simulated shape of the acoustic diffuser are marked in red. The blue shape is a cross-section through the area of nodes representing the shape of the simulated room. For clarity of presentation, the nodes of the propagation medium were not highlighted in color. They are located inside the area enclosed by the nodes of the room walls.

The proposed room geometry is subject to an acoustic treatment process employing the use of acoustic diffusers designed by genetic and reinforcement learning-based optimization methods. To be able to compare those methods to standard approaches used to modify acoustic of such spaces as one presented in this Section, a baseline approach has to be introduced. This baseline method involves the use of PRD and QRD design methodologies to obtain diffuser designs. In the experiment, the FDTD simulation was used to assess the quality of each of the diffuser types tested. In the case of QRD diffusers, 42 geometries were examined, from among which the ten best designs with the lowest value of the parameter σ_D were selected. Similarly, the ten best PRD diffuser designs were chosen. This selection was made from 118 pseudo-random sequences. The differences in the number of groups from which the best acoustic diffuser designs were selected result from the fact that the maximum value of the prime number for which the pseudo-random sequences were calculated was limited. It has been assumed that QRD sequences based on prime numbers in the range from 5 to 199 will be used. This allows reducing the part of the generated sequences that has a structure growing following the progress of the square, which appears because these values have already reached higher values than the assumed first value. Therefore the operation of finding the remainder from division present in Eq. 3.1 does not affect them. Similarly, for the PRD sequence, generation

to the range of prime numbers from 5 to 47 was limited. Sequences were generated for each primary element associated with the used prime number.

5.2. Evaluation of acoustic diffuser designs generated by the baseline techniques and genetic algorithm

The third group of diffusers refers to the geometries calculated by the genetic algorithm. It operated for 8 hours, and during this time, 30 generations of projects were produced. The ten best projects appeared between generations. The same number of projects signalized the end result of the genetic algorithm. Each diffuser consisted of 25 columns and rows. Its dimensions were selected each time so that it filled in the wall on which it was placed to the maximum. An example of a visualization of the simulation carried out for the diffuser generated by the genetic algorithm is presented in Fig. 5.3. It depicts a 2D cross-section of the computational domain during the evaluation of an acoustic diffuser in a shoebox-type room.

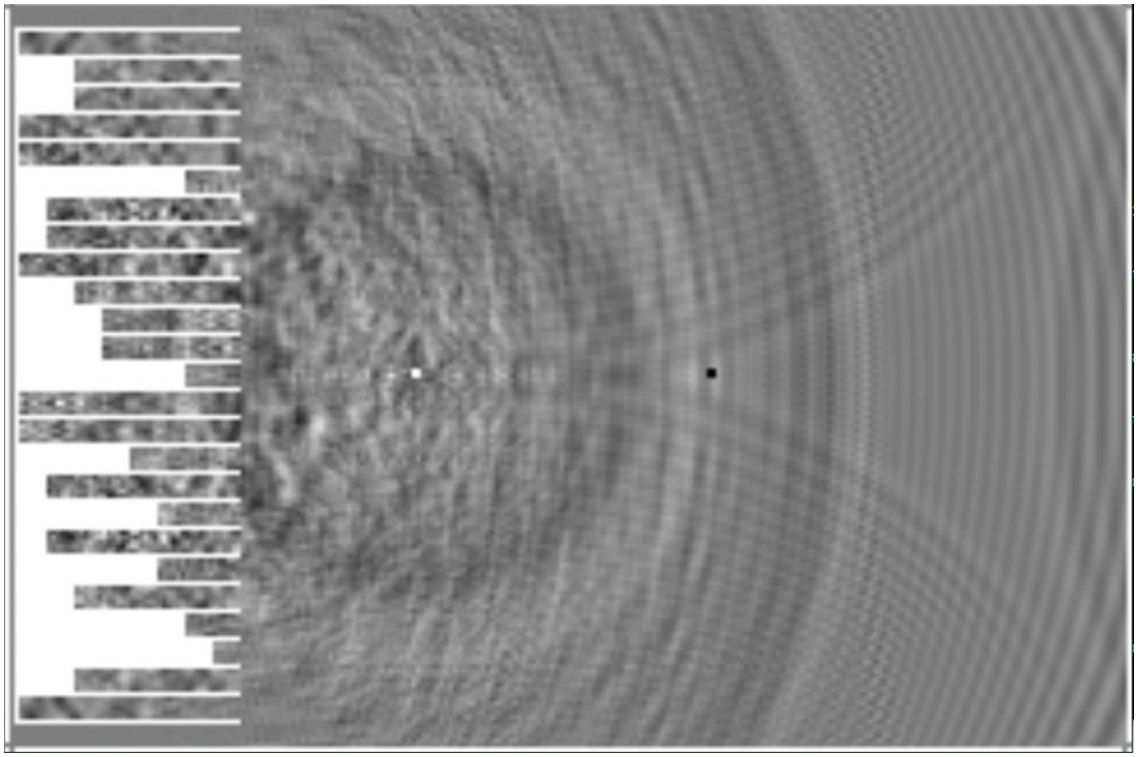


Fig. 5.3. An example of a simulation frame showing a cross-section (from above) through a simulated room. The position of the source of the acoustic wave is marked with a white dot, the position of the listener is marked with a black dot. The image depicts a horizontal cross-section of the 3-dimensional computational domain used in the simulation.

This Section presents an evaluation of acoustic diffuser designs generated by using baseline (QRD and PRD) methods and the genetic algorithm. The second one refers to the assessment of acoustic diffuser designs generated by the reinforcement learning algorithms. All the results are presented in terms of their statistical properties, i.e., the Kruskal-Wallis and Dunn's post-hoc tests are applied for that purpose.

The easiest way to assess the quality of diffusers is to analyze the value of the σ_D fitness function, which is also a measure of the frequency response homogeneity at the listening position. These values are collected in Tab. 5.1. They were calculated for the top 10 diffusers in each of the three diffuser types tested and sorted by increasing values of the evaluation function. The values of the σ_D function indicate that the best properties in terms of flatness of the frequency characteristic are diffusers designed by the genetic algorithm. The second in terms of the result obtained is the best diffuser design obtained by the QRD design method, and one obtained employing the PRD algorithm was the worst.

The frequency characteristics associated with the best-performing diffusers from each group, along with the simulation of the reference measurement in a room without a diffuser, are shown in Fig. 5.4. The diffusers have a similar effect on the frequency range below 1.5 kHz. Significant differences begin to appear in the frequency range from 1.5 kHz to 5 kHz. Within this range, two narrow frequency attenuation ranges can be identified for the QRD diffuser. The results of the PRD diffusers and those designed by the genetic algorithm are similar. They require in-depth analysis using statistical methods to draw meaningful conclusions about which diffuser has better properties. For this purpose, the Kruskal-Wallis statistical test was used.

Tab. 5.1 Values of the fitness function obtained for the best projects of QRD, PRD diffusers and generated by the genetic algorithm. The values are multiplied by -1 with respect to Eq. 5.2. A smaller value means a more even frequency response (the smaller the metric, the better result).

No.	**QRD**	**PRD**	**genetic**
1	5.36	4.78	3.75
2	5.69	4.98	3.83
3	5.77	5.07	3.93
4	6.04	5.25	3.94
5	6.20	5.25	3.95
6	6.22	5.26	4.00
7	6.40	5.36	4.03
8	6.44	5.36	4.05
9	6.52	5.37	4.09
10	6.70	5.58	4.20

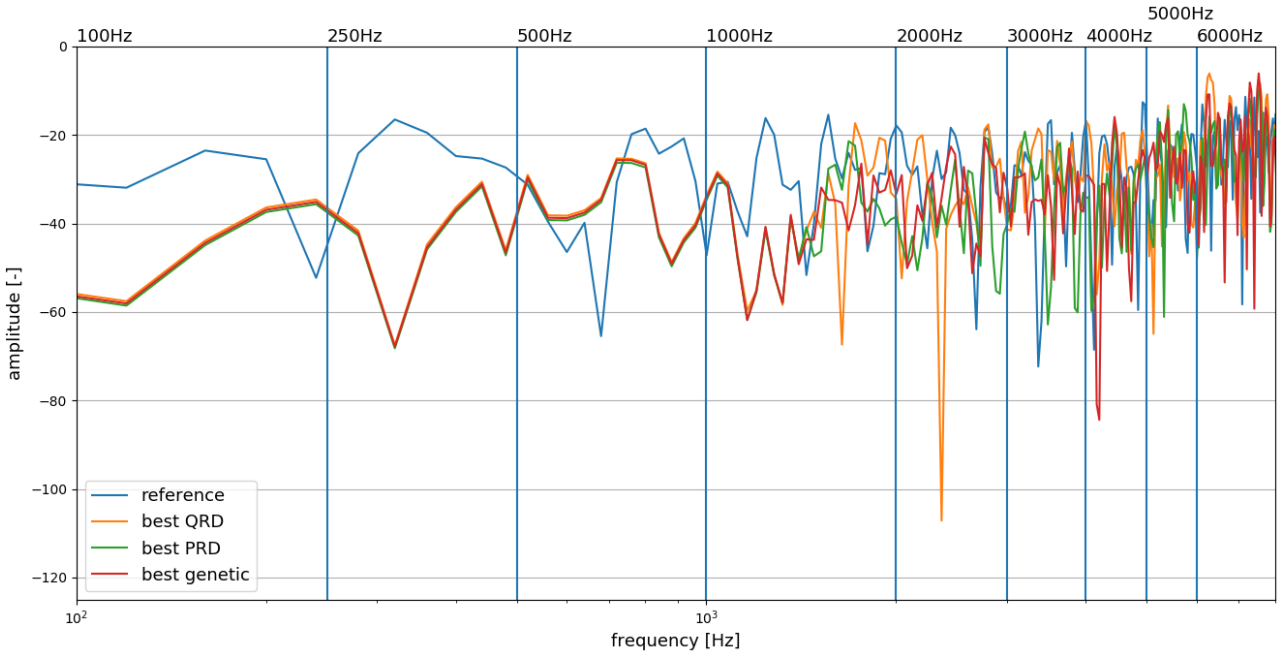


Fig. 5.4. Frequency responses of diffusers designed by three tested methods.

The input for the statistical test was a series of differences between the values of the σ_D fitness function for a diffuser designed by the QRD or PRD method and the result of the genetic algorithm. The test analyzed both partial values from which the σ_D measure was calculated, as well as its value, including all analyzed frequency bands. The test input data were the values of the σ_D function and its components obtained for the top 10 diffusers from each of the subgroups. For the analysis of the resultant σ_D values, the test input values correspond to those in Tab. 5.1.

Besides, a series of 10 zeros, which is the reference series, has been included in the test. If the average difference in the value of the function σ_D is statistically significantly smaller than the value of the sequence of 10 zeros, it can be concluded that the diffuser generated by the genetic algorithm showed better properties. In the opposite case - the diffuser designed by classical methods has better properties. If there are no statistically significant differences, the sequence of differences has an average value of zero - the solutions are the same in terms of how even the room frequency response is. In the case of comparisons with both QRD and PRD diffusers, the value of the Kruskal-Wallis test statistic was less than 10^{-3}, so it can be concluded that at least part of the series of differences between diffuser groups has an average value significantly different from zero. Dunn's post-hoc test was conducted to check which of these differences were statistically significant. Its results are visualized in Figs. 5.5 and 5.6. Differences that were not statistically significant were represented by gray bars. Each bar also has a *p*-value returned by Dunn's test. A negative difference value is in favor of the diffuser generated by the genetic algorithm.

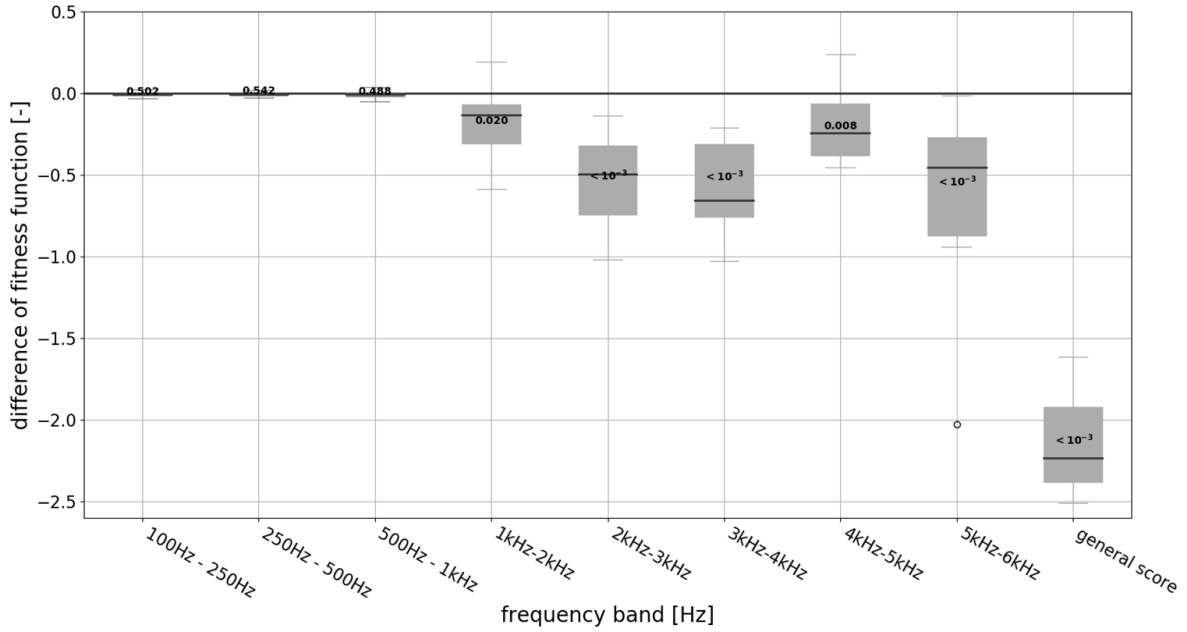


Fig. 5.5. Results of the statistical test comparing solutions obtained using a genetic algorithm with a diffuser based on the QRD sequence.

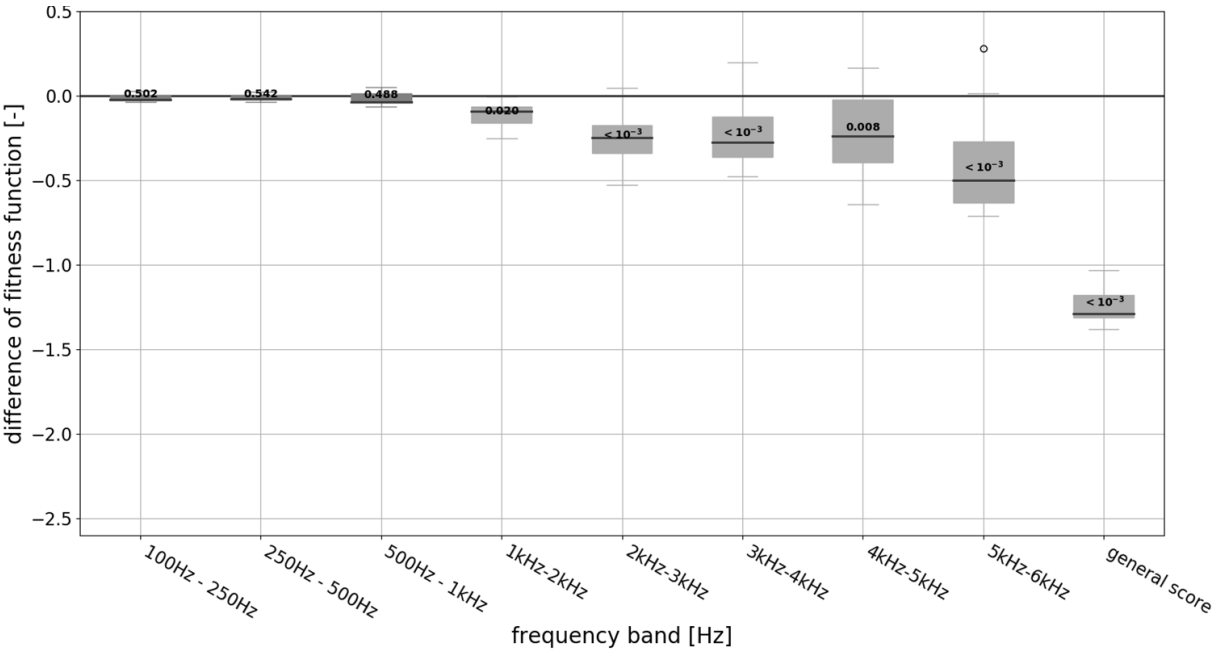


Fig. 5.6. The difference in diffuser ratings along with the result of a statistical test comparing solutions obtained using a genetic algorithm with a diffuser based on the PRD sequence.

The values obtained using Dunn's test show that statistically significant differences in the impact of diffusers on the uniformity of the room frequency characteristics appear for frequency bands higher than 1 kHz. The most noticeable differences are visible in the 2–3 kHz and 3–4 kHz bands. The values of differences in this range are also characterized by low variance compared to higher frequencies. This is especially evident in the results obtained for comparison with PRD diffusers. The differences in

the σ_D measure, marked in the drawings as a global assessment, also proved statistically significant.

The baseline approach using QRD and PRD sequences has the advantage that their use is very fast compared to the genetic algorithm solution. Their disadvantage is, in turn, that the solution designed using pseudo-random sequences may not be optimal in the context of the given listening room in which the designed diffuser is to be installed. This hypothesis was confirmed by conducting an experiment based on a room simulation following the LEDE principle. The genetic algorithm generated such a depth system of cavities in an acoustic diffuser located on the front wall of the room, which provided the smallest standard deviation of frequency characteristics in 8 frequency bands in the range from 100 Hz to 6 kHz. The optimization algorithm presented in Section 3.3 generated a diffuser geometry design, which, in the context of the proposed measure of success related to room response linearization at the listening point, behaved better than reference geometries prepared using QRD and PRD techniques. However, it should be remembered that the genetic algorithm is a time-consuming solution comparing to the very fast process of generating pseudo-random sequences. Additionally, computer simulation, even using FDTD techniques accelerated with graphics cards, requires at least several hours of calculations for the algorithm to test and match the appropriate solution. For this reason, the genetic algorithm is best suited to difficult or atypical cases when designing acoustic adaptation using classical methods is impossible or significantly hindered.

The results obtained in this experiment confirm thesis no. 1, which states that it is possible to employ numerical simulation as a means of calculating fitness, reward function, or similar optimization metric and achieve satisfying results.

Also, thesis no. 2 was proven, as both optimization methods were able to provide solutions that were associated with higher scores assigned to them by a simulation than the baseline approach based on pseudo-random sequences.

Another advantage of the genetic algorithm is the ability to focus on achieving the best effect at a specific point in the room being tested and conducting the optimization process with a measure of success corresponding to the selected criterion. This criterion can be defined in a way tailored to a given situation and the implementation of the diffuser. This means that it can be better suited to the needs existing in a specific room area. Not without significance is the fact that the criterion function can be used to optimize the value resulting from many different criteria. This may correspond, for example, to the situation when the selected room parameter is optimized in more than one point or when the algorithm simultaneously optimizes the values of two or more parameters characterizing the room acoustics.

Compared to the aforementioned work by Patraquim [Patraquim2017], the algorithm proposed in this Section showed measurable performance in a similar range of frequencies analyzed by the simulation algorithm. When calculating the metric determining the unevenness of the frequency response, the algorithm proposed achieved this effect after 30 iterations. This coincides with the observations made in

the publication of Patraquim [Patraquim2017]. However, an exact comparison of the results is not entirely possible because, in the cited publication, the diffusivity factor of the adopted measurement frequencies is the optimized metric. In the approach proposed in this doctoral dissertation, the frequency response is analyzed, which facilitates, e.g., a precise definition of the frequency band in which the algorithm introduces the best improvement measured by increasing the uniformity of the frequency response in this band.

It is also worth noting that computer simulation is a good way to pre-check the behavior of many different types of diffusers without performing many measurement trials. For this reason, it can work as a method of pre-selection of the most promising geometries of acoustic wave obstacles, which can then be tested in real conditions in an anechoic chamber and at the site of application. This procedure reduces the amount of time and financial costs associated with the preparation and testing prototypes of devices for testing.

5.3. Evaluation of acoustic diffuser designs generated by the reinforcement learning algorithms

In this Section, QRD and PRD baseline techniques are to be employed for the performance comparison with reinforcement learning algorithms. Architectures of convolutional neural networks employed for this purpose, i.e., a deep dueling Q neural network (DDQN) and a (DPG) neural network, are depicted in Figs. 5.7 and 5.8.

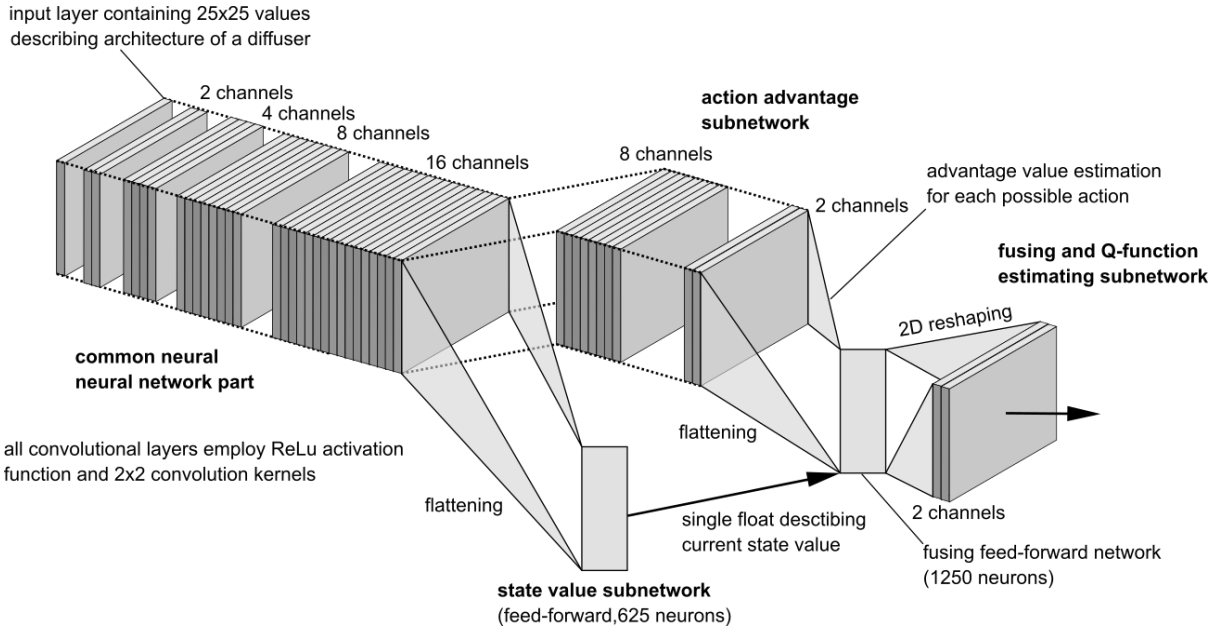


Fig. 5.7. The architecture of a deep dueling Q neural network (DDQN) used in the experiment.

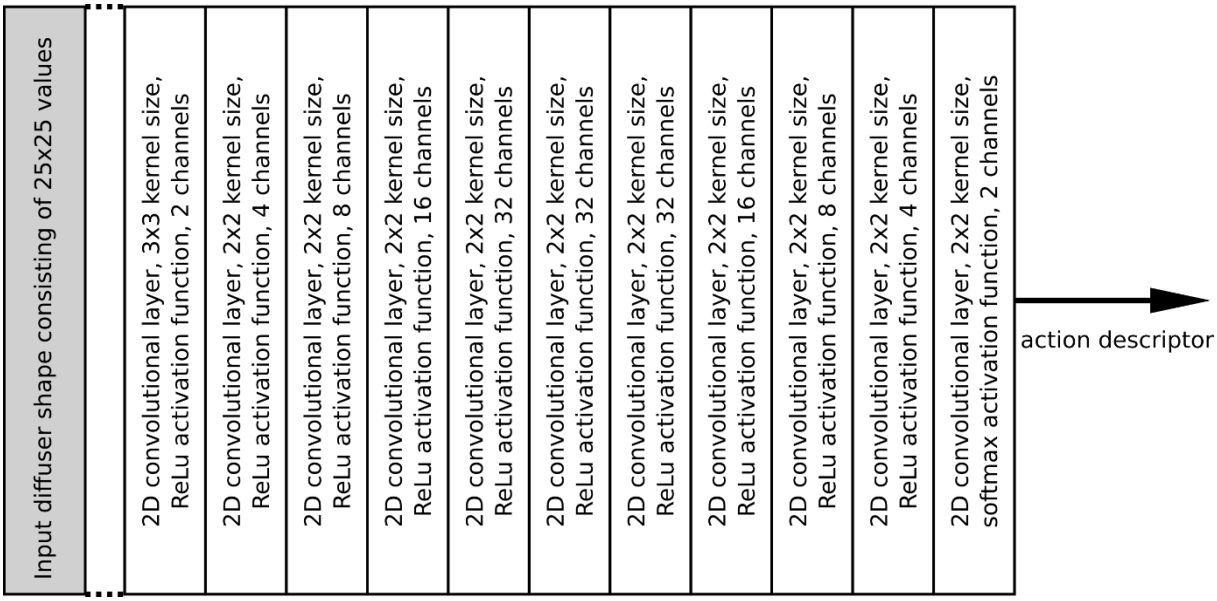


Fig. 5.8. The architecture of a convolutional neural network used as a policy approximator in a deep policy-gradient algorithm.

The DDQN neural network returns a matrix of action values, DPG returns a matrix of probabilities of choosing each action. In the case of DDQN, an action with the highest Q value is selected. Contrary, in the case of DPG, the action is chosen randomly with respect to the probability distribution obtained from the policy neural network. The final action is encoded and executed in a form visible in Fig. 5.9.

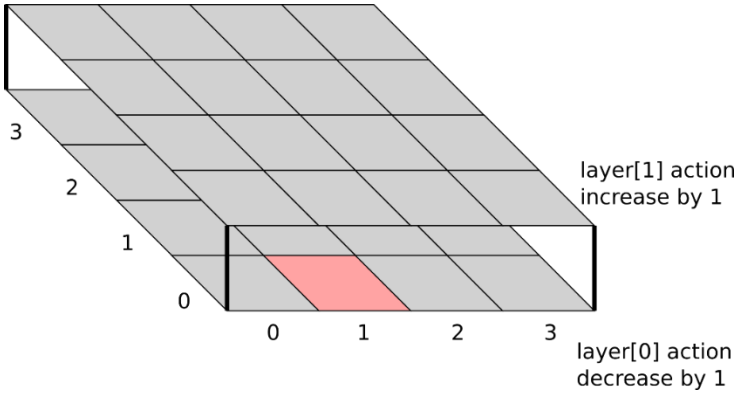


Fig. 5.9 Encoding decision derived from the Q-matrix generated by DDQN and retrieved from the output of the DPG neural network. Gray indices contain values equal to 0; the red one has a value of 1 and indicates the action to be taken by the algorithm.

For each of the reinforcement learning algorithms, training lasted for 10 hours, which let to perform 64 training episodes. Each episode contained 100 steps involving choosing a particular decision regarding changes in the design of the diffuser. The fitness function of the best design found in each episode was stored for the purpose of algorithm performance tracking. The best 10 values of a fitness function for each of the baseline and reinforcement learning algorithms are provided in Tab. 5.2. Calculations were performed with the use of the Nvidia GeForce RTX2080 Ti graphics card. It was used for both calculations related to machine learning and FDTD-based simulation of

the diffuser behavior in the room.

Tab. 5.2 Values of the fitness function obtained for the best projects of QRD, PRD diffusers, and those generated by deep dueling Q-network (DDQN) and deep policy-gradient (DPG) reinforcement learning algorithms. The values are multiplied by -1 with respect to Eq. 5.2. A smaller value means a more even frequency response.

No.	**QRD**	**PRD**	**DDQN**	**DPG**
1	5.36	4.78	3.65	4.57
2	5.69	4.98	4.00	4.58
3	5.77	5.07	4.33	4.86
4	6.04	5.25	4.77	5.15
5	6.20	5.25	4.90	5.24
6	6.22	5.26	5.3	5.25
7	6.40	5.36	5.43	5.29
8	6.44	5.36	5.44	5.50
9	6.52	5.37	5.63	5.55
10	6.70	5.58	5.74	5.59

Results from Tab. 5.2 are presented in the form of a box plot in Fig. 5.10. Visually, the performance of the proposed reinforcement learning algorithms is similar to one of the PRD sequence-based diffusers. However, an inter-quartile interval is larger than a PRD-related one in both the case of DDQN and DPG algorithms.

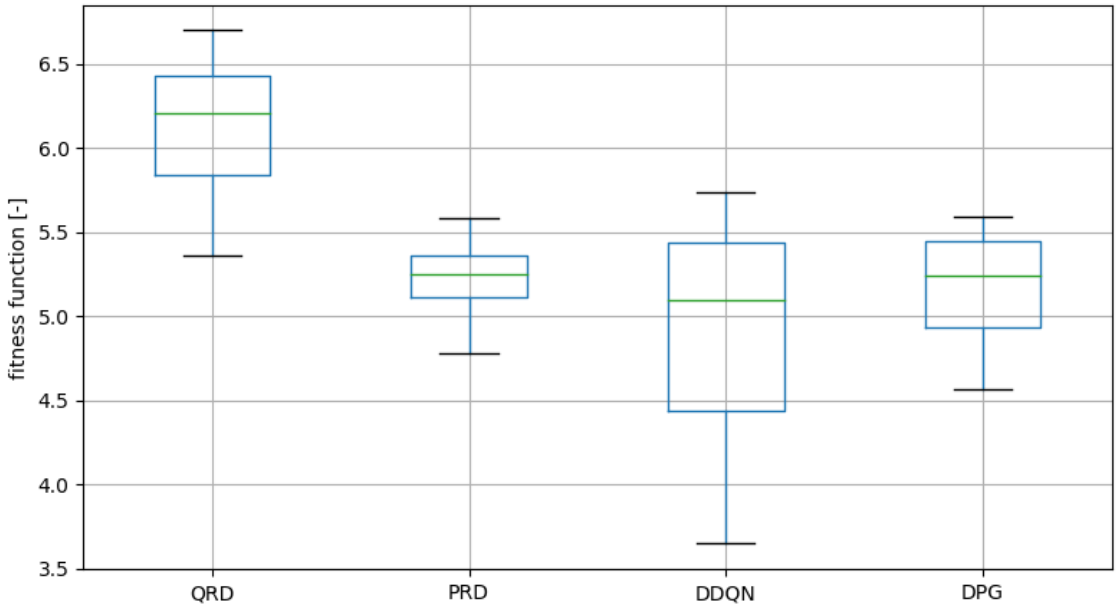


Fig. 5.10. Boxplot representing the fitness of diffusers generated by reinforcement learning algorithms employed in the experiment.

To analyze the statistical significance of differences observed in Tab. 5.2 and Fig. 5.10, a series of statistical tests was conducted. The significance coefficient was assumed to have a typical value of 0.05. First, the normality of distributions of fitness functions of best designs was tested. To achieve this, a Shapiro-Wilk test for normality of statistical data distribution was conducted. Next, the Holm-Bonferroni correction was applied to mitigate the effects of multiple testing problems; therefore corrected *p*-values of the Shapiro-Wilk test are given in Tab. 5.3.

Tab. 5.3 Corrected *p*-values of the Shapiro-Wilk test for normality of distributions of fitness function values produced by each tested algorithm.

	QRD	PRD	DDQN	DPG
p-value	0.801	0.801	0.686	0.536

Each *p*-value was greater than the threshold of 0.05, a null hypothesis was not rejected in any of the analyzed cases. As a result, all algorithms are assumed to produce values of fitness having Gaussian distribution.

The next step is the test for equality of variance. This is checked with the use of the Levene test for equality of variance of multiple data sets. In the case of diffuser-related data, the value of the Levene test statistic is equal to 4.041, and hence, the *p*-value of the test is equal to 0.014. The *p*-value is lesser than a significance threshold of 0.05; therefore, the null hypothesis of the Levene test stating that all variances that are equal to each other have to be rejected. Due to the fact that variances are not equal, the ANOVA test for equality of mean values of fitness function values cannot be performed,

and the Kruskal-Wallis statistical test has to be conducted instead.

The value of the Kruskal-Wallis test statistic is equal to 18.972, which gives a *p*-value of 0.00027. Therefore, the medians of data sets are not equal. To find which differences are significant, the Dunn post-hoc test was conducted. The resulting *p*-values are presented in 5.4

Tab. 5.4 Values of Dunn's post-hoc test comparing medians of best fitness function values generated by the algorithms investigated.

	QRD	PRD	DDQN	DPG
QRD		$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
PRD	$< 10^{-3}$		0.744	0.833
DDQN	$< 10^{-3}$	0.744		0.908
DPG	$< 10^{-3}$	0.833	0.908	

In the boxplot presented in Fig. 5.10, it is visible that the variance of results is visually greater in both the DDQN and the DPG algorithms than in PRD one. To test this hypothesis, two additional Levene tests were conducted to compare the variances of PRD-DDQN and PRD-DPG algorithms. *P*-values of that comparison were corrected with the Holm-Bonferroni correction and were both lesser than 10^{-3}. Therefore, it can be concluded that both algorithms had a greater variance of results than the PRD algorithm.

Results of statistical tests allow conclusions that in terms of the median of performance, the PRD, DDQ, and DPG algorithms performed similarly. However, DDQN and DPG algorithms had a greater variance of fitness function values. It can be seen that both of them achieved some designs which had the fitness of significantly lower value when compared to any of the designs created by the PRD algorithms. Therefore, while the average performance is similar, both the DDQN and DPG algorithms have the chance to find a solution that is significantly better than any design generated by the PRD algorithm. On the other hand, the consistency of the design quality is greater for the PRD algorithm.

This way, thesis no. 1 of this doctoral dissertation was proved, as the proposed optimization algorithms employed a numerical simulation to perform the optimization process.

Moreover, thesis no. 2 was also proven, as both the genetic and reinforcement learning algorithms were capable of finding designs having better properties (in terms of the best-obtained uniformity of a frequency response of a sound-treated room) as designs obtained from baseline methods (such as QRD, and PRD methods).

6. EVALUATION OF ACOUSTIC DIFFUSERS GENERATED BY OPTIMIZATION ALGORITHMS IN ANECHOIC CONDITION

Algorithms proposed in the thesis were found to improve simulated uniformity of frequency response of a given room. A standard method of evaluating the performance of a diffuser is a measurement of a diffusion coefficient. Also, the autocorrelation diffusion coefficient can be calculated based on a polar response of a given acoustic diffuser. This is also a common way of characterizing the scattering properties of diffusers for the purpose of selecting appropriate diffusers for the given application. Therefore it is crucial to determine if genetic and reinforcement learning algorithms can optimize the diffusion coefficient and design diffusers better than the ones obtained by simple randomization of lengths of Schroeder diffuser segments. A simulation employing PML (perfectly matched layers) layers should be used to achieve this goal, as anechoic conditions are necessary for both simulation-based prediction and measurement of diffusion coefficient. The experiment meant to investigate the performance of the aforementioned design principles is schematically depicted in Fig. 6.1.

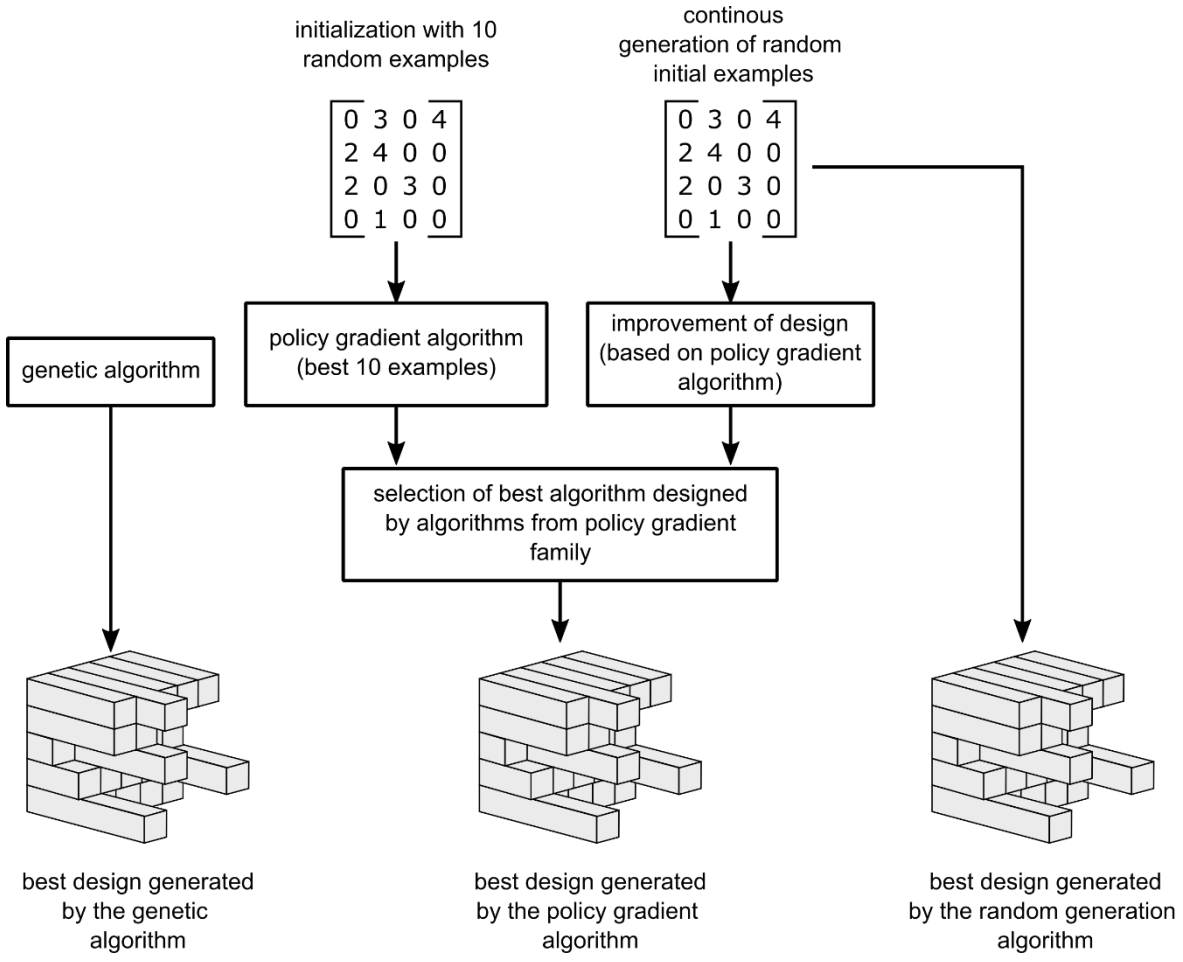


Fig. 6.1 Diagram of the experiment conducted for testing the simulation-based optimization and comparing outcomes of this method with a process of acoustic diffuser design based on a random selection of the diffuser segment height.

There are four methods of acoustic diffusers design investigated:

- random selection of the length of diffuser pattern segments,
- optimization of the length of diffuser pattern segments employing genetic algorithms,
- optimization of the length of diffuser pattern segments employing deep policy gradient algorithm, which optimizes patterns generated randomly,
- optimization of the length of diffuser pattern segments employing a deep policy gradient algorithm, which optimizes patterns sampled from 10 best designs generated by the algorithm at the moment of creation of the initial pattern.

As can be seen in Fig. 6.1, the DDQN algorithm was not included in this experiment. The reason for this is a specific modification of action space which allows faster modification of generated designs. As was shown in the experiment with optimization of the frequency response of a mock-up shoebox-type room, one of the fundamental assumptions on all reinforcement learning algorithms optimizing diffuser patterns is a type of action space in which each reinforcement learning agent is operating. For the aforementioned experiment, an action space that involves a change in only one segment of the diffuser pattern was investigated. This allows the use of both the DDQN and DPG algorithms to optimize the acoustic diffuser patterns. Still, a major drawback of this approach is the optimization process requiring simulation after the change of every single segment of a diffuser. As the simulation is the most time-consuming step of the optimization process, such a necessity of frequent simulations significantly slows down the process of diffuser design. To speed up computations, the action space was modified to make it possible for the agent to change multiple elements at once. Visualization of an example action after such a modification is depicted in Fig. 6.2. The structure of the modified DDQN neural network is shown in Fig. 6.3.

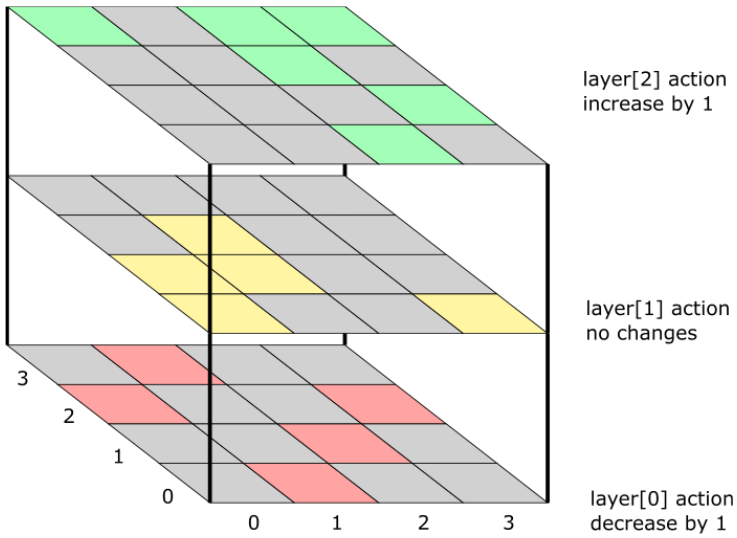


Fig. 6.2 Modified structure of action being an effect of inference carried out by the DPG neural network

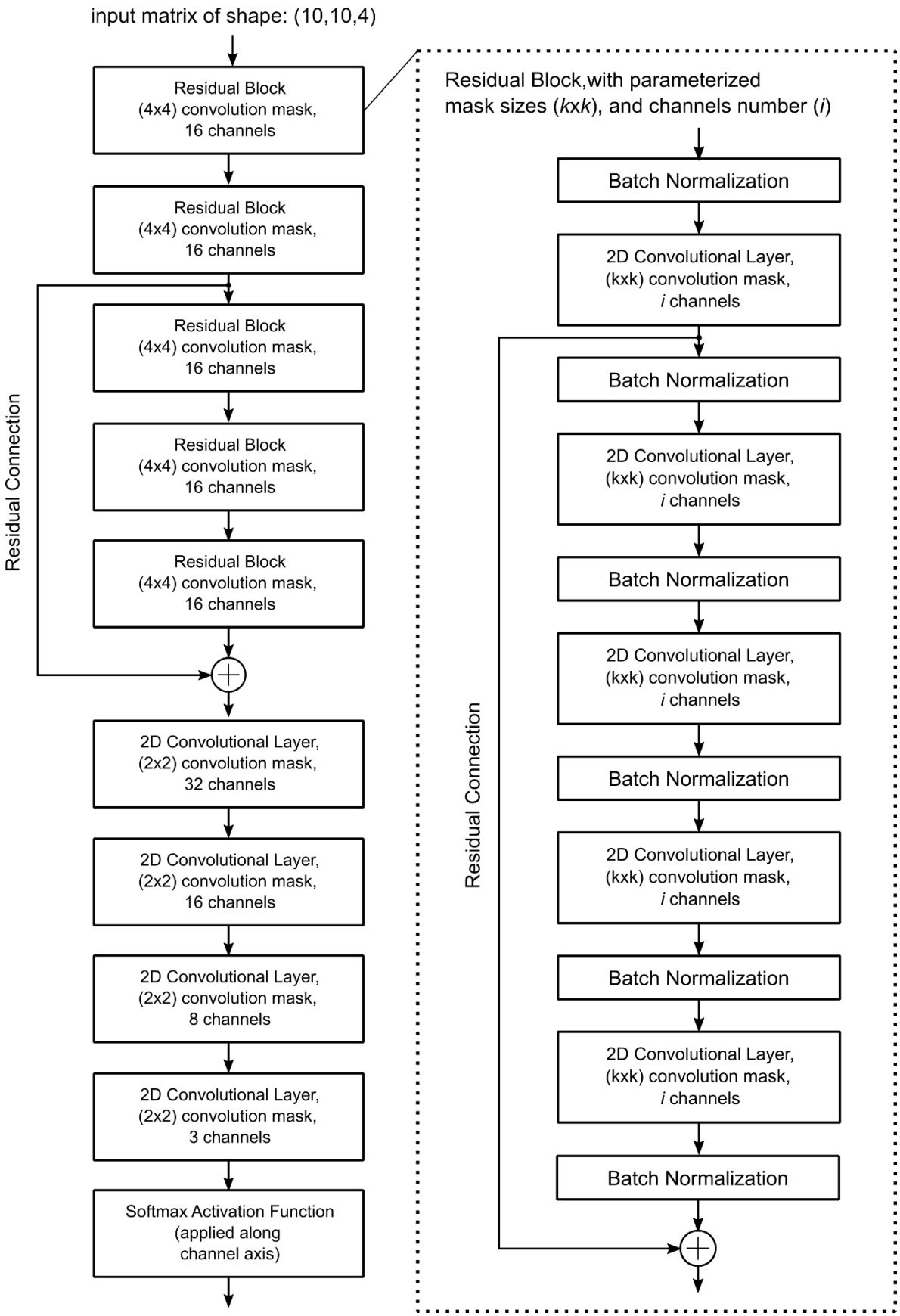


Fig. 6.3. Structure of a DPG neural network that learns and performs a strategy of a reinforcement learning agent optimizing designs in the experiment. The activation function of all layers but the last, softmax one, is a parameterized rectified linear unit (PReLU).

Designs in the experiment have equal width and height patterns, being 10 segments, and for each segment, three decisions can be made. The decisions are:

- decreasing the element height by 1,
- leaving the element unchanged,
- increasing the element height by 1.

As only heights from 0 up to 10 are accepted, the values are trimmed after modifications introduced to the pattern by the neural network to avoid negative segment heights and heights greater than 10. This means that there are $3^{10 \cdot 10} = 3^{100}$ possible actions in the action space. Such a large number of actions makes it impossible to employ the DDQN algorithm for the optimization task, as the output vector length would also be equal to 3^{100}. Therefore in the final experiment, only random, genetic algorithm and DPG-based approaches are compared.

As denoted in Fig. 6.3, the input to the DPG neural network was also modified, additional channels were added. Instead of providing only the information about the diffuser pattern, the neural network also takes:

- 2-dimensional FFT of the pattern,
- 2-dimensional cepstrum of the pattern,
- the autocorrelation of the pattern interpolated from the shape of $(19,19)$ to the shape of $(10,10)$.

In the case of autocorrelation, a spline-based interpolation is implemented in the *RectBivariateSpline* class from the Python *scipy.interpolate* package was used. The version number of SciPy library employed for this experiment was 1.5.4. The cepstrum of the Schroeder diffuser pattern was calculated according to the formula below:

$$\mathbf{X}_{\mathbf{cepstrum}} = \mathrm{FFT2D}(|\mathrm{FFT2D}(\mathbf{X})|) \tag{6.1}$$

where:

$\mathbf{X}$ is a matrix containing the pattern of the Schroeder diffuser,

$\mathrm{FFT2D}$ is a 2-dimensional fast Fourier transformation,

$\mathbf{X}_{\mathbf{cepstrum}}$ is a spatial cepstrum of $\mathbf{X}$.

The order of spline used for interpolation in both the x and y axes is 5. Such an approach allows the neural network to estimate the properties of diffusers by analyzing their shape in terms of spatial frequencies and autocorrelation. The use of cepstrum allows for easier sensitization of the neural network to take into account the harmonicity of spatial frequencies present in the design. An example of the parameterization result, together with the initial matrix containing the pattern definition of a diffuser, is shown in Fig. 6.4.

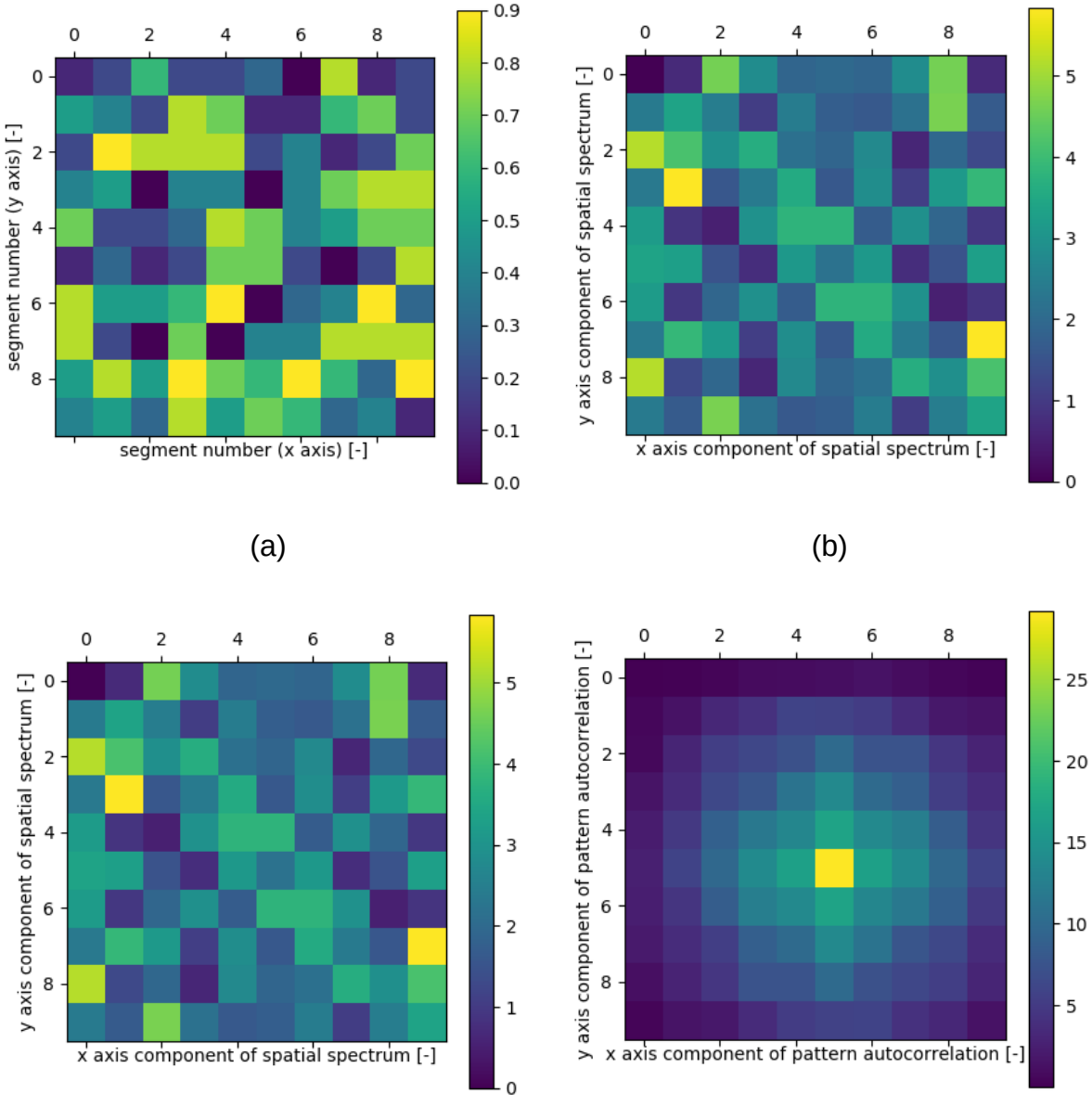


Fig. 6.4 Example of input fed into the neural network of the structure shown in Fig. 6.3. The original pattern matrix is presented in a), in b) one can see the spatial spectrum of the input pattern, c) depicts spatial cepstrum, and d) shows interpolated autocorrelation of the pattern.

Two approaches to DPG-based optimization were investigated:

1. the approach in which a neural network tries to improve wideband autocorrelation diffusion coefficient of randomly generated diffuser patterns,
2. the approach in which, through the whole process of training, the best 10 historical designs are retained and updated. A neural network tries to improve a design randomly chosen from the aforementioned set of best diffuser designs.

Neural networks are trained in a loop of interaction between the network and the environment. The algorithm of training is presented in Fig. 6.5. The environment, in this case, is an FDTD simulation of the diffuser interaction with a band-limited impulse. Due

to the necessity of limiting the numerical dispersion, the bandwidth of simulation was limited to 4 kHz – this is the maximum frequency of exciting impulse employed in the simulation.

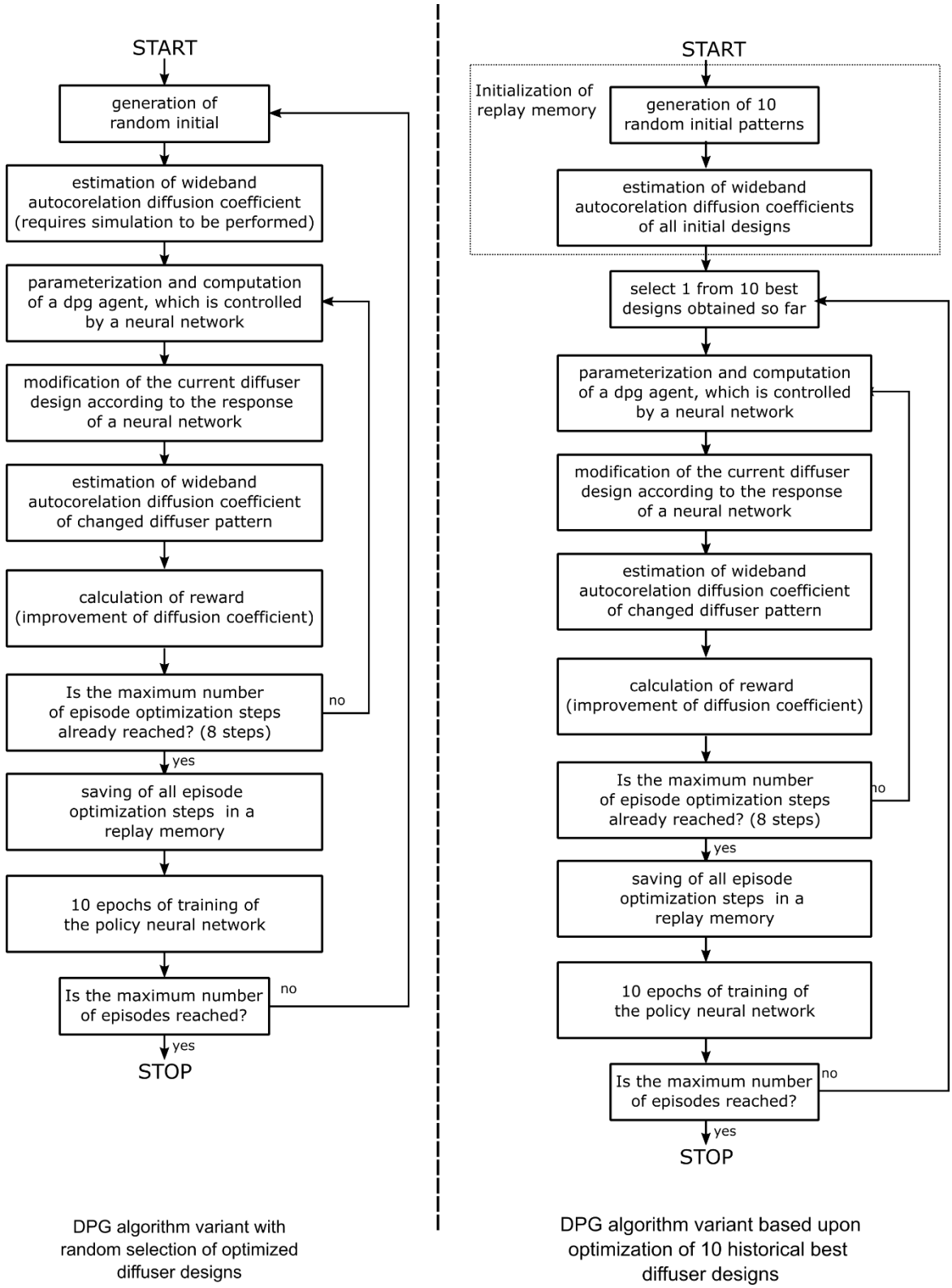


Fig. 6.5 Training procedures for reinforcement learning agents using the neural network architecture described in this Section.

Each training step was performed for 10 epochs. Adam algorithm was employed for learning rate optimization; the initial learning rate of the Adam algorithm was set to 0.001. The momentum parameter of batch normalization layers was set to 0.95. The discount factor was set to 0.99.

A reward signal calculated for each step of interaction between a particular realization of neural network architecture from Fig. 6.3 was calculated as follows:

$$\mathrm{reward}[\mathbf{n}] = \frac{\mathbf{1}}{\mathbf{2}}\left(d_{\psi,n,xy}[\mathbf{n}] - d_{\psi,n,xy}[\mathbf{n}-\mathbf{1}]\right) + \frac{\mathbf{1}}{\mathbf{2}}\left(d_{\psi,n,yz}[\mathbf{n}] - d_{\psi,n,yz}[\mathbf{n}-\mathbf{1}]\right) \tag{6.2}$$

where:

$\mathrm{reward}[\mathbf{n}]$ is a reward signal value for step *n*

$d_{\psi,n,xy}[\mathrm{n}]$, $d_{\psi,n,yz}[\mathrm{n}]$ are normal incidence autocorrelation diffusion coefficients obtained in the current step of optimization calculated after application of changes obtained from a neural network for xy, and yz planes of the diffuser, respectively,

$d_{\psi,n,xy}[\mathrm{n}-1]$, $d_{\psi,n,yz}[\mathrm{n}-1]$ are normal incidence autocorrelation diffusion coefficients obtained for a diffuser design before applying changes obtained from a neural network for xy, and yz planes of the diffuser, respectively.

Designed and simulated diffusers consisted of 100 segments and had a shape of 10 x 10 segments. Each segment could have a length within the range from 0 to 10. The maximum length of a diffuser was set to 30 cm; therefore, the height of a single element is 3 cm. To speed up computations and limit the numerical dispersion, the bandwidth of simulation was limited to 4 kHz by using a band-limited Gaussian pulse. The sampling rate of the simulation was 50.2 kHz. This is a minimum sampling rate providing sufficient spatial resolution for simulation of lengths as small as 3 cm. Spatial resolution is equal to 1.1 cm and was calculated with an assumption that the Courant number is equal to $\sqrt{1/3}$. The temporal duration of the simulation, and thus – the duration of obtained impulse responses used to calculate diffusion coefficient, is 15 ms.

In terms of propagation medium properties, the temperature of the air was assumed to be equal to 25 °C, atmospheric pressure was considered to be equal to 1000 hPa, and relative humidity of the air was assumed to be equal to 50%. Therefore, the speed of sound in the propagation medium was 347.43 m/s, and the characteristic impedance of air was equal to 403.52 rayls. As simulations were intended to replicate outcomes obtained in anechoic conditions, Berenger PML was applied to borders of a computational domain. The PML had a thickness of 30 elements, and the maximum damping factor was 25000. Admittance of rigid materials present in the computational domain was assumed to be equal to 10^{-10}. A semicircular pattern of pressure measurement points used for calculating the diffusion coefficient had a radius of 0.95 m. The source was positioned at a 2 m distance from a diffuser. The incidence angle of the acoustic wave on an acoustic diffuser was 90° (perpendicular incidence).

Dimensions of the domain (with PML and object margins) were 2.9m x 3.4 m x 2.9 m.

Graphical depictions of reward signals and achieved diffusion coefficients obtained from the policy gradient optimization process are shown in Figs. 6.6 and 6.7. The detailed results obtained by all machine learning agents are contained in Appendix C, and Appendix D. As each of the graphs is related to a different realization of a neural network (for instance, they are initialized with different sets of random weights). Therefore, each realization of the neural network is referred to as “agents” to emphasize this fact. For the algorithm using the random pattern as input for optimization, there were active 20 agents; for the algorithm performing optimization of 1 of 10 best patterns, there were 23 agents. The training lasted for approximately 15 hours. The number of episodes of training for each agent varied and was in the range between 60 up to 110. This variability was a consequence of parallel training of agents on multiple workstations equipped with different GPU cards characterized by different performance. For this experiment, 2 RTX Titan, 3 RTX 2080 Ti, and 2 RTX 2060 graphic cards were used. The aforementioned graphic cards were used to carry out both the training of neural networks and the simulation-based prediction of the diffusion coefficient of diffusers generated by algorithms.

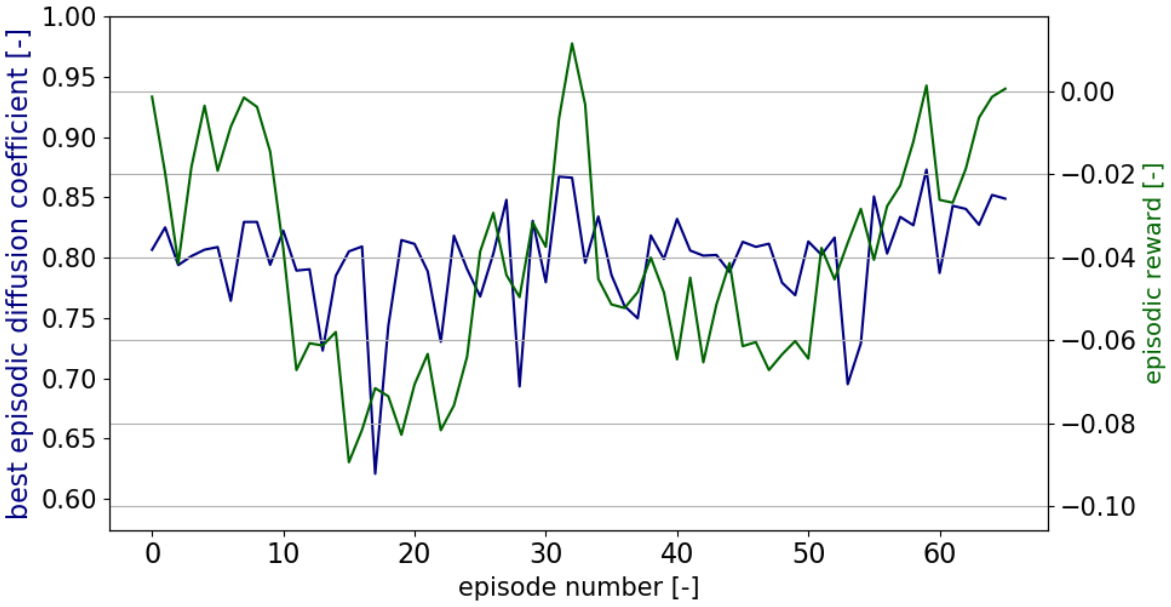


Fig. 6.6 Results obtained from agent no. 10, which was fed with random input diffuser designs to be improved over each episodes.

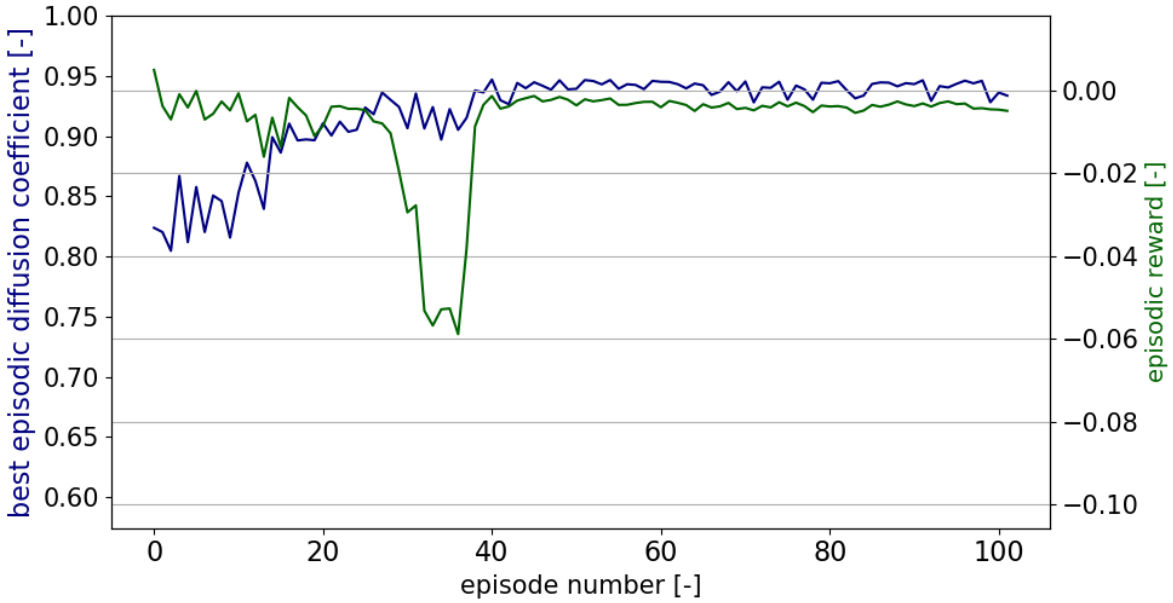


Fig. 6.7 Results obtained from agent no. 18, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

As can be seen, the best result was obtained for the approach employing improvements of 10 historical best designs. With increasing the episode number, the value of the obtained diffusion coefficient increased. Reward signal values also increased, which denotes that the neural network was learning during the training and succeeded in finding modifications that potentially can improve the design. On the other hand, it is not likely to significantly decrease the already obtained high diffusion coefficient. The random starting design did not converge, which can be seen in Fig. 6.6. With increasing the episode number, no or little improvement in terms of the obtained diffusion coefficient and reward signal can be found. Visualizations of the behavior of all reinforcement learning agents are shown in Appendices C and D.

To compare the neural network performance in terms of optimization capabilities with other heuristic optimization methods, a genetic algorithm was also employed to design diffusers with the same design specification as ones generated by the deep policy gradient algorithm. The algorithm had the same structure as in the first experiment associated with optimization of diffuser properties in a mock-up, shoebox-type room (see Section 5.3). However, a chance for swapping rows or columns was decreased to 0.2, probability of single-segment mutation was also set to 0.2. To make changes similar to ones undertaken by a neural network, a mutation change of each segment value can only be -1 or 1. This algorithm was used to carry out 3 instances of genetic algorithm design processes. Each of them employed a population of 30 diffusers. For each of those groups, 18 generations of diffusers were generated.

Histograms of diffusion associated with all designs generated by two optimization processes based on policy gradient and one process based on the genetic algorithm are presented in Fig. 6.8. To investigate if outcomes of optimization are better than a random selection of the diffuser segment length, a histogram of diffusions provided by

randomly generated diffusers also is shown. These random designs are obtained from the DPG algorithm with a random initial design. These initial, purely random designs from the beginning of all episodes can be utilized as a baseline for all designs optimized by DPG and genetic algorithm.

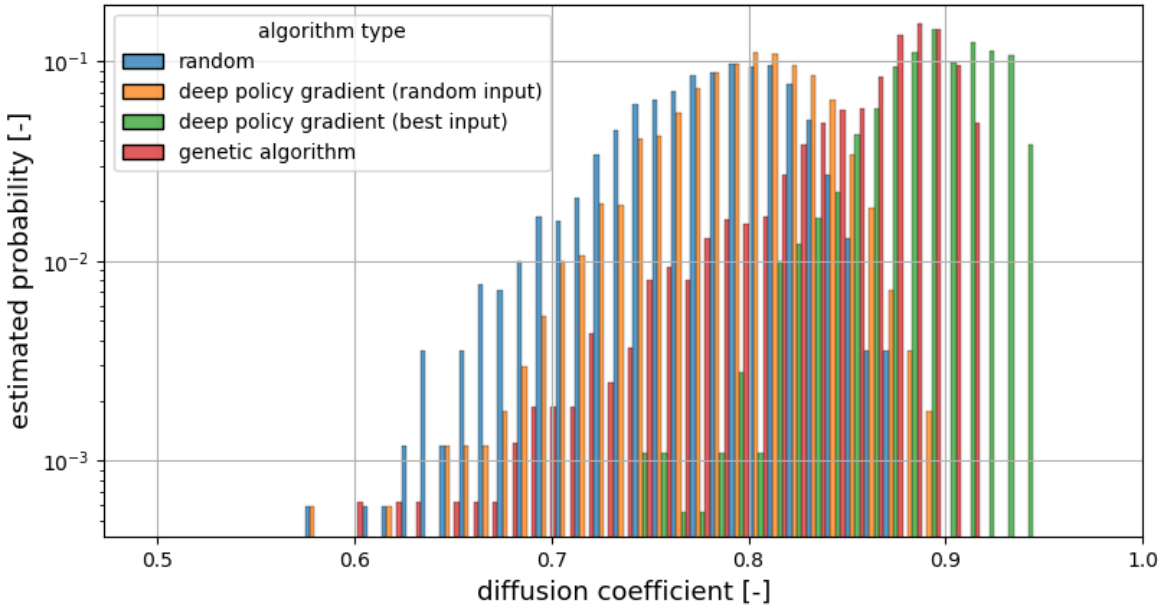


Fig. 6.8 Histograms depicting probability distributions of creating the design associated with a specified diffusion coefficient. Four algorithms are shown: DPG with randomly generated input diffusers designs, DPG with input designs picked from 10 best historic designs, genetic algorithm, and random selection of diffuser segments length.

In Fig. 6.8, it can be seen that the best designs in terms of diffusion coefficients calculated based on the FDTD simulation were ones generated by the DPG algorithm with input patterns selected from 10 best historical designs. The second best algorithm was the genetic algorithm. If the DPG method was fed with inputs obtained in a random manner, a histogram of obtained diffusion coefficients was barely different from a purely random choice of diffuser segments length.

To find out if the median differences visible in Fig. 6.8 are statistically significant, statistical tests were performed. For all tests, a significance level of 0.05 was assumed. First, a Levene test for equality of variance of all algorithm-related distributions was carried out. A test statistic was equal to 41.9, thus the *p*-value is lesser than 10^{-3}. Therefore, one must conclude that variances of estimated probability distributions visible in Fig. 6.8 are not equal. Therefore, the ANOVA test for equality of means of distributions cannot be carried out. Instead, a nonparametric alternative for ANOVA has to be used. In this case, the Kruskal-Wallis test for equality of medians can be employed. The statistic of this test for the data investigated is equal to 4370.24, and therefore *p*-value, in this case, is also lesser than 10^{-3}. This means that at least one pair of medians of distributions visible in Fig. 6.8 is not equal. To find out which one this concerns, a post-hoc test has to be carried out. In the case of the Kruskal-Wallis test, a Dunn's test can be employed. In the case of data under question, all *p*-values of Dunn's test are also lesser than 10^{-3}. Therefore, the *p*-value matrix is not shown

here, and one can conclude that all differences visible in Fig. 6.8 are statistically significant. Medians calculated for data from Fig. 6.8 are contained in Tab. 6.1.

Tab. 6.1 Medians of diffusion coefficients averaged over xy, and yz planes for designs generated by four algorithms investigated in the second experiment.

algorithm	**median of the averaged diffusion coefficient**
deep policy gradient (best of 10 input)	0.896
genetic algorithm	0.875
deep policy gradient (random input)	0.803
random pattern selection	0.786

The DPG algorithm with the best of 10 input designs was found out to be the best in terms of optimizing diffuser designs on the basis of FDTD simulation. This confirms thesis no. 2 which states, that reinforcement learning can be used to design acoustic diffusers through the means of simulation-based optimization. The DPG algorithm provided designs with a diffusion coefficient median significantly higher than any other design methods. The best designs generated by each of the algorithms are shown in Tab. 6.2.

In the course of numerical experiments, the following numbers of repetitions for each calculation scenario were carried out:

- the best design from a pool of random designs – 1692 patterns were generated; therefore, a number of designs generated by a DPG with the random initial design are also 1692,
- genetic algorithm generated a pool of 1620 diffusers designs
- deep policy gradient with input design selected from the best 10 historic designs resulted in the creation of 1816 designs.

Tab. 6.2 Patterns and diffusion coefficients of the best designs obtained from all four optimization algorithms. The diffusion coefficient was averaged for xy, and yz planes

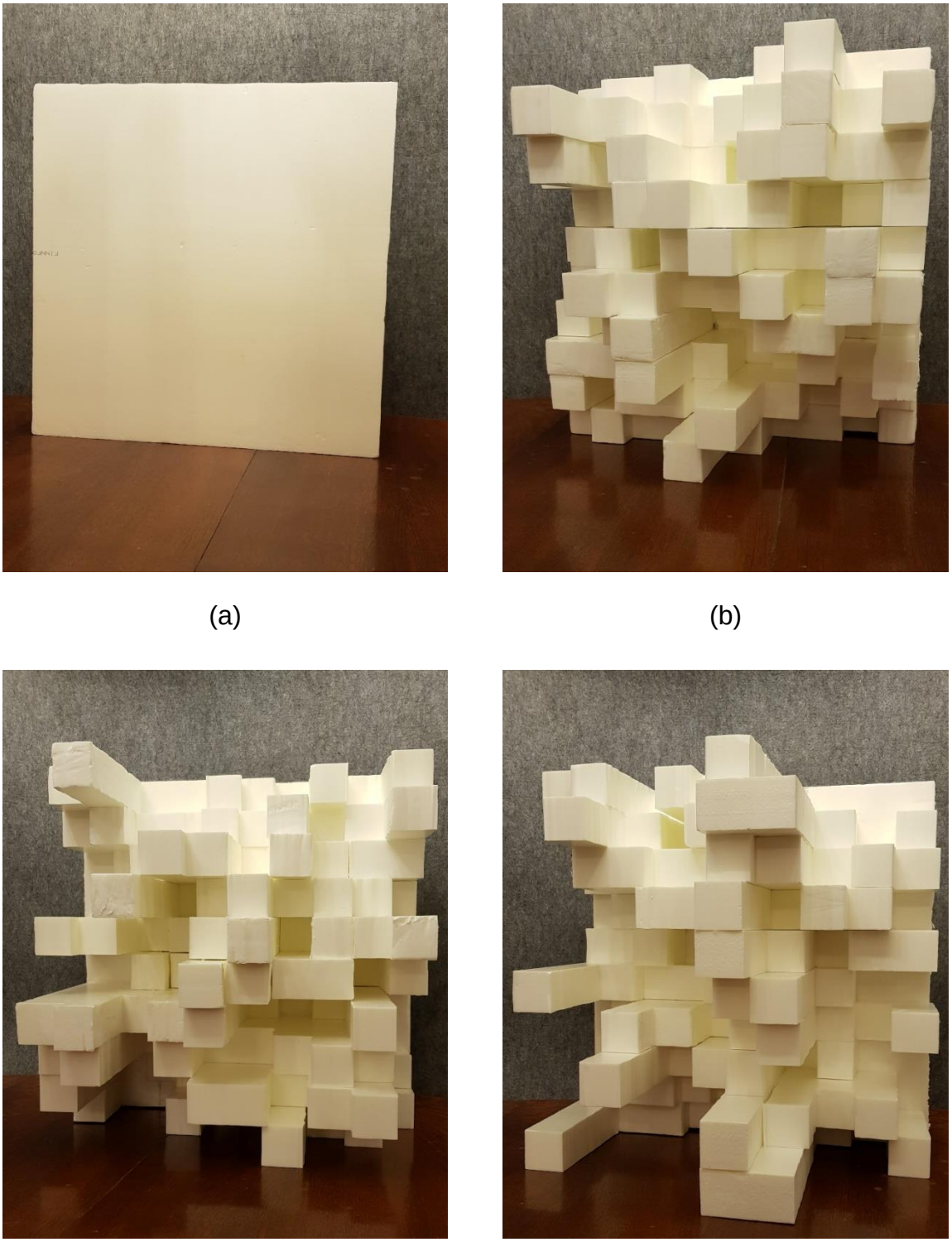
algorithm	**best-generated diffuser pattern**	**averaged diffusion coefficient**
DPG (best of 10 input designs)	5 6 0 5 3 9 0 0 0 0 0 8 4 0 6 10 10 3 0 2 1 6 6 5 5 9 8 4 5 3 0 1 2 7 7 10 4 7 5 0 1 1 3 0 3 10 6 2 4 4 8 3 2 2 3 8 9 3 0 0 0 2 7 5 2 5 8 2 1 4 0 3 9 3 5 6 0 2 0 0 1 2 2 1 0 10 7 4 5 0 7 1 1 0 1 10 10 1 0 4	0.947
genetic algorithm	5 9 1 1 2 1 0 4 1 6 3 4 2 2 3 2 6 2 3 7 0 1 4 9 8 2 7 4 4 4 0 1 10 1 6 7 5 6 5 2 7 6 2 3 8 10 1 7 7 8 0 0 4 3 10 9 7 7 1 5 9 10 6 8 9 1 0 1 9 0 7 9 2 7 4 4 6 8 4 0 7 0 1 2 8 8 5 8 9 1 3 0 0 4 2 1 8 4 5 2	0.916
DPG (random input designs)	3 7 6 7 6 0 3 0 1 5 1 2 3 7 7 0 7 9 2 5 6 3 5 6 5 1 0 9 1 4 0 9 3 8 6 1 7 2 7 3 1 4 8 1 7 5 9 8 0 9 4 2 7 6 7 2 0 7 9 0 0 1 3 7 8 6 2 9 6 3 0 7 8 1 1 4 7 2 8 4 2 4 3 5 5 0 4 5 9 1 5 1 6 6 8 8 0 5 2 0	0.896
random generation algorithm	0 1 1 3 0 0 5 8 3 3 1 9 6 4 3 5 7 9 5 8 4 8 4 8 3 1 8 9 6 6 1 2 4 8 9 6 7 2 4 6 2 5 6 1 7 9 5 5 9 5 2 2 9 8 5 6 8 3 9 8 4 2 6 9 0 0 6 6 1 0 5 5 3 9 4 0 1 4 2 6 5 0 5 5 4 9 4 1 3 1 3 4 3 3 8 4 1 1 0 1	0.877

Machine learning algorithms were able to optimize the estimated performance of diffusers. However, the practical applicability of such diffuser designs, being an outcome of numerical calculations, is limited by the error of simulation used to predict the diffusion coefficient. To find out if designs proposed in previous Sections are viable for practical purposes, three prototypes were prepared. The following designs were chosen for measurement-based evaluation:

1. design generated by the DPG algorithm (best of 10 input designs)
2. design generated by the genetic algorithm,
3. best diffuser chosen from randomly generated designs.

The outcome of the DPG algorithm with random input was omitted because the result achieved by this method was just slightly better than the ones obtained by the random generation method.

Each diffuser consisted of 100 segments arranged as a matrix of 10 x 10 elements. Each segment of the diffuser can have a length varying from 0 cm up to 30 cm. The cross-section of the element has dimensions of 5cm x 5 cm, which means that the dimensions of the whole diffusers are 50 cm by 50 cm. This size of a prototype means that a method of their manufacturing should be carefully considered. On the one hand, there are very precise methods such as 3D printing; on the other hand – such very precise methods tend to generate very small samples or require large diffusers to be printed in many parts, which then have to be assembled manually anyway, which reintroduces a chance for human-related errors [Reinhardt2016]. Therefore, in the case of prototypes that have dimensions of 50 cm x 50 cm x 30 cm, it was decided to prepare measurement prototypes out of extruded polystyrene (XPS). As the cross-section of a single segment of a diffuser is 5 cm x 5 cm, panels of 5 cm thick XPS were used to prepare prototypes. The material was cut with the hot wire method. As patterns allow segment lengths of 0, a back plate of 50 cm x 50 cm dimensions was also used to assemble prototype diffusers. Elements cut from the XPS were glued together with the back plate. Final prototypes obtained with this method are depicted in Fig. 6.9.



(a) (b)

(c) (d)

Fig. 6.9 Presentation of the reference plate and diffuser configurations obtained by various optimization methods: a) – a flat reference plate, b) – random selection, c) – genetic algorithm, d) – deep policy gradient

After manufacturing, prototypes were transported to the anechoic chamber. The measurement procedure was based upon one described in Section 3.2, which is also

corresponding to the way, the simulation was carried out in the FDTD simulation. The diffuser was positioned at the height of 1 m above the floor of the chamber and at a distance of 2 meters from the source.

Fig. 6.10 Arrangement of objects in the anechoic chamber to measure the diffusion coefficient of the flat plate. An APS Coax studio monitor used as a sound wave source and Microflown Acoustic Probe used for the measurement are also visible in the picture. Also, parts of the Cartesian robot structure are visible on the sides of the measurement area.

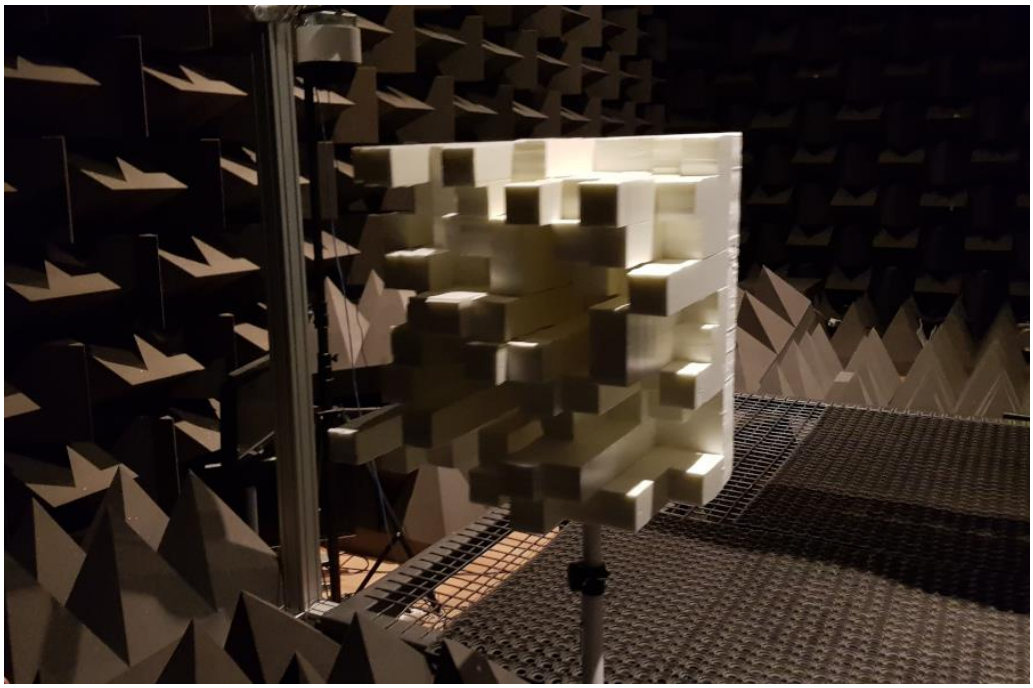


Fig. 6.11 Front side of the acoustic diffuser (designed by the genetic algorithm) positioned in in the anechoic chamber to measure the diffusion coefficient. A fragment of a Cartesian robot structure is visible on the left part of the image.

Measurement was performed to obtain both the diffusion coefficient in the xy and yz planes of scattering. Therefore for each prototype, there were two measurements performed. In total, 8 measurements were taken:

1. measurement of the room without any object positioned on the stand,
2. measurement of diffusion coefficient introduced by a flat plane,
3. measurement of design generated by selection of best randomly generated diffuser (xy plane),
4. measurement of design generated by selection of best randomly generated diffuser (yz plane),
5. measurement of design generated by genetic algorithm (xy plane),
6. measurement of design generated by genetic algorithm (yz plane),
7. measurement of design generated by DPG algorithm (xy plane),
8. measurement of design generated by DPG algorithm (yz plane).

The result of each measurement was a set of 37 impulse responses, as similarly to the simulation, a 5° spatial resolution was assumed. Although the acoustic Microflown Ultimate Sensor Probe (USP) vector probe was employed for the measurement, only the pressure channel was used for further analyses [Comesaña2015].

The measurement in an anechoic chamber has to be processed differently from the one employed for post-processing associated with the FDTD simulation. In the simulation, one can simply subtract the impulse response of the numerical computation domain from the impulse response obtained from the simulation carried out with the 3D representation of an acoustic diffuser.

In a practical setting, there is a problem associated with residual reflections present in signals obtained from measurement. Even if impulse responses are synchronized by aligning the initial excitation impulse, still there are residua reflections primarily occurring during the period after the impulse is reflected from the diffuser. Due to this fact, extraction of the impulse reflected from the diffuser was performed by trimming the signal gathered during the measurement. An example of a signal from which the room signal was subtracted is depicted in Fig. 6.12. Due to this fact, it was decided not to subtract room-related reference measurements from measurements conducted for investigated diffusers. Instead, a time window was applied to isolate only the fragment of impulse responses containing reflections originating from acoustic diffusers positioned on the stand. The most probable origin of residual reflections is the presence of a Cartesian robot used to automate the measurement of polar responses of the diffuser. On the one hand, the robot significantly speeds up measurement, as the whole process is automated, but on the other hand – one has to take into account the fact that the construction of the robot is made out of acoustically reflective materials. In some cases, reflections from the parts of the robot have to be taken into account in the process of data post-processing.

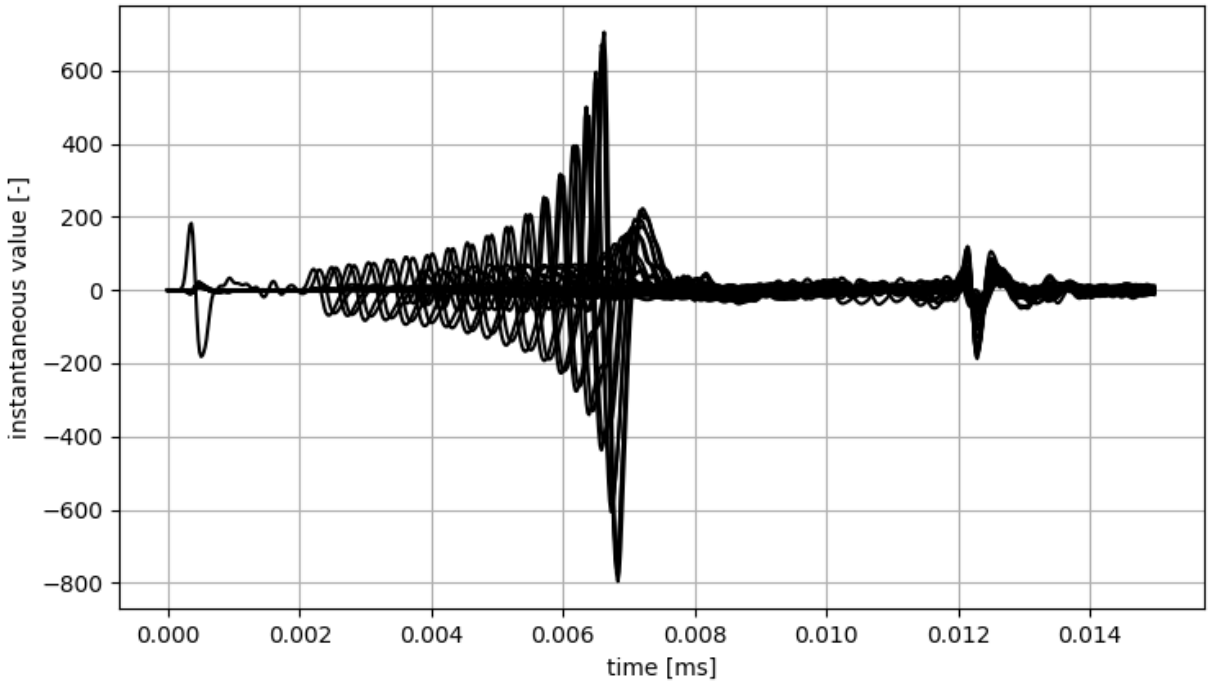


Fig. 6.12 Example of 37 overlapped impulse response fragments for the measurement of a flat plate. It can be seen that despite subtraction of a room reference measurement, still some reflections that are not originating from the plate itself can be found at the beginning (around 0 ms) and the ending (around 12 ms) of the shown signal fragment.

Polar responses of the flat reference plate and acoustic diffusers diffusion coefficients were used to calculate autocorrelation diffusion coefficients for three investigated diffuser designs in both the xy and yz planes. As the simulation was performed for a maximum frequency of 4 kHz, signals from measurement were filtered in a band-pass manner to constrict the bandwidth of signals to the range from 176.78 Hz to 5656.85 Hz. Those particular frequencies were chosen to provide a half octave margin for a 5^{th} order Butterworth band-pass filter. The lower frequency is positioned half an octave below 250 Hz, and the upper one – half an octave above 4000 Hz. Coefficients obtained in both the simulation and measurement are shown in Tab. 6.3. Errors of autocorrelation diffusion coefficient estimation are provided in Tab. 6.4.

Tab. 6.3 Autocorrelation diffusion coefficients predicted by the FDTD simulation, and the corresponding values of this number measured in an anechoic chamber.

design algorithm name	**diffusion coefficients obtained in the simulation**			**diffusion coefficients obtained in the measurement**		
	xy plane	**yz plane**	**averaged**	**xy plane**	**yz plane**	**averaged**
DPG (best of 10 input)	0.960	0.889	0.924	0.824	0.797	0.810
genetic algorithm	0.896	0.935	0.915	0.851	0.846	0.849
best of randomly generated	0.847	0.845	0.846	0.768	0.848	0.808

Tab. 6.4 Errors in autocorrelation diffusion coefficient estimation made by the FDTD simulation.

design algorithm name	**difference of diffusion coefficients obtained in the simulation (simulation – measurement)**		
	xy plane	**yz plane**	**averaged**
DPG (best of 10 input)	0.136	0.092	0.114
genetic algorithm	0.045	0.089	0.066
best of randomly generated	0.079	-0.003	0.038

A confidence interval for data from Tab. 6.4 with the assumption of confidence interval equal to 0.05 is (0.0234, 0.123). Therefore, one can conclude that the simulation has a statistically significant tendency to overestimate the diffusion coefficient, and this overestimation is likely not greater than 0.12. It is an important observation in the context of diffusion coefficient values obtained for the DPG and genetic algorithms, which turned out to be swapped in terms of the value of diffusion coefficient provided by them. It should also be noted that the worst measured coefficient was observed for one of the planes of randomly generated designs.

Still, all observed values of diffusion coefficient, which come from the measurement process, have values that allow their use in a practical setup. Moreover, the confidence interval error of the diffusion coefficient estimation allows the selection of designs that

will probably have desired diffusing properties in a real context.

This observation proves thesis no. 3, which states that designs obtained through numerical optimization are viable for use in practice and retain their sound scattering properties if designs are implemented as physical devices.

An interesting observation can also be made if the diffusion coefficient is visualized separately in octave bands. Such a type of visualization can be seen in Fig. 6.13.

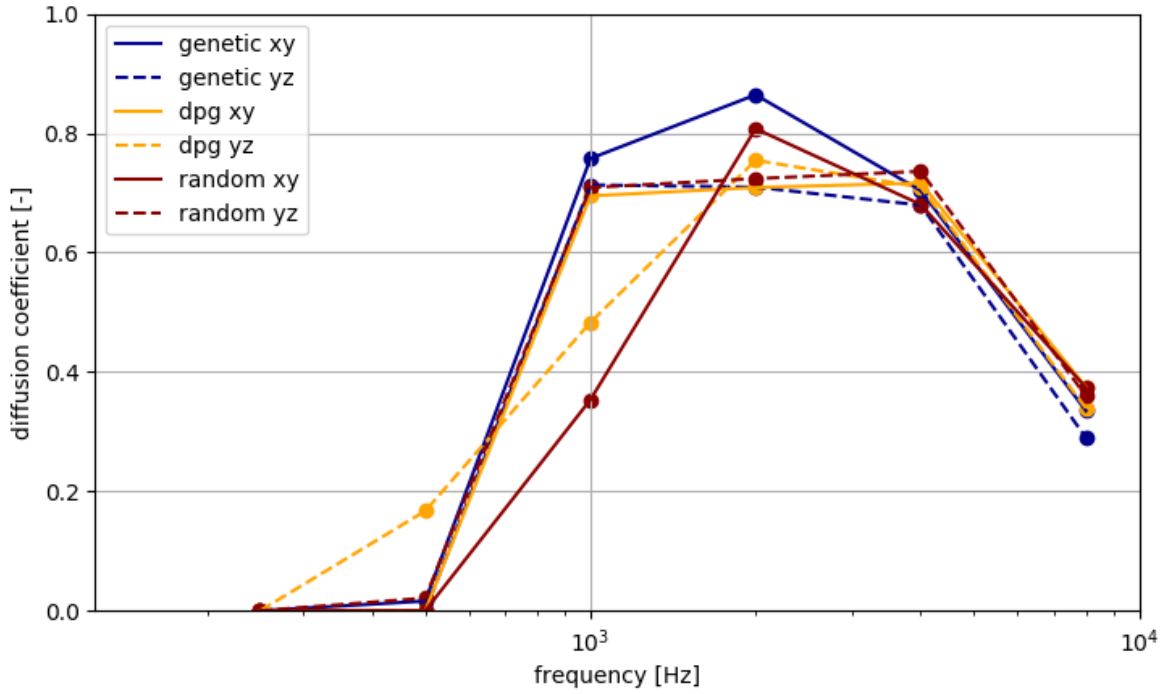


Fig. 6.13. Band-pass autocorrelation diffusion coefficient of 3 investigated acoustic diffuser designs. Diffusion coefficients are calculated for the following set of frequencies: 250 Hz, 500 Hz, 1 kHz, 2 kHz, and 4 kHz.

An interesting fact is that designs optimized by a computer are characterized by a wider usable bandwidth in which they provide diffusion. This can be a consequence of maximizing a mean broadband diffusion coefficient. Such optimization promotes maximization of diffusion in every possible frequency band. Therefore, for instance, the DPG algorithm provides uniform diffusion coefficients, which are non-negligible already from the frequency of 500 Hz.

Results presented in this Section once more refer to theses no. 1 and 2 and prove them as the metric used for the optimization of acoustic scattering coefficient of diffusers was calculated on the basis of the numerical simulation (thesis no. 1).

Outcomes of the simulation were used as a fitness function for a genetic algorithm or a reward signal for a reinforcement learning-based method, which supports claims given in thesis no 2.

Moreover, analysis of prototypes obtained from the simulation found that, in fact, all geometries designed by using an optimized computer simulation were characterized by an autocorrelation scattering coefficient which was close to the one predicted by the simulation (the error of estimation belongs to range between 0.0234, and 0.123). This means that designs obtained with methods presented in this thesis are feasible to be used in a practical context if the prediction error between (0.0234 and 0.123) is tolerable for the desired implementation.

7. CONCLUSIONS AND FUTURE DIRECTIONS

This chapter summarizes the study performed in this Ph.D. dissertation. The main findings are also contained. Moreover, possible directions of future research are proposed and discussed. The introductory Sections of this dissertation are excluded from the discussion.

The scope of this work was largely focused on employing a numerical acoustic wave propagation model and machine learning algorithms to create an autonomous program capable of designing acoustic diffusers with optimized values of given metrics. For optimization, an autocorrelation diffusion coefficient was chosen. However, any other metric which can be predicted by a numerical simulation can be selected for such kind of optimization.

The author derived a novel, modified version of the FDTD equation, which allows fast computation of fitness and reward signals using GPU. Also, a Python-based implementation of the numerical model for such a purpose was proposed. Next, the model was employed to evaluate the performance of three selected optimization algorithms, namely the genetic algorithm and two reinforcement learning algorithms – the deep Q network and the policy gradient neural network. The experiment carried out to investigate the performance of the aforementioned algorithms made it possible to support theses no.1 and no. 2 of this doctoral dissertation, which are recalled below:

1. **It is possible to employ a numerical simulation as a fitness or reward estimator used by an artificial intelligence algorithm to optimize a Schroeder diffuser design.**
2. **It is possible to employ machine learning methods such as genetic algorithms or deep reinforcement learning for optimization of the Schroeder skyline diffuser geometry to achieve a design having desired acoustic properties of autocorrelation diffusion coefficient when the measurement is performed in an anechoic condition.**

These theses were proven by the outcome of experiments with optimization of the frequency response of a simple shoebox-type room carried out using the FDTD-based fitness and reward estimators. The investigation involving both the genetic algorithms and reinforcement learning algorithms show that they were capable of optimizing acoustic properties of designed diffusers (in terms of an autocorrelation diffusion coefficient) based on feedback generated by the simulation. Also, all optimization methods achieved results that are statistically significantly better than ones employing classical approaches based on pseudo-random sequences.

The final experiment was designed with the intention of verifying if acoustic diffuser designs created with the use of numerical FDTD simulations can be used to manufacture physical prototypes and if those prototypes also will have desired features that correspond to outcomes of simulation employed by optimization processes. In this stage of the experiment, only a deep policy gradient algorithm was employed, as optimization of the design speed, which was a major issue in the first experiment, led

to a combinatorial explosion of possible action choices. This made it impossible to use the deep Q network algorithm in the second experiment. It was found out that the deep policy gradient algorithm can achieve statistically significantly better performance if optimizing the design of acoustic diffuser even if the action space is as large as 3^{100} possible actions. Results of optimizations translated to the quality of physical prototypes were assessed by measurement in an anechoic chamber. It occurred that for selected bands, a randomly-generated acoustic diffuser performed better than designs generated by optimization algorithms. Moreover, it was found that especially the design obtained from the deep policy gradient method provided a more stable diffusion coefficient across the bandwidth in which the diffuser has a measurable effect on the acoustic field. Also, all three evaluated acoustic diffuser prototypes had autocorrelation diffusion coefficient, which is reasonably closer to the one predicted by the simulation, which proves thesis no. 3, namely:

3. **Prepared geometries of acoustic scattering elements acquired in the computer-based optimization are feasible for practical implementation and can be used as the means of acoustic room treatment.**

The following **major original contributions** were introduced in this dissertation:

- derivation of the modified homogenous FDTD difference equation based on works by Webb and Bilbao [Webb2011], which allows the use of Berenger PML in such kind of simulations,
- derivation of an FDTD model which is capable of handling PML layers and take into account such properties of the propagation medium as air temperature, atmospheric pressure, and relative humidity,
- derivation of machine learning models employing FDTD simulation as an environment for a reinforcement learning agent which are capable of designing acoustic diffusers with maximized values of the acoustic diffusion coefficient
- a Python-based implementation of the proposed FDTD simulation method,
- the use of the FDTD-based simulation as a fitness function estimator of a genetic algorithm that performs maximization of desired properties of an acoustic diffuser
- evaluation of acoustic diffusers prototypes created by machine learning algorithms (namely – deep duelling Q-networks, DDQNs, and deep policy gradient, DPG) in real-life conditions with the use of physical prototypes in which acoustic properties were measured in an anechoic chamber.

The presented achievements may provide new insight into the artificial intelligence-driven design of passive acoustic treatment devices. The use of numerical simulations allows optimizing especially desired properties of designed elements such as acoustic diffusers. In experiments presented in this dissertation, the main focus was on the maximization of the autocorrelation diffusion coefficient. However, the optimization process does not have to be limited just to one single parameter. An interesting future direction of research may be the optimization of multiple parameters and optimization of the behavior of a diffuser or other acoustic treatment device which is tailored to the

geometry of a room. This would require to additionally take into account the model of the space in which the acoustic treatment device would be implemented, but potentially such an approach can help to exploit some properties of treated spaces and use them as an advantage in the process of improving the acoustics of a given space under treatment.

Finally, an interesting topic for future research is to modify the proposed methods to apply them in the design of other types of diffusers.

REFERENCES

1. [Abadi2015] Abadi, M., et. al., TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
2. [Adamczyk2011] Adamczyk, J., Targosz, J., The concept of limitation of the vibration generated by rail-vehicles at railway stations and railway crossings, Arch. Transp., 23 (1), 5–22, 2011, doi: 10.2478/v10174-011-0001-1.
3. [Alles2011] Alles, E.; van Dongen, K., W., A., Perfectly matched layers for frequency-domain integral equation acoustic scattering problems, IEEE Transactions on ultrasonics, ferroelectrics, and frequency control 58, 1077-1086, 2011, doi:10.1109/TUFFC.2011.1908.
4. [Arik2017] Arik, S., Ö.; Chrzanowski, M.; Coates, Adam; Diamos, Gregory; Gibiansky, Andrew; Kang, Yongguo; Li, Xian; Miller, John; Ng., Andrew; Raiman, Jonathan; Sengupta, Shubho; Shoeybi, Mohammad, Deep voice: real-time neural text-to-speech, 34th International Conference on Machine Learning - Volume 70 (ICML'17), Sydney, Australia, 6-11.8.2017.
5. [Arvidsson 2020] Arvidsson, E.; Nilsson, E.; Hagberg, D.B.; Karlsson, O.J.I., The Effect on Room Acoustical Parameters Using a Combination of Absorbers and Diffusers—An Experimental Study in a Classroom. Acoustics 2020, 2, 505-523. doi: 10.3390/acoustics2030027.
6. [Astolfi2008] Astolfi, A., Pellerey, F., Subjective and objective assessment of acoustical and overall environmental quality in secondary school classrooms, J. Acoust. Soc. Am., 123 (1), 163–173, 2008, doi: 10.1121/1.2816563.
7. [Avis2004] Avis, M., R.; Xiao, L.; Cox, T., J., Practical Measurements on Active Diffusers, The 18th International Congress on Acoustics ICA 2004, Kyoto, Japan, 4-9.4.2004.
8. [Avis2005] Avis, M. R.; Xiao, L.; Cox, T. J., Stability and Sensitivity Analyses for Diffusers with Single and Multiple Active Elements, JAES 53 (11), 1047-1060, 2005.
9. [Baker2016] Baker, B.; Gupta, O.; Naik, N.; Raskar, R., Designing neural network architectures using reinforcement learning, ArXiv preprint no. 1611.02167, 2016, url: https://arxiv.org/abs/1611.02167
10. [Baulac2008] Baulac M.; Defrance J.; Jean P., Optimisation with genetic algorithm of the acoustic performance of T-shaped noise barriers with a reactive top Surface, Applied Acoustics 69, 332-342, 2009.
11. [Becache2005] Bécache, E.; Chaigne, Antoine; Derveaux, Gregoire; Joly, Patrick, Numerical simulation of a guitar, Computers & Structures 83 (2-3), 107-126, 2005, doi:10.1016/j.compstruc.2004.04.018.
12. [Beranek1996] Beranek, L.L., Concert and Opera Halls: How they Sound, Acoust. Soc. Amer., Woodbury, NY, 1996.
13. [Berardi2020] Berardi, U. Iannace, G., The acoustic of Roman theatres in Southern

Italy and some reflections for their modern uses, Appl. Acoust., 170, 2020, doi: 10.1016/j.apacoust.2020.107530.

14. [Berenger1994] Berenger, J.-P., A perfectly matched layer for the absorption of electromagnetic waves, Journal of Computational Physics Volume 114 (2), 185-200, 1994, doi:doi.org/10.1006/jcph.1994.1159.
15. [Bergman2018] Bergman, David, R., Computational Acoustics: Theory and Implementation, John Wiley and Sons Ltd, 2018, doi:10.1002/9781119277323.
16. [Bermudez2007] Bermúdez, A.; Hervella-Nieto, L.; Prieto, A.; Rodrıguez, R., An optimal perfectly matched layer with unbounded absorbing function for time-harmonic acoustic scattering problems, Journal of Computational Physics 223 (2), 469-488, 2007, doi:10.1016/j.jcp.2006.09.018.
17. [Bianco2019] Bianco, M., Gerstoft, P., Traer, J., Ozanich, E., Roch, M., Gannot, S., Deledalle C., Machine learning in acoustics: Theory and applications, J Acoust Soc Am. 2019 146(5) 3590. doi: 10.1121/1.5133944.
18. [Blaauw2017] Blaauw, M.; Bonada, J.., A Neural Parametric Singing Synthesizer Modeling Timbre and Expression from Natural Songs, Applied Sciences 7, 2017, doi:10.3390/app7121313.
19. [Blackstock2000] Blackstock, D., T., Fundamentals of pshysical acoustics, John Wiley & Sons, 2000.
20. [Blauert2009] Blauert, J.; Xiang, N., Acoustics for Engineers: Troy Lectures, Second Edition, Springer-Verlag Berlin Heidelberg, 2009, doi:10.1007/978-3-642-03393-3.
21. [Bongard2011] Bongard, F.; Lissek, H.; Mosig, J., R., Transmission line based metamaterials for acoustic waves, XXXth URSI General Assembly and Scientific Symposium, Istanbul, Turkey, 1-4.8.2011, doi:10.1109/URSIGASS.2011.6050389.
22. [Botts2014] J. Botts and L. Savioja, “Spectral and pseudospectral properties of finite difference models used in audio and room acoustics,” IEEE Trans. Audio, Speech Lang. Process. 22 (9), 1403–1412, 2014, doi: 10.1109/TASLP.2014.2332045.
23. [Boulanger-Lewandowski2013] Boulanger-Lewandowski, N.; Bengio, Y.; Vincent, P., AUDIO CHORD RECOGNITION WITH RECURRENT NEURAL NETWORKS, 14th International Society for Music Information Retrieval Conference Curitiba, Curitiba, Brazil, 4-8.11.2013.
24. [Brennan2013] Brennan, B., Kirby, R., C.; Zweck,J.; Minkoff, S., High-Performance Python-based Simulations of Pressure and Temperature Waves in a Trace Gas Sensor, PyHPC 2013, Denver, USA, 18.11.2013.
25. [Brick2008] Brick, H.; Ochmann, Martin, A half ‐ space BEM for the simulation of sound propagation above an impedance plane, The Journal of the Acoustical Society of America 123 (5), 3418, 2008, doi:10.1121/1.2934160.
26. [Brown2015] Brown A. D.; Stecker, G. C.; Tollin, D. J., The precedence effect in sound localization, J Assoc Res Otolaryngol. 16 (1), 46753, 2015,

doi:10.1007/s10162-014-0496-2.

27. [Buduma2017] Buduma, N.; Locasio, N., Fundamentals of Deep Learning. Designing next-generation machine intelligence algorithms, O'Reilly Media, Inc., 2017.

28. [Cao2020] Cao, X., Cai. Y., Cui, X., A parallel numerical acoustic simulation on a GPU using an edge-based smoothed finite element method, Adv. Eng. Softw., 148, 2020, doi: 10.1016/j.advengsoft.2020.102835.

29. [Carvalho1996] Carvalho, A., Morgado, A., Henrique, L., Relationships between subjective and objective acoustical measures in churches, J. Acoust. Soc. Am., 100 (4), 2707–2707, 1996, doi: 10.1121/1.416099.

30. [Comesaña2015] Comesaña, D., Steltenpool, S., Korbasiewicz, M., Emiel, T., Direct acoustic vector field mapping: new scanning tools for measuring 3D sound intensity in 3D space, conference paper presented at EuroNoise2015, Maastricht, 31.05-03.06.2015.

31. [Chern2019] Chern, A., A reflectionless discrete perfectly matched layer, Journal of Computational Physics 381, 91-109, 2019, doi:10.1016/j.jcp.2018.12.026.

32. [Choi2017] Choi, J.; Lee, B.-J.; Zhang, B.-T., Multi-focus Attention Network for Efficient Deep Reinforcement Learning, Workshops at the thirty-first AAAI Conference on Artificial Intelligence (AAAI-17), San Francisco, USA, 4-5.2.2017.

33. [Chollet2015] Chollet F, others. Keras [Internet]. GitHub; 2015. Available from: https://github.com/fchollet/keras

34. [Citarella2011] Citarella, R.; Landi, M., Acoustic analysis of an exhaust manifold by indirect Boundary Element Method, The Open Mechanical Engineering Journal 14, 138-151, 2011, doi:10.2174/1874155X01105010138.

35. [Collet2009] Collet, M.; David, P.; Berthillier, M., Active acoustical impedance using distributed electrodynamical transducers, J Acoust Soc Am. 125 (2), 882-894, 2009, doi:10.1121/1.3026329.

36. [Collino1998] Collino, F.; Monk, P. B., Optimizing the perfectly matched layer, Computer Methods in Applied Mechanics and Engineering 164 (1-2), 157-171, 1998, doi:10.1016/S0045-7825(98)00052-8.

37. [Cox2006] Cox, T., J.; Avis, M., R.; Xiao, L., Maximum length sequence and Bessel diffusers using active technologies, Journal of Sound and Vibration 289 (4-5), 807-829, 2006, doi:10.1016/j.jsv.2005.02.019.

38. [Cox2017] Cox, T., J.; D'Antonio, P., Acoustic Absorbers and Diffusers: Theory, Design and Application, CRC Press, 2017.

39. [Cucharero2019] Cucharero, M., J.; Hänninen, T.; Lokki, T.. , Influence of Sound-Absorbing Material Placement on Room Acoustical Parameters, Acoustics 1, 644-660, 2019, doi:10.3390/acoustics1030038.

40. [Davies1980] Davies, D.; Davies, C., The LEDE concept for the control of acoustic and psychoacoustic parameters in recording control rooms, J. Audio Eng. Soc. 28 (3), 585-595, 1980.

41. [Derveaux2003] Derveaux, G.; Chaigne, A.; Joly, Pa.; Bécache, E., Time-domain simulation of a guitar: Model and method, The Journal of the Acoustical Society of America 114 (6), 2003, doi:10.1121/1.1629302.

42. [Desai2009] Desai, S.; Raghavendra, E.; Yegnanarayana, B.; Black, A.; Prahallad, Kishore, Voice conversion using Artificial Neural Networks, IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2009, Taipei, Taiwan, 19-24.4.2009, doi:10.1109/ICASSP.2009.4960478.

43. [Donahue2018] Donahue, C.; McAuley, J.; Puckette, Miller, Adversarial audio synthesis, ArXiv preprint no. 1802.04208, 2018, url: https://arxiv.org/abs/1802.04208

44. [Drake2014] Drake, J.; Likhachev, M.; Safonova, A., Prioritized Computation for Numerical Sound Propagation, 17th International Conference on Digital Audio Effects (DAFx), Erlangen, Germany, 1-5.9.2014.

45. [Engel2017] Engel, J.; Resnick, C.; Roberts, A.; Dieleman, S.; Norouzi, M.; Eck, D.; Simonyan, K., Neural audio synthesis of musical notes with WaveNet autoencoders, 34th International Conference on Machine Learning - Volume 70 (ICML'17), Sydney, Australia, 6-11.8.2017.

46. [Everest2015] Everest, A.; Pohlmann, K., Master handbook of acoustics, McGraw Hill Education, 2015.

47. [Fahy1995] Fahy, F., J., Sound Intensity. Second Edition, CRC Press, 1995.

48. [Farlow1993] Farlow, S., J., Partial Differential Equations for Scientists and Engineers, Dover Publications, Inc., 1993.

49. [Fastl2007] Fastl, H., Zwicker, E., Psychoacoustics: Facts and models, Psychoacoustics Facts Model.), 1–463, 2007, doi: 10.1007/978-3-540-68888-4.

50. [Fernandez-Blanco2020] Fernandez-Blanco, E., Rivero, D., Pazos, A., Convolutional neural networks for sleep stage scoring on a two-channel EEG signal, Soft Comput. 24 (6), 4067–4079, 2020, doi: 10.1007/s00500-019-04174-1.

51. [Feynman1965] R. P. Feynman, R. B. Leighton, M. Sands, and E. M. Hafner, "The Feynman Lectures on Physics; Vol. I," *Am. J. Phys.* 33 (9), 750–752, 1965, doi: 10.1119/1.1972241.

52. [Fiala2010] Fiala, P.; Huijssen, J.; Pluymers, B.; Hallez, R.; Desmet, W., Fast multipole BEM modeling of head releated transfer functions of a dummy head and torso, ISMA2010 International Conference on Noise and Vibration Engineering; 2010, Leuven, Belgium, 20-23.9.2010.

53. [Foerster2017] Foerster, J.; Nardelli, N.; Farquhar, G.; Afouras, T.; Torr, P., H.; Kohli, P.; Whiteson, S., Stabilising experience replay for deep multi-agent reinforcement learning, 34th International Conference on Machine Learning, Sydney, Australia, 6-

11.8.2017.

54. [François-Lavet2018] François-Lavet, V., Henderson, P., Islam, R., Bellemare, M.G., & Pineau, J., An Introduction to Deep Reinforcement Learning, Foundations and Trends in Machine Learning 11, 219-354, 2018.

55. [Gao2020] Gao, Y., Gao, B., Chen, Q., Liu, J., Zhang, Y., Deep convolutional neural network-based epileptic electroencephalogram (EEG) signal classification, Front. Neurol. 11, 2020, doi: 10.3389/fneur.2020.00375.

56. [Gerard2012] Gérard, F.; Tournour, M.; el Masri, N.; Cremers, L.; Felice, M.; Selmane, A. , Acoustic Transfer Vectors for Numerical Modeling of Engine Noise, Sound Vib. Mag. 36, 20-25, 2012.

57. [Gervals2011] Gervals, R., Home Recording Studio. Build it like the pros. Second edition, Cengage Learning, 2011.

58. [Giménez2014] Giménez, A., Cibrián, R., Cerdá, S., Girón, S., Zamarreño, T., Mismatches between objective parameters and measured perception assessment in room acoustics: A holistic approach, Build. Environ., 74, 119–131, 2014, doi: 10.1016/j.buildenv.2013.12.022.

59. [Gołaś2009] Gołaś, A., Suder-Dębska, K., Analysis of Dome Home Hall theatre acoustic field, Arch. Acoust. 34, (3) 273–293, 2009.

60. [Gołaś2012] Gołaś, A., Suder-Dębska, K., Multi-channel system for sound creation in open areas, Arch. Acoust., 37 (3), 323–329, 2012, doi: 10.2478/v10168-012-0041-4.

61. [Grochowina2020] Grochowina, M., Leniowska, L., Gala-Błądzińska, A., The prototype device for non-invasive diagnosis of arteriovenous fistula condition using machine learning methods, Sci. Rep., 10 (1), 2020, doi: 10.1038/s41598-020-72336-5.

62. [Grondman2012] Grondman, I.; Busoniu, L.; Lopes, G., A., D.; Babuska, R., A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42 (6), 1291-1307, 2012, doi: 10.1109/TSMCC.2012.2218595.

63. [Guidorzi2015] Guidorzi, P., Barbaresi, L., D'Orazio, D., Garai, M., Impulse responses measured with MLS or Swept-Sine signals applied to architectural acoustics: An in-depth analysis of the two methods and some case studies of measurements inside theaters, Energy Procedia 78, 1611–1616, 2015, doi: 10.1016/j.egypro.2015.11.236.

64. [Hamilton2017] Hamilton, B., Bilbao, S., FDTD Methods for 3-D Room Acoustics Simulation With High-Order Accuracy in Space and Time, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 25 (11), 2112-2124, 2017, doi: 10.1109/TASLP.2017.2744799.

65. [Hamming1986] Hamming, R., W, Numerical Methods for Scientists and Engineers, Dover Publications, Inc., 1986.

66. [Hansen2018] Hansen, C., H., Foundations of vibroacoustics, CRC Press, 2018.

67. [Hemanth2020] Hemanth, D., EEG signal based Modified Kohonen Neural Networks for Classification of Human Mental Emotions, J. Artif. Intell. Syst. 2 (1), 1–13, 2020, doi: 10.33969/ais.2020.21001.

68. [Hinton2012] Hinton, G.; Deng, li; Yu, D.; Dahl, G.; Mohamed, Abdel-Rahman; J., Navdeep; Sr., A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.; Kingsbury, B., Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups, IEEE Signal Processing Magazine 29, 82-97, 2012, doi:10.1109/MSP.2012.2205597.

69. [Hsiao2011a] Hsiao, G.; Liu, F.; Sun, J.; Xu, L.., A coupled BEM and FEM for the interior transmission problem in acoustics, J. Computational Applied Mathematics 235, 5213-5211, 2011, doi:10.1016/j.cam.2011.05.011.

70. [Hsiao2011b] Hsiao, G., C.; Xu, L., A system of boundary integral equations for the transmission problem in acoustics, Applied Numerical Mathematics 61 (9), 1017-1029, 2011, doi:10.1016/j.apnum.2011.05.003.

71. [Huang2015] Huang, P.; Kim, M.; Hasegawa-Johnson, M.; Smaragdis, P., Joint Optimization of Masks and Deep Recurrent Neural Networks for Monaural Source Separation, IEEE/ACM Transactions on Audio, Speech, and Language Processing 23 (12), 2136-2147, 2015, doi:10.1109/TASLP.2015.2468583.

72. [Huang2020] Huang, Z., Liu, C., Fei, H., Li, W., Yu, J., Cao, Y., Urban sound classification based on 2-order dense convolutional network using dual features, Appl. Acoust. 164, 2020, doi: 10.1016/j.apacoust.2020.107243.

73. [Ibrahim2020] Ibrahim, I. Silva, R., Mohammadi, M., Ghorbanian, V., Lowther, D., Surrogate-Based Acoustic Noise Prediction of Electric Motors, IEEE Transactions on Magnetics, 56 (2), 1-4, 2020 doi: 10.1109/TMAG.2019.2945407.

74. [Inan2011] Inan, U., S.; Marshall, R., A., Numerical Electromagnetics: The FDTD Method, Cambridge University Press, 2011.

75. [ITURBS1116] ITU-R BS.1116, ITU-R recommendation, assessment of audible differences of sound systems using multiple stimuli without a given reference, document available online: https://www.itu.int/rec/R-REC-BS.1116/en, access date: 10.02.2021.

76. [ITURBS1284] ITU-R BS.1284, ITU-R recommendation, General methods for the subjective assessment of sound quality, document available online: https://www.itu.int/rec/R-REC-BS.1284/en, access date: 10.02.2021.

77. [ITURBS1534] ITU-R BS.1534, ITU-R recommendation, subjective assessment of intermediate quality level of audio systems, document available online: https://www.itu.int/rec/R-REC-BS.1534/en, access date: 10.02.2021.

78. [ITURBS2132] ITU-R BS.2132, ITU-R recommendation, Assessment of audible differences of sound systems using multiple stimuli without a given reference), document available online: https://www.itu.int/rec/R-REC-BS.2132/en, access date: 10.02.2021.

79. [Jaques2016] Jaques, N.; Gu, S.; Turner, R., E.; Eck, D., Generating Music by Fine-Tuning Recurrent Neural Networks with Reinforcement Learning, Deep Reinforcement Learning Workshop, NIPS (2016), Barcelona, Spain, 9.12.2016.

80. [Jee2000] Jee, S.-H.; Yi, J.-C., The Application of the Simulation Techniques to Reduce the Noise and Vibration in Vehicle Development, FISITA World Automotive Congress F2000H240, Seoul, South Korea, 12-15.6.2000.

81. [Jimenez2017] Jiménez, N.; Cox, T.; Romero-García, V.; Groby, J.-P., Metadiffusers: Deep-subwavelength sound diffusers, Scientific Reports. 7, 2017, doi:10.1038/s41598-017-05710-5.

82. [Kahana2007] Kahana, Y.; Nelson, P., A., Boundary element simulations of the transfer function of human heads and baffled pinnae using accurate geometric models, Journal of Sound and Vibration 300 (3-5), 552-579, 2007, doi:10.1016/j.jsv.2006.06.079.

83. [Karkar2014] Karkar, S.; Rivet, E.; Lissek, H.; Strobino, Da.; Pittet, A., et al.. , Electroacoustic absorbers for the low-frequency modal equalization of a room: what is the optimal target impedance for maximum modal damping, Forum Acusticum 2014, Kraków, Poland, 7-12.9.2014.

84. [Katz2007] Katz, B., Mastering Audio: the art. And the science, Focal Press, 2007.

85. [Kempka2016] Kempka, M.; Wydmuch, M.; Runc, G.; Toczek, J.; Jaśkowski, W. , Vizdoom: A doom-based ai research platform for visual reinforcement learning, IEEE Conference on Computational Intelligence and Games (CIG), Santorini, Greece, 20-23.8.2016, doi:10.1109/CIG.2016.7860433.

86. [Kereliuk2015] Kereliuk, C.; Sturm, B., L.; Larsen, J., Deep Learning and Music Adversaries, IEEE Transactions on Multimedia 17 (11), 2059-2071, 2015, doi:10.1109/TMM.2015.2478068.

87. [Kim2010] Kim, Y.-H., Sound Propagation: An Impedance Based Approach, John Wiley & Sons (Asia) Pte Ltd, 2010, doi:10.1002/9780470825853.

88. [Kim2013] Kim, Y.; Lee, H.; Provost Mower, E., , Deep learning for robust feature generation in audiovisual emotion recognition, IEEE international conference on acoustics, speech and signal processing, Vancouver, Canada, 26-31.5.2013, doi:10.1109/ICASSP.2013.6638346.

89. [Kim2019] Kim, D., A., Modified PML Acoustic Wave Equation, Symmetry 11 (2), 177, 2019, doi:10.3390/sym11020177.

90. [Kirkup1998] Kirkup, S., The Boundary Element Method in Acoustics: A Development in Fortran, Integrated Sound Software, 1998.

91. [Kobayashi2004] Kobayashi, Y.; Aiyoshi, E., Automatic core design using reinforcement learning, 2004 American Control Conference, Boston, USA, 30-2.6-7.2004, doi:10.23919/ACC.2004.1384779.

92. [Kotus2015] Kotus J., Kostek B., Measurements and Visualization of Sound Intensity Around the Human Head in Free Field Using Acoustic Vector Sensor; J. Audio Eng. Soc., No. 1/2, 63,. 99 - 109, 2015, doi: 10.17743/jaes.2015.0009.].

93. [Kowalczyk2009] Kowalczyk K.; van Walstijn M., Room acoustics simulation using 3-D compact explicit FDTD schemes, IEEE Transactions on Audio, Speech and Language Processing 1 (1), 2009.

94. [Kurowski2016] Kurowski, A., Kotus, J., Kostek, B., Czyżewski, A., Numerical modeling of sound intensity distributions around acoustic transducer. Abstract-precis Reviewed Convention Paper 146, 1-10, 2016.

95. [Kurowski2017] Kurowski A., Kotus J., Kostek B., Measurement and visualization of sound intensity vector distribution in proximity of acoustic diffusers, 142nd Audio Engineering Society International Convention 2017, AES 2017, Berlin: , 2017, s.1-6.

96. [Kurowski2018] Kurowski A., Koszewski D., Kotus J., Kostek B., A Stand for Measurement and Prediction of Scattering Properties of Diffusers, 144nd Audio Engineering Society International Convention 2018, AES 2018, Mediolan: , 2018, s.1-4.

97. [Kurowski2019a] Kurowski, A., Mrozik, K., Kostek, B., Czyżewski, A., Method for Clustering of Brain Activity Data Derived from EEG Signals. Fundamenta Informaticae, 168, 249-268, 2019, doi: https://doi.org/10.3233/fi-2019-1831.

98. [Kurowski2019b] Kurowski, A., Mrozik, K., Kostek, B., Czyżewski, A., Comparison of the effectiveness of automatic EEG signal class separation algorithms, J. Intell. Fuzzy Syst., 37 (6), 7537–7543, 2019, doi: 10.3233/JIFS-179360.

99. [Kurowski2019c] Kurowski A., Zaporowski S., Czyżewski A., Automatic labeling of traffic sound recordings using autoencoder-derived features, 2019 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Poznan, Poland, 2019, 38-43, doi: 10.23919/SPA.2019.8936709.

100. [Kurowski2020] Kurowski, A., Zaporowski, S., Czyzewski, A., 1D convolutional context-aware architectures for acoustic sensing and recognition of passing vehicle type, 2020 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Poznan, Poland, 2020, 142-145, doi: 10.23919/SPA50552.2020.9241256.

101. [Kutruff2009] Kutruff, H., Room Acoustics, Taylor & Francis, 2009.

102. [Langtangen2016] Langtangen, H., P.; Logg, A., Solving PDEs in Python: The FEniCS Tutorial I, Springer International Publishing, 2016, doi:10.1007/978-3-319-52462-7.

103. [Le2012] Le, Q., V.; Ranzato, M.'A.; Monga, R.; Devin, M.; Chen, K.; Corrado, G., S.; Dean, Jef.; Ng, A., Y., Building High-level Features Using Large Scale Unsupervised Learning, 29th International Conference on Machine Learning, Edinburgh, United Kingdom, 26-1.6-7.2012.

104. [Leniowska2012] Leniowska, L., Leniowski, R., The joint vibration analysis of a multi-link surgical manipulator, Arch. Acoust., 37 (4), 475–482, 2012, doi: 10.2478/v10168-012-0059-7.

105. [Lissek2009] Lissek, H.; Boulandet, R.; René, P.-J., Shunt loudspeakers for modal control in rooms, 16th International Congress on Sound and Vibration, Kraków, Poland, 5-9.7.2009.

106. [Lissek2013] Lissek, H., Electroacoustic metamaterials: achieving negative acoustic properties with shunt loudspeakers, 21st International Congress on Acoustics, Montreal, Canada, 2-7.6.2013.

107. [Li2016] D. Li, D. I. W. Levin, W. Matusik, and C. Zheng, "Acoustic voxels: Computational optimization of modular acoustic filters," ACM Trans. Graph. 35 (4), 2016, doi: 10.1145/2897824.2925960.

108. [Liu1997] Liu, Q.-H.; Tao, J.., The perfectly matched layer for acoustic waves in absorptive media, Journal of The Acoustical Society of America 102, 2072-2082, 1997, doi:10.1121/1.419657.

109. [Logg2010] Logg, A.; Wells, G., N., DOLFIN: Automated finite element computing, ACM Trans. Math. Softw. 37 (2), 2010, doi:10.1145/1731022.1731030.

110. [Long2014] M. Long, *Architectural Acoustics: Second Edition*, Academic Press, 2014.

111. [Luo2020] Luo Y., et al., EEG-Based Emotion Classification Using Spiking Neural Networks, IEEE Access 8, 46007–46016, 2020, doi: 10.1109/ACCESS.2020.2978163.

112. [Makarewicz2004] Makarewicz, R., Dźwięki i fale, Wydawnictwo Naukowe UAM, 2004. (in polish)

113. [Markall2012] Markall , G., R.; Slemmer, A.; Ham, D., A.; Kelly, P., H., J., Cantwell, C., D.; Sherwin, S., J., Finite element assembly strategies on multi‐core and many‐core architectures, International Journal for Numerical Methods in Fluids 71 (1), 2012, doi:10.1002/fld.3648.

114. [Martinez2017] Martinez Ramirez, M.; Reiss, J., Deep Learning and Intelligent Audio Mixing, 3rd Workshop on Intelligent Music Production, Salford, United Kingdom, 15.9.2017.

115. [Matten2017] Matten, G.; Ouisse, M.; Collet, M.; Karkar, S.; Lissek, H., et al.. , Design and experimental validation of an active acoustic liner aircraft engine noise reduction, Euro-Mediterranean Conference on Structural Dynamics and Vibroacoustics, Sevilla, Spain, 25-27.4.2017.

116. [McCarthy2016] McCarthy, B., Sound Systems: Design and Optimization. Modern Techniques and tools for sound system design and alignment. Third Edition, Focal Press, 2016.

117. [Mnih2015] Mnih, V.; Kavukcuoglu, K.; Silver, D. et al., Human-level control through deep reinforcement learning, Nature 518, 529–533, 2015, doi:10.1038/nature14236.

118. [Mobin2016] Mobin, S.; Bruna, J., Voice conversion using convolutional neural networks, ArXiv preprint no. 1610.08927, 2016, url: https://arxiv.org/abs/1610.08927

119. [Moreau2009] Moreau, A.-S.; Lissek, Hervé; Boulandet, R., Study of an Electroacoustic Absorber, COMSOL Conference 2009, Milan, Italy, 14-16.10.2009.

120. [Müller2008] Müller, S., Measuring Transfer-Functions and Impulse Responses. In: Havelock D., Kuwano S., Vorländer M. (eds) Handbook of Signal Processing in Acoustics. Springer, New York, 2008. https://doi.org/10.1007/978-0-387-30441-0_5

121. [Naranjo-Alcazar2020] Naranjo-Alcazar, J., Perez-Castanos, S., Zuccarello, P., Cobos, M., Acoustic Scene Classification with Squeeze-Excitation Residual Networks, IEEE Access 8, 112287–112296, 2020, doi: 10.1109/ACCESS.2020.3002761.

122. [Nataf2013] Nataf, F., Absorbing boundary conditions and perfectly matched layers in wave propagation problems, from Direct and Inverse problems in Wave Propagation and Applications, Radon Ser. Comput. Appl. Math. 14, 219-231, de Gruyter, 2013.

123. [Nguyen2017] Nguyen, V., D.; Jansson, J.; Leoni, M.; Janssen, B.; Goude, A.; Hoffman, J., Modelling of rotating vertical axis turbines using a multiphase finite element method, MARINE 2017: Computational Methods in Marine Engineering, Nantes, France, 15-17.5.2017.

124. [Niedermayer2015] Niedermayer, U., Bench Measurements and Simulations of Beam Coupling Impedance, CAS-CERN Accelerator School on Intensity Limitation in Particle Beams, Geneva, Switzerland, 2-11.11.2015.

125. [Oberst2010] Oberst, S.; Lai, J., C., S,., Numerical methods for simulating brake squeal noise, 20th International Congress on Acoustics 2010, ICA 2010, Sydney, Australia, 23-27.8.2010.

126. [Oord2016] van den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, Nal; Sr., A.; Kavukcuoglu, K., WaveNet: A Generative Model for Raw Audio, ArXiv preprint no. 1609.03499, 2016, url: https://arxiv.org/abs/1609.03499

127. [Owsinski2009] Owsinski, B., The Recording Engineer's Handbook. Second Edition, Cengage Learning, 2009.

128. [Owsinski2017a] Owsinski, B., The Mastering Engineer's handbook, fourth

edition, Bobby Owsinski Media Group Publishing, 2017.

129. [Owsinski2017b] Owsinski, B., The Mixing Engineer's Handbook, fourth edition, Bobby Owsinski Media Group Publishing, 2017.

130. [Patraquim2017] Patraquim, R., Godinho L., Mendes P., Redondo J., Design and optimization of sound diffusers using RBF-based shapes and genetic algorithms, TecniAcustica - A Coruña 2017, Coruña, Spain, 4-6.10.2017.

131. [Phunpeng2015] Phunpeng, V.; Baiz, P., M., Mixed finite element formulations for strain-gradient elasticity problems using the FEniCS environment, Finite Elements in Analysis and Design 96, 23-40, 2015, doi:10.1016/j.finel.2014.11.002.

132. [Picard2008] A. Picard, R. S. Davis, M. Gläser, and K. Fujii, "Revised formula for the density of moist air (CIPM-2007)," Metrologia 45 (2), 149–155, 2008, doi: 10.1088/0026-1394/45/2/004.

133. [Pilch2020] A. Pilch, "Optimization-based method for the calibration of geometrical acoustic models," Appl. Acoust. 170, 2020, doi: 10.1016/j.apacoust.2020.107495.

134. [Pogson2010] Pogson M.; Whittaker D.; Ghering G.; Cox T.; Hughes R.; Angus J., Diffusive benefits of cylinders in front of a Schroeder diffuser, The Journal of the Acoustical Society of America 128 (3), 1149-54, 2010.

135. [Pompei2009] Pompei, A., Sumbatyan, M.A. & Todorov, N.F. Computer models in room acoustics: The ray tracing method and the auralization algorithms. Acoust. Phys. 55, 821 (2009). https://doi.org/10.1134/S1063771009060177

136. [Redondo2007] Redondo J.; Picó R.; Roig B.; Avis M. R., Time domain simulation of sound diffusers using finite difference schemes, Acta Acustica & Acustica 93 (4), 611-622, 2007.

137. [Redondo2019] Redondo J., Herrero J., Godinho L., Patraquim R., Cox T., Application of multi-objective optimization techniques to the design of sound diffusers, 23rd International Congress on Acoustics, Aachen, Germany, 9-13.9.2019.

138. [Reinhardt2016] Reinhardt D., Cabrera D., Jung A., Watt R. (2016) Towards a Micro Design of Acoustic Surfaces. In: Reinhardt D., Saunders R., Burry J. (eds) Robotic Fabrication in Architecture, Art and Design 2016. Springer, Cham. https://doi.org/10.1007/978-3-319-26378-6_10

139. [Rivet2012] Rivet, E.; Boulandet, R.; Lissek, H.; Rigas, I., Study on room modal equalization at low frequencies with electroacoustic absorbers, Acoustics 2012, Nantes, France, 23-27.4.2012.

140. [Rivet2017] Rivet, E.; Karkar, S.; Lissek, H., Broadband Low-Frequency Electroacoustic Absorbers Through Hybrid Sensor-/Shunt-Based Impedance Control, IEEE Transactions on Control Systems Technology 25 (1), 63-72, 2017, doi:10.1109/TCST.2016.2547981.

141. [Romac2019] Romac, C.; Béraud, V., Deep Recurrent Q-Learning vs Deep Q-

Learning on a simple Partially Observable Markov Decision Process with Minecraft, ArXiv preprint no. 1903.04311, 2019, url: https://arxiv.org/abs/1903.04311.

142. [Rothbucher2013] Rothbucher, M., Veprek, K., Paukner, P., Habigt, T., Diepold, K., Comparison of head-related impulse response measurement approaches, J. Acoust. Soc. Am., 134 (2), EL223–EL229, 2013, doi: 10.1121/1.4813592.

143. [Salakhutdinov2009] Salakhutdinov, R.; Hinton, G., Deep Boltzmann Machines, 12th International Conference on Artificial Intelligence and Statistics (AISTATS) 2009, Clearwater Beach, USA, 16-18.4.2009.

144. [Sallab2016] Sallab, A., E.; Abdou, M.; Perot, E.; Yogamani, S., End-to-end deep reinforcement learning for lane keeping assist, ArXiv preprint no. 1612.04340, 2016, url: https://arxiv.org/abs/1612.04340

145. [Sheaffer2014] J. Sheaffer, M. van Walstijn, and B. Fazenda, "Physical and numerical constraints in source modeling for finite difference simulation of room acoustics," J. Acoust. Am., 135 (1), 251–261, 2014, doi: 10.1121/1.4836355.

146. [Shen2020] Shen, Y., Cao, J., Wang, J., Yang, Z., Urban acoustic classification based on deep feature transfer learning, J. Franklin Inst. 357 (1), 667–686, 2020, doi: 10.1016/j.jfranklin.2019.10.014.

147. [Schmidt2016] Schmidt, S.; Wadbro, E.; Berggren, M., Large-Scale Three-Dimensional Acoustic Horn Optimization, J. Scientific Computing 38, 2016.

148. [Schwan2017] Schwan, L.; Umnova, O.; Boutin, C., Sound absorption and reflection from a resonant metasurface: Homogenisation model with experimental validation, Wave Motion 72, 154-172, 2017, doi:10.1016/j.wavemoti.2017.02.004.

149. [Senior2015] Senior, M., Recording Secrets for the Small Studio, Focal Press, 2015.

150. [Śmigaj2015] Śmigaj, W.; Betcke, T.; Arridge, S.; Phillips, J.; Schweiger, M., Solving Boundary Integral Problems with BEM++., ACM Trans. Math. Softw. 41 (2), 2015, doi:10.1145/2590830.

151. [Spuhler2018] Spühler, J., H.; Jansson, J.; Jansson, N.; Hoffman, J., 3D Fluid-Structure Interaction Simulation of Aortic Valves Using a Unified Continuum ALE FEM Model, Frontiers in physiology 9, 2018, doi:10.3389/fphys.2018.00363.

152. [Sturm2016] Sturm, B., L.; Santos, J., F.; Ben-Tal, O.; Korshunova, I., Music transcription modelling and composition using deep learning, ArXiv preprint no. 1604.08723, 2016, url: https://arxiv.org/abs/1604.08723

153. [Su2020] J. Su, L. Zheng, and J. Lou, "Simulation and Optimization of Acoustic Package of Dash Panel Based on SEA," Shock Vib. 2020, 2020, doi: 10.1155/2020/8855280.

154. [Sun2020] H. Sun et al., "3D focusing acoustic lens optimization method using multi-factor and multi-level orthogonal test designing theory," Appl. Acoust. 170,

2020, doi: 10.1016/j.apacoust.2020.107538.

155. [Sutton1998] Sutton, R., S.; Barto, A., G., Reinforcement Learning: An Introduction, The MIT Press, 1998.

156. [Szczodrak2016] Szczodrak, M.; Kurowski, A.; Kotus, J.; Czyżewski, A.; Kostek, B., A system for acoustic field measurement employing cartesian robot, Metrology and Measurement Systems 23 (3), 333-343, 2016, doi:10.1515/mms-2016-0037.

157. [Tang2020] Tang, Z., Bryan, N., Li, D., Langlois, T., Manocha, D., Scene-Aware Audio Rendering via Deep Acoustic Analysis, IEEE Transactions on Visualization and Computer Graphics, 26 (5), 1991-2001, doi: 10.1109/TVCG.2020.2973058.

158. [Thede2004] Thede, S., An introduction to genetic algorithms, J. Comput. Sci. Coll. 20 (1), 115-123, 2014.

159. [Thoma2016] Thoma, M., Creativity in Machine Learning, ArXiv preprint no. 1601.03642, 2016, url: https://arxiv.org/abs/1601.03642, access date: 10.02.2021.

160. [Toledo2015] Toledo R.; Aznárez J.; Maeso O.; Grenier D., Optimization of thin noise barier designs using Evolutionary Algorithms and a Dual BEM Formulation, Journal of Sound and Vibration 334 (6), 219-238, 2015.

161. [Tournour2000] Tournour, M.; Tournour, M.; Cremers, L.; Guisset, P.; Augusztinovicz, F.; Marki, F., INVERSE NUMERICAL ACOUSTICS BASED ON ACOUSTIC TRANSFER VECTORS, Seventh International Congress on Sound and Vibration, ICSV-7, Garmisch-Partenkirchen, Germany, 4-7.7.2000.

162. [Usunier2016] Usunier, N.; Synnaeve, G.; Lin, Z.; Chintala, S., Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks, ArXiv preprint no. 1609.02993, 2016, url: https://arxiv.org/abs/1609.02993

163. [Valada2018] Valada, A., Spinello, L. Burgard, W., Deep Feature Learning for Acoustics-Based Terrain Classification, 21–37, 2018, doi: 10.1007/978-3-319-60916-4_2.

164. [Vandekerckhove2016] Vandekerckhove, S.; Wells, G., N.; De Gersem, H.; Abeele, K., V., D., Automatic calibration of damping layers in finite element time domain simulations, ArXiv preprint no. 1601.07941, 2016, url: https://arxiv.org/abs/1601.07941.

165. [Varshavskaya2008] Varshavskaya, P.; Kaelbling, L. P.; Rus, D., Automated Design of Adaptive Controllers for Modular Robots using Reinforcement Learning, The International Journal of Robotics Research 27 (3-4), 505-526, 2008, doi:10.1177/0278364907084983.

166. [Vinyals2017] Vinyals, O.; Ewalds, T.; Bartunov, S.; Georgiev, P.; Vezhnevets, A., S.; Yeo, M.; Quan, J. , Starcraft ii: A new challenge for reinforcement learning, ArXiv preprint no. 1708.04782, 2017, url: https://arxiv.org/abs/1708.04782

167. [Webb2011] C. J. Webb and S. Bilbao, “Computing room acoustics with CUDA

- 3D FDTD schemes with boundary losses and viscosity," ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc., 317–320, 2011, doi: 10.1109/ICASSP.2011.5946404.

168. [Weyna2001] Weyna, S., Rozpływ Energii Akustycznych źródeł rzeczywistych, Wydawnictwa Naukowo-Techniczne, 2001. (in polish)

169. [Wyner2013] Wyner, J., Audio Mastering: Esential Practices, Berklee Press, 2013.

170. [Xiao2005] Xiao, L.; Cox, T., J.; Avis, M., R., Active diffusers: some prototypes and 2D measurements, Journal of Sound and Vibration 258 (1-2), 321-339, 2005, doi:10.1016/j.jsv.2004.08.031.

171. [Xu2012] J. Xu, Q. Wei, S. Peng, and Y. Yu, "Error of saturation vapor pressure calculated by different formulas and its effect on calculation of reference evapotranspiration in high latitude cold region," Procedia Eng. 28, 43–48, 2012, doi: 10.1016/j.proeng.2012.01.680.

172. [Yeh2018] Yeh, C. K.; Hsieh, C. Y.; Lin, H. T., Automatic bridge bidding using deep reinforcement learning, IEEE Transactions on Games 10 (4), 365-377, 2018.

173. [Yuan1997] X. Yuan, D. Borup, J. W. Wiskin, M. Berggren, R. Eidens, and S. A. Johnson, "Formulation and validation of Berenger's PML absorbing boundary for the FDTD simulation of acoustic scattering," IEEE Trans. Ultrason. Ferroelectr. Freq. Control 44 (4), 816–822, 1997.

174. [Yuksel2012] Yuksel, E.; Kamci, G.; Basdogan, I., Vibro-Acoustic Design Optimization Study to Improve the Sound Pressure Level Inside the Passenger Cabin, ASME. J. Vib. Acoust. 134 (6), 2012, doi:10.1115/1.4007678.

175. [Zampolli2008] Zampolli, M.; Malm, N.; Tesei, A., Improved Perfectly Matched Layers for Acoustic Radiation and Scattering Problems, COMSOL Conference 2008, Hannover, Germany, 4-6.11.2008.

176. [Zera1997] Zera, J., Brammer, A., Pan, G., Comparison between subjective and objective measures of active hearing protector and communication headset attenuation, J. Acoust. Soc. Am., 101 (6), 3486–3497, 1997, doi: 10.1121/1.418356.

177. [Zolzer2011] Zölzer, U., DAFX: Digital Audio Effects, Second Edition, John Wiley and Sons Ltd, 2011.

APPENDIX A: PYTHON PROCEDURES FOR GENERATION OF K MATRIX

This appendix contains the source code used for the generation of **K** matrix and all derivative matrices used for FDTD simulation. As this task can very often be tedious and complicated, a set of tools was prepared. They allow for the fast, automated creation of **K** matrices that are compatible with assumptions of the homogenous equation of the FDTD method and automatically ensure that all K values correspond to valid values of boundary conditions. Additionally, procedures for the visualization of generated shapes are also provided in this fragment of source code. The code is assumed to be stored in a file named Kgen.py. This is important, as the code in Appendix B, which is responsible for carrying out FDTD simulation, imports code from this appendix from the file of such name.

```
# ------------------------------------------------- #
# A library of functions used for creation of geometries
# of objects simulated with the FDTD method
# ------------------------------------------------- #

from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib.pyplot as plt
import copy as cp
import mayavi.mlab
import numba as nb
import time

# -----------------------------------------------
# Helper functions used for shapes generation
# -----------------------------------------------

# A function for creation of an object "mold" which takes a form
# of a matrix, where 1 is a voxel occupied by an object, and 0 is a voxel
# which is not occupied by an object to a form of K matrix required
# by a FDTD method
# (5 - a surface, 4 - an edge, 3 - a vortex)
#
# Voxels of the mold HAVE TO be assigned value of 1!
#
# Moreover, any shapes in the computational domain CAN NOT
# have thickness of 1 voxel, as this will cause the bake() function
# to behave in an unspecified manner
# (thickness of any object has to be at least 2)

def bake(shape_mold):
    # Bidirectional differentiating - permits detection of corners and
edges
    def bidirect_differencing(mtx, axis, padding='prepend'):
        output  = np.zeros_like(mtx)

        if padding == 'prepend':
            kwargs = {'prepend':0}
        elif padding == 'append':
            kwargs = {'append':0}
        else:
            raise RuntimeError(f'bad value specified for padding keyword
argument: {padding}')
```

```
        mtx_rev = np.flip(mtx,axis=axis)
        output += np.diff(mtx,axis=axis, **kwargs)
        output += np.flip(np.diff(mtx_rev,axis=axis, **kwargs),axis=axis)

        return output

    # Application of bidirectional differentiation on the whole 3D space -
    # the result of this operation allows detection of surfaces, edges, and
vertices
    def unidirectional_differencing(shape_mold):
        output  = np.zeros_like(shape_mold)
        output += bidirect_differencing(shape_mold, 0)
        output += bidirect_differencing(shape_mold, 1)
        output += bidirect_differencing(shape_mold, 2)
        return output

    # Application of non-directional differentiating and limitation
    # of results only to the volume taken by the mold
    # geometrii
    diff_stencil = unidirectional_differencing(shape_mold)
    diff_stencil[np.where(shape_mold==0)] = 0

    # Matching of the generated geometry to their
    # corresponding boundary conditions (5 - surface,
    # 4 - edge, 3 - vertex, 0 – interior of an object,
    # 6 – propagation medium)
    baked_object = np.zeros_like(diff_stencil)
    baked_object[diff_stencil==1] = 5
    baked_object[diff_stencil==2] = 4
    baked_object[diff_stencil==3] = 3
    baked_object[shape_mold  ==0] = 6

    return baked_object

# Many operations are based on embedding a "form"
# made of many cuboids - hence a base function for
# easy embedding of forms, and setting voxels of the form
# to desired values (value), usually 1 (required by the bake function)

def embed_cuboid(K, x_pos, y_pos, z_pos, width, length, height, value=1):
    K[x_pos:x_pos+width, y_pos:y_pos+length, z_pos:z_pos+height] = value
    return K

# ------------------------------------------------
# Shape generation
# ------------------------------------------------

# Axes of the simulation:
#   z - height of the computational domain in a FDTD simulation
#   y – depth of the diffuser in the domain
#   x – width of the diffuser
# Object of the computational domain which fulfills assumptions of
# the FDTD equation with K matrix
def make_computational_domain(x_dim,y_dim,z_dim):
    # Allocation of space for the computational domain
    plate_obj = np.zeros((x_dim, y_dim, z_dim))

    # Border of the domain is a surface-type boundary condition
    plate_obj = embed_cuboid(plate_obj, 1, 1, 1, x_dim-2, y_dim-2, z_dim-2,
value=1)
```

```
    # Conversion of the form to the target format.
    plate_obj = bake(plate_obj)

    # Zeros have to be set on the outside of computational domain, in other
    # case, the equation will not be stable.
    plate_obj[plate_obj==6] = 0

    # Inside of the domain is marked as a propagation medium
    plate_obj = embed_cuboid(plate_obj, 2, 2, 2, x_dim-4, y_dim-4, z_dim-4,
value=6)

    return plate_obj

# Embeding f the shape(diffuser) into a computational domain in the given
position in the 3D space.
def embed_shape_into_domain(computational_domain, beta, shape_obj,
admittance, x_pos=0, y_pos=0, z_pos=0):

    computational_domain[
        x_pos:(x_pos+shape_obj.shape[0]),
        y_pos:(y_pos+shape_obj.shape[1]),
        z_pos:(z_pos+shape_obj.shape[2])] = shape_obj[:,:,:]

    for vox_type in [5,4,3,0]:
        mask_args = list(np.where(shape_obj==vox_type))
        mask_args[0]    += x_pos
        mask_args[1]    += y_pos
        mask_args[2]    += z_pos
        mask_args        = tuple(mask_args)
        beta[mask_args]  = admittance

    return computational_domain, beta
# Generator of skyline-type diffuser with a 1D pattern
def
get_1D_Skyline_diffuser(pattern=[0,1,0,0,1,2,1,1,2,1,0,0,1,0,0,1,0,0,1,2,1,
1,2,1,0,0,1,0],segment_height=15,segment_width=6,diffuser_height = 120):
    # Input always should have numpy.ndarray type
    if type(pattern) == list:
        pattern = np.array(pattern)

    # calculation of information about diffuser segments dimmensions
    num_segments    = len(pattern)
    segment_heights = (pattern+1)*segment_height
    segment_heights = segment_heights.tolist()
    diffuser_width = num_segments*segment_width
    diffuser_elem_height = num_segments*segment_width

    # Preparation of the diffuser form, and cuboids which will become
    # segments of the diffuser
    shape_mold =
np.zeros((diffuser_width,diffuser_elem_height,diffuser_height))
    for i, height in enumerate(segment_heights):
        shape_mold = embed_cuboid(shape_mold,i*segment_width,0,0,
segment_width, height, diffuser_height)

    # Conversion of a form to the final definition of a diffuser.
    baked_object = bake(shape_mold)

    # Removal of vertices, and edges detected by the bidirectional
    # differentiating (they are not needed in K matrix specifications)
```

```
    baked_object[baked_object==3] = 6
    baked_object[baked_object==4] = 6

    return baked_object

# Generation of classical 2D Schroeder diffuser
def generate_2D_Skyline_diffuser(pattern =
[[1,0,2,1],[0,0,3,0],[2,3,0,1]],element_size=10,element_seg_depth=10):
    # Input always should have numpy.ndarray type

    if type(pattern) == list:
        pattern = np.array(pattern)

    # Dimensions of the diffuser
    diffuser_height = pattern.shape[0]*element_size
    diffuser_width  = pattern.shape[1]*element_size

    # Adaptation of the pattern of cavities to the generator loop of the
    # diffuser shape(it is necessary, because we assume that 0,0 point is
    # positioned in the lower left corner of a diffuser
    flipped_pattern = np.flip(pattern, axis=0)
    flipped_pattern = np.flip(flipped_pattern, axis=1)
    flipped_pattern = flipped_pattern + 1

    # Preparation of a matrix in which the geometry of the diffuser
    # will be embedded
    diffuser_depth = np.max(flipped_pattern)*element_seg_depth
    diffuser_obj   =
np.zeros((diffuser_width,diffuser_depth,diffuser_height))

    # The loop generating wells of the diffuser
    for col_idx in range(flipped_pattern.shape[0]):
        for row_idx in range(flipped_pattern.shape[1]):

            # Coordinates of wells in the 3D space
            element_pos_x = element_size*row_idx
            element_pos_z = element_size*col_idx
            element_pos_y = 0

            # Dimensions of the well
            element_size_x = element_size
            element_size_z = element_size
            element_size_y =
flipped_pattern[col_idx,row_idx]*element_seg_depth

            # Cutting of the well in the diffusor object
            diffuser_obj = embed_cuboid(diffuser_obj,
                element_pos_x,element_pos_y,element_pos_z,
                element_size_x,element_size_y,element_size_z,
                value=1)

    # Conversion of the shape to the target format.
    diffuser_obj = bake(diffuser_obj)

    # Removal of vertices, and edges detected by the bidirectional
    # differentiating (they are not needed in K matrix specifications)
    diffuser_obj[diffuser_obj==3] = 6
    diffuser_obj[diffuser_obj==4] = 6

    return diffuser_obj
```

```
# A flat plate object - needed for measurement of scattering, and diffusion
# coefficients.
def generate_plate(x_dim=50,y_dim=50,z_dim=100):
    # Wygenerowanie formy płyty
    plate_obj = np.zeros((x_dim, y_dim, z_dim))
    plate_obj = embed_cuboid(plate_obj, 0, 0, 0, x_dim, y_dim, z_dim,
value=1)

    # Conversion of matrix to format in which edges, and vertices
    # can be differentiated
    plate_obj = bake(plate_obj)

    # Removal of vertices, and edges detected by the bidirectional
    # differentiating (they are not needed in K matrix specifications)
    plate_obj[plate_obj==3] = 6
    plate_obj[plate_obj==4] = 6

    return plate_obj
# Object of classic 2D Schroeder diffuser
def generate_2D_Schroedder_diffuser(pattern =
[[1,0,2,1],[0,0,3,0],[2,3,0,1]],well_size=10,well_seg_depth=10,diffuser_dep
th=40):

    # Input always should have numpy.ndarray type
    if type(pattern) == list:
        pattern = np.array(pattern)

    # Thickness of the diffuser wells - for the bake procedure
    # to work properly it cannot be less than 3. On the other hand
    # it should not be too small, so the best value of thickness is 3
    WELL_WALL_WIDTH = 3

    # A distance between corresponding points of each well
    well_increment  = (WELL_WALL_WIDTH+well_size)

    # Shape of the diffuser
    diffuser_height = pattern.shape[0]*well_increment + WELL_WALL_WIDTH
    diffuser_width  = pattern.shape[1]*well_increment + WELL_WALL_WIDTH

    # Adaptation of the pattern of cavities to the generator loop of the
    # diffuser shape(it is necessary, because we assume that 0,0 point is
    # positioned in the lower left corner of a diffuser
    flipped_pattern = np.flip(pattern, axis=0)
    flipped_pattern = np.flip(flipped_pattern, axis=1)

    # Preparation of a matrix in which the geometry of the diffuser
    # will be embedded
    diffuser_obj = np.ones((diffuser_width,diffuser_depth,diffuser_height))

    # The loop generating wells of the diffuser
    for col_idx in range(flipped_pattern.shape[0]):
        for row_idx in range(flipped_pattern.shape[1]):

            # Coordinates of wells in the 3D space
            well_pos_x = WELL_WALL_WIDTH+well_increment*row_idx
            well_pos_z = WELL_WALL_WIDTH+well_increment*col_idx

            # The y direction has to be modulated to modulate
            # depth of wells
            well_pos_y = WELL_WALL_WIDTH +
flipped_pattern[col_idx,row_idx]*well_seg_depth
```

```
                # Size of well
                well_size_x = well_size
                well_size_z = well_size

                # The y direction has to be modulated to modulate
                # depth of wells
                well_size_y = diffuser_depth-WELL_WALL_WIDTH-
flipped_pattern[col_idx,row_idx]*well_seg_depth

               # Cutting of the well in the diffusor object
                diffuser_obj = embed_cuboid(diffuser_obj,
                    well_pos_x,well_pos_y,well_pos_z,
                    well_size_x,well_size_y,well_size_z,
                    value=0)

    # Conversion of the shape to the target format.
    diffuser_obj = bake(diffuser_obj)


    # Removal of vertices, and edges detected by the bidirectional
    # differentiating (they are not needed in K matrix specifications)
    diffuser_obj[diffuser_obj==3] = 6
    diffuser_obj[diffuser_obj==4] = 6

    return diffuser_obj

# ------------------------------------------------
# Generation of the beta matrix
# ------------------------------------------------

# Assignment of admittance value (beta) to the boundary conditions
# of the computational domain
# It is important to call this function on NEWLY CREATED MATRIX.
# if it is necessary to assign admittance values to walls of the domain.
# (there cannot be any objects embedded into the K matrix)
def make_beta(K, volume_init_val=0, all_walls_init=None,
    beta_wall_left=None,   beta_wall_right=None,
    beta_wall_bottom=None, beta_wall_top=None,
    beta_wall_down=None,   beta_wall_up=None):

    # Inicjalizacja macierzy impedancji
    beta          = np.ones_like(K)*volume_init_val

    if all_walls_init is None:
        walls_init_values = np.zeros(6)
    else:
        walls_init_values = np.ones(6)*all_walls_init

    for idx, value in
enumerate([beta_wall_top,beta_wall_bottom,beta_wall_left,
    beta_wall_right,beta_wall_up,beta_wall_down]):
        if value is not None:
            walls_init_values[idx] = value

    beta[0:2,:,:]   = walls_init_values[0]
    beta[:,0:2,:]   = walls_init_values[2]
    beta[:,:,0:2]   = walls_init_values[4]
    beta[-2:beta.shape[0],:,:] = walls_init_values[1]
    beta[:,-2:beta.shape[1],:] = walls_init_values[3]
    beta[:,:,-2:beta.shape[2]] = walls_init_values[5]
```

```
    return beta

def make_BK(K,lam,beta):
    BK     = (6-K)*lam*beta/2
    return BK

# ------------------------------------------------
# Geometry visualization
# ------------------------------------------------

# Three-dimensional preview of generated diffuser geometries.
# (Based on Matplotlib library)
def plot_with_color(K,ax, node_type_number, color):
    colors = np.empty(K.shape, dtype='object')
    voxels = np.zeros_like(K)
    colors[K==node_type_number] = color
    voxels[K==node_type_number] = 1
    ax.voxels(voxels, facecolors=colors, edgecolor='k', alpha = 0.5)

# Three-dimensional preview of generated diffuser geometries.
# (Based on Mayavi library)
def show_shape(K, opacity = 1, mode='voxel', show_result=True):

    if mode == 'voxel':
        objects = []
        objects.append({'K_number':5,'color':(0.4, 0.4, 1)}) # powierzchnie
        objects.append({'K_number':4,'color':(0.4, 1, 0.4)}) # krawędzie
        objects.append({'K_number':3,'color':(1, 0.4, 0.4)}) # wierzchołki

        # Voxels of the propagation medium (6)
        # are not visualized

        for obj_def in objects:
            xx, yy, zz = np.where(K == obj_def['K_number']) # surfaces
            mayavi.mlab.points3d(xx, yy, zz, mode="cube",
color=obj_def['color'], scale_factor=1, opacity=opacity, line_width=5)

    elif mode=='scalar_field':
        scalar_field = mayavi.mlab.pipeline.scalar_field(K,opacity=opacity)
        env_plot     = mayavi.mlab.pipeline.volume(scalar_field)
        mayavi.mlab.colorbar(orientation='vertical')

    elif mode=='scalar_cut_plane':
        scalar_field = mayavi.mlab.pipeline.scalar_field(K,opacity=opacity)
        mayavi.tools.modules.scalar_cut_plane(scalar_field,
plane_orientation='x_axes')
        mayavi.tools.modules.scalar_cut_plane(scalar_field,
plane_orientation='y_axes')
        mayavi.tools.modules.scalar_cut_plane(scalar_field,
plane_orientation='z_axes')
        mayavi.mlab.colorbar(orientation='vertical')

    else:
        raise RuntimeError('Wybrano niewłaściwą opcję wizualizacji.')

    if show_result:
        # Show the result
        mayavi.mlab.show()
```

APPENDIX B: IMPLEMENTATION OF A FDTD ACOUSTIC SIMULATION METHOD

The Appendix contains a set of procedures for carrying out FDTD simulations. The source code provided in this appendix provides 3 interfaces to the GPU-accelerated implementation of the FDTD method. It is also assumed that **K** matrix was generated with the use of code presented in Appendix A. Some procedures from this appendix are also necessary for the code presented in this appendix, and therefore, the code is imported from a file that is assumed to be named Kgen.py.

```
# ------------------------------------------------------
# Computational engine for the GPU-accelerated FDTD method
# ------------------------------------------------------

import numpy as np
import matplotlib.pyplot as plt
import numba as nb
from   numba import prange, cuda, float32
from   scipy.interpolate import RectBivariateSpline
from   mayavi import mlab
import copy as cp
import os
import cv2
from numba import cuda
from   _imports.sim_core.Kgen import *

# ------------------------------------------------------
# Helper functions
# A function for checking actual index for which calculations should be
# carried out - it prevents from performing computations if the index
# is outside the computational domain

@cuda.jit('boolean(int32,int32,int32,int32,int32,int32)', device=True)
def pressure_outside_domain(ll,mm,ii,sh_ll,sh_mm,sh_ii):
    if ll < 1 or ll >= sh_ll-1:
        return True
    if mm < 1 or mm >= sh_mm-1:
        return True
    if ii < 1 or ii >= sh_ii-1:
        return True
    return False

# A CUDA computation step
@cuda.jit
def FDTD_step_CUDA(K,courant_number,X,T,sigma_arr,c,rho,p_all_steps,BK,
p_current):

    cuda.syncthreads()
    ll,mm,ii = cuda.grid(3)
    if pressure_outside_domain(ll,mm,ii,*(p_current.shape)): return
    cuda.syncthreads()

    damping_A = rho[0]*c[0]*c[0]*sigma_arr[ll, mm, ii]*T[0] + sigma_arr[ll,
mm, ii]*T[0]/rho[0]
    damping_B = c[0]*c[0]*sigma_arr[ll, mm, ii]*sigma_arr[ll, mm,
ii]*T[0]*T[0]

    # Application of the discretized wave equation with taking into
    # account boundary conditions, and dumping factor of PML layers.
```

```
    p_current[ll,mm,ii] = (
    (2-K[ll, mm,
ii]*courant_number[0]*courant_number[0])*p_all_steps[ll,mm,ii,1]
    + (BK[ll, mm, ii] + damping_A/2 -1)*p_all_steps[ll,mm,ii,2]
    + courant_number[0]*courant_number[0]*(p_all_steps[ll+1,mm,ii,1] +
p_all_steps[ll-1,mm,ii,1] + p_all_steps[ll,mm+1,ii,1] + p_all_steps[ll,mm-
1,ii,1] + p_all_steps[ll,mm,ii+1,1] + p_all_steps[ll,mm,ii-1,1])
    ) / (1+BK[ll, mm, ii] + damping_A/2 + damping_B)

# Update of matrix containing pressure distribution data from consecutive
# simulation steps
@cuda.jit
def shift_pressure_matrices_CUDA(p_all_steps):
    ll,mm,ii = cuda.grid(3)
    if pressure_outside_domain(ll,mm,ii,*(p_all_steps.shape[0:3])): return
    p_all_steps[ll,mm,ii,2] = p_all_steps[ll,mm,ii,1]
    p_all_steps[ll,mm,ii,1] = p_all_steps[ll,mm,ii,0]

# A procedure for obtaining actual pressure distribution
@cuda.jit
def update_pressure_CUDA(p_all_steps,p_current):
    ll,mm,ii = cuda.grid(3)
    if pressure_outside_domain(ll,mm,ii,*(p_current.shape)): return
    p_all_steps[ll,mm,ii,0] = p_current[ll,mm,ii]

# Injection of next excitation signal sample (soft source)
@cuda.jit
def inject_source_CUDA(p_all_steps,source_pos_disc,excitation_sample):
    ll,mm,ii = cuda.grid(3)
    if pressure_outside_domain(ll,mm,ii,*(p_all_steps.shape[0:3])): return
    p_all_steps[source_pos_disc[0],source_pos_disc[1],source_pos_disc[2],1]
= p_all_steps[source_pos_disc[0],source_pos_disc[1],source_pos_disc[2],1] +
excitation_sample[0];

# Enforcement of zero pressure values inside scattering objects
# in matrix K_glMem
@cuda.jit
def dead_spaces_CUDA(p_all_steps,zero_K):
    i = cuda.grid(1)

    # Disable the function if index is outside computation domain.
    if i < 0 or i >= zero_K.shape[1]:
        return

    # Enforcement of zero pressure values inside objects.
    p_all_steps[zero_K[0,i],zero_K[1,i],zero_K[2,i],0] = 0
    p_all_steps[zero_K[0,i],zero_K[1,i],zero_K[2,i],1] = 0
    p_all_steps[zero_K[0,i],zero_K[1,i],zero_K[2,i],2] = 0

@cuda.jit
def read_result_CUDA(p_all_steps,p_plane_xy,p_plane_yz,source_pos_disc):
    ll,mm,ii = cuda.grid(3)
    if pressure_outside_domain(ll,mm,ii,*(p_all_steps.shape[0:3])): return
    p_plane_xy[ll,mm] = p_all_steps[ll,mm,source_pos_disc[2],0]
    p_plane_yz[mm,ii] = p_all_steps[source_pos_disc[0],mm,ii,0]

# -----------------------------------------------------
# An object interface for GPU-accelerated FDTD solver
class FDTDSimulation:
    def __init__(self, K, beta, source_pos_disc, excitation_signal,
courant_number, X, T, c=340, rho = 1.225, PML_damping = 0.15,
```

```
    PML_width = 50, TPB=8):

        # Update of the object fields:
        #   computational domain geometry and medium properties
        self.K                  = K
        self.c                  = c
        self.rho                = rho
        self.BK                 = make_BK(K,courant_number,beta)
        self.zero_K             = np.array(np.where(K==0))
        #   excitation signal:
        self.source_pos_disc    = source_pos_disc
        self.excitation_signal = excitation_signal
        #   simulation resolution:
        self.courant_number     = courant_number
        self.X                  = X
        self.T                  = T
        #   PML layers:
        self.PML_damping        = PML_damping
        self.PML_width          = PML_width
        #   jobs distribution on  GPU:
        self.TPB                = TPB
        #   simulation frame number
        self.sim_frame_number  = 0

        # Allocation of pressure matrix
        # The main variable contains 3 consecutive pressure
        # distributions in the computational domain
        self.p_all_steps  = np.zeros(list(self.K.shape)+[3])
        #   3D pressure distribution (1st step)
        self.p_current = np.zeros(self.K.shape)
        #   2D pressure distribution
        self.p_plane_xy = np.zeros((self.K.shape[0],self.K.shape[1]))
        self.p_plane_yz = np.zeros((self.K.shape[1],self.K.shape[2]))

        # A variable for passing current value of the excitation signal
        self.excitation_sample = self.excitation_signal[0]

        # Allocation of FDTD calculation threads on the GPU
        self.num_threads = [self.TPB,self.TPB,self.TPB]
        self.num_blocks  = []
        for i,n_thr in enumerate(self.num_threads):

self.num_blocks.append(int(np.ceil(self.p_current.shape[i]/n_thr)))

        # PML layers
        self.PML_sigma       = np.zeros_like(K)
        # Damping profile (the damping is greater if the point
        # is deeper into the PML)
        self.damping_profile =
np.power(np.linspace(1,0,self.PML_width),1.4)*self.PML_damping
        # Application of PML profile to all borders of computational
domain.
        for i in range(self.PML_width):
            local_damping_fctr = self.damping_profile[i]
            self.PML_sigma[i,:,:]                              =
local_damping_fctr
            self.PML_sigma[self.PML_sigma.shape[0]-i-1,:,:] =
local_damping_fctr
            self.PML_sigma[:,i,:]                              =
local_damping_fctr
```

```
            self.PML_sigma[:,self.PML_sigma.shape[1]-i-1,:] =
local_damping_fctr
            self.PML_sigma[:,:,i]                           =
local_damping_fctr
            self.PML_sigma[:,:,self.PML_sigma.shape[2]-i-1] =
local_damping_fctr

        # Variables allocated in the GPU memory
        self.K_glMem                = cuda.to_device(self.K)
        self.c_glMem                = cuda.to_device(np.array([self.c]))
        self.rho_glMem              = cuda.to_device(np.array([self.rho]))
        self.BK_glMem               = cuda.to_device(self.BK)
        self.zero_K_glMem           = cuda.to_device(self.zero_K)
        self.source_pos_disc_glMem  = cuda.to_device(self.source_pos_disc)
        self.excitation_sample_glMem =
cuda.to_device(np.array([self.excitation_sample]))
        self.courant_number_glMem                =
cuda.to_device(np.array([self.courant_number]))
        self.X_glMem                = cuda.to_device(np.array([self.X]))
        self.T_glMem                = cuda.to_device(np.array([self.T]))
        self.sigma_glMem            =
cuda.to_device(np.array(self.PML_sigma))
        self.p_all_steps_glMem      = cuda.to_device(self.p_all_steps)
        self.p_current_glMem        = cuda.to_device(self.p_current)
        self.p_plane_xy_glMem       = cuda.to_device(self.p_plane_xy)
        self.p_plane_yz_glMem       = cuda.to_device(self.p_plane_yz)

    def step(self):
        # Execution of single FDTD step

FDTD_step_CUDA[self.num_blocks,self.num_threads](self.K_glMem,self.courant_
number_glMem,self.X_glMem,self.T_glMem,self.sigma_glMem,self.c_glMem,self.r
ho_glMem,self.p_all_steps_glMem,self.BK_glMem, self.p_current_glMem)

        # Assignment of the current step result to the matrix containing
"current state"

update_pressure_CUDA[self.num_blocks,self.num_threads](self.p_all_steps_glM
em,self.p_current_glMem)

        # Enforcement of zero pressure inside objects
        # It is necessary to define 1D thread specifications
        # specific to this task
        enforcer_num_threads = 32
        enforcer_num_blocks  = int(np.ceil(self.zero_K.shape[1]/32))

dead_spaces_CUDA[enforcer_num_blocks,enforcer_num_threads](self.p_all_steps
_glMem,self.zero_K_glMem)

        # Reading of the 2D pressure distributions (for later readout from
th ep_plane variable)

read_result_CUDA[self.num_blocks,self.num_threads](self.p_all_steps_glMem,s
elf.p_plane_xy_glMem,self.p_plane_yz_glMem,self.source_pos_disc_glMem)

        # Injection of the next excitation value:
        #   sending excitation value

self.excitation_sample_glMem.copy_to_device(np.array([self.excitation_signa
l[self.sim_frame_number]]))
        self.sim_frame_number += 1
```

```
        #   source update

inject_source_CUDA[self.num_blocks,self.num_threads](self.p_all_steps_glMem
,self.source_pos_disc_glMem,self.excitation_sample_glMem)

        # Shifting of simulation time steps

shift_pressure_matrices_CUDA[self.num_blocks,self.num_threads](self.p_all_s
teps_glMem)

    def get_pressure_plane(self, plane_name = 'xy'):
        # The data about selected 2D plane
        # (height is the same as source height)
        if plane_name == 'xy':
            self.p_plane_xy_glMem.copy_to_host(self.p_plane_xy)
            return self.p_plane_xy
        elif plane_name == 'xz':
            self.p_plane_yz_glMem.copy_to_host(self.p_plane_yz)
            return self.p_plane_yz
        elif plane_name == 'both':
            self.p_plane_xy_glMem.copy_to_host(self.p_plane_xy)
            self.p_plane_yz_glMem.copy_to_host(self.p_plane_yz)
            return self.p_plane_xy, p_plane_yz
        else:
            raise RuntimeError(f'A bad plane name for readout was
specified: ({plane_name})')

    def get_pressure_3D(self):
        # Pressure data from the selected 2D plane:
        self.p_current_glMem.copy_to_host(self.p_current)
        return self.p_current

def draw_pressure_2D_cv(K, p_plane, objects_mask, source_pos_disc,
observerPosD, measurement_points, img_scale=None, axe_names=['A','B'],
transpose=False):

    # Automatic gain regulation - to make shure, that details are visible
    # if excitation has especially high values
    ref = np.max(np.abs(p_plane))
    if ref < 1: ref = 1

    # Calculation of black-and-white visualizations
    # of given simulation plane.
    bw_image = (127+(p_plane/ref)*127).astype(np.uint8)
    bw_image[objects_mask] = 255

    # Marking of the source position (white dot)
    bw_image[source_pos_disc[0]-1:source_pos_disc[0]+1,source_pos_disc[1]-
1:source_pos_disc[1]+1] = 255

    # Marking of the impulse response measurement point(black dots)
    for coords in measurement_points:

        # Coordinates are floating point numbers,
        # so the antialiasing will be necessary for proper display of
points.
        cv2.circle(bw_image, (np.float32(coords[1]),
np.float32(coords[0])),1,0,-1)

    # Upscaling of the result image (if specified)
    if img_scale is not None:
```

```
        height, width   = bw_image.shape
        newX,newY       = bw_image.shape[1]*img_scale,
bw_image.shape[0]*img_scale
        bw_image        = cv2.resize(bw_image,(int(newX),int(newY)))

    if transpose:
        bw_image = cv2.transpose(bw_image)

    lower_left_corner   = np.array([0, bw_image.shape[0]])
    first_label_coords  = tuple(lower_left_corner+np.array([5,-50]))
    second_label_coords = tuple(lower_left_corner+np.array([50,-10]))

    cv2.putText(bw_image, axe_names[0], first_label_coords,
cv2.FONT_HERSHEY_COMPLEX_SMALL ,1,0,2)
    cv2.putText(bw_image, axe_names[1], second_label_coords,
cv2.FONT_HERSHEY_COMPLEX_SMALL ,1,0,2)

    return bw_image

def display_pressure_2D_cv(images_list, time_str, margin=5,
wnd_title="Podglad symulacji FDTD"):
    if type(images_list) != list:
        images_list = [images_list]

    x_sizes = []
    y_sizes = []
    for image_arr in images_list:
        x_sizes.append(image_arr.shape[0])
        y_sizes.append(image_arr.shape[1])

    max_x = np.max(x_sizes)
    max_y = np.max(y_sizes)

    for i, image_arr in enumerate(images_list):
        eff_margin = margin
        if i == len(images_list)-1: margin = 0
        new_img =
np.zeros((max_x,images_list[i].shape[1]+margin)).astype(np.uint8)
        new_img[0:images_list[i].shape[0],0:images_list[i].shape[1]] =
images_list[i]
        images_list[i] = new_img

    compound_image = np.concatenate(images_list,axis=1)

    cv2.putText(compound_image, f'czas: {time_str}', (20,30),
cv2.FONT_HERSHEY_COMPLEX_SMALL   ,1,0,2)

    cv2.imshow(wnd_title,compound_image)

def sample_with_interpolation(measurement_points, p_plane):
    pressure_interpolator =
RectBivariateSpline(np.arange(p_plane.shape[0]),np.arange(p_plane.shape[1])
,p_plane,kx=5,ky=5)
    set_of_impres = []
    for coords in measurement_points:
        p_interpolated = pressure_interpolator(coords[0],coords[1])[0,0]
        set_of_impres.append(p_interpolated)
    return set_of_impres
```

```
# ------------------------------------------------------
# Simplified simulation interface with 2D preview
def run_fdtd(K, beta, source_pos_disc, excitation_signal, sim_length,
courant_number, X, T, measurement_points_xy=[], measurement_points_yz=[],
show_preview = True, write_video = False, img_scale=4,
ofname='last_simulation.mp4', PML_damping=0.15, PML_width=50,
GPU_device_id=0):

    # For the multi-GPU systems - we choose desired device for
    # calculations to be performed
    num_gpus = len(nb.cuda.gpus)

    if (GPU_device_id >= num_gpus) or (GPU_device_id < 0):
        raise RuntimeError(f'Bad GPU id was specified GPU ({GPU_device_id},
number of available GPUs: {num_gpus})')
    cuda.select_device(GPU_device_id)

    # Swapping axis in the xz plane
    measurement_points_yz_tmp = []
    for coord in measurement_points_yz:
        measurement_points_yz_tmp.append([coord[1],coord[0]])
    measurement_points_yz = measurement_points_yz_tmp

    # Creation of the simulation object
    sim_obj = FDTDSimulation(K, beta, source_pos_disc, excitation_signal,
courant_number, X, T, PML_damping=PML_damping, PML_width=PML_width)

    # Handling of the visualization saving on the hard drive
    if show_preview and write_video:
        video_frames   = []
        fourcc = cv2.VideoWriter_fourcc(*'mp4v')
        out    = cv2.VideoWriter(ofname,fourcc, 30.0,
(K.shape[0]*img_scale,K.shape[1]*img_scale),isColor=False)

    # A variable to which a value of measured
    # impulse response will be saved
    imp_res_set_xy = []
    imp_res_set_yz = []

    # The main FDTD simulation loop:
    for nn in range(1,sim_length):
        print('%3i/%3i\r'%(nn+1,sim_length), end='')

        # Calculation of simulation step, and obtaining
        # of the data from the GPU
        sim_obj.step()
        p_plane_xy = sim_obj.get_pressure_plane('xy')
        p_plane_yz = sim_obj.get_pressure_plane('xz')

        # Measurement and saving of the impulse responses


imp_res_set_xy.append(sample_with_interpolation(measurement_points_xy,
p_plane_xy))

imp_res_set_yz.append(sample_with_interpolation(measurement_points_yz,
p_plane_yz))

        if show_preview:
            bw_images = []
```

```
            bw_images.append(draw_pressure_2D_cv(K, p_plane_xy,
K[:,:,source_pos_disc[2]]==0, source_pos_disc[[0,1]],
source_pos_disc[[0,1]], measurement_points_xy, img_scale=img_scale,
axe_names=['x','y'], transpose=False))

            bw_images.append(draw_pressure_2D_cv(K, p_plane_yz,
K[source_pos_disc[0],:,:]==0, source_pos_disc[[1,2]],
source_pos_disc[[0,2]], measurement_points_yz, img_scale=img_scale,
axe_names=['z','y'], transpose=True))
            display_pressure_2D_cv(bw_images, '%2.2f ms'%((nn*1000)*T))

            # Saving video visualization and display of the image
            if write_video:
                out.write(bw_image)

            # If visualization is turned on, the simulation
            # can be terminated by pressing ESC.
            key = cv2.waitKey(1)
            if key == 27:
                break

    # Closing file with the simulation data dump
    if show_preview:
        # Closing the preview window
        cv2.destroyAllWindows()
        if write_video:
            out.release()

    imp_res_set_xy = np.array(imp_res_set_xy).T
    imp_res_set_yz = np.array(imp_res_set_yz).T

    return [imp_res_set_xy,imp_res_set_yz]

# ------------------------------------------------------
# Simplified simulation interface with 3D preview
def run_fdtd_render3D(K, beta, source_pos_disc, excitation_signal,
sim_length, courant_number, X, T, vis_halfspread=20, display_time = True,
frame_skip=None, ofname='render_3d.avi',
encode_video=False,PML_damping=25000):

    if ofname is not None:
        # reation of the visualizatio window
        output_folder = '_output_renders'
        if not os.path.isdir(output_folder):
            os.mkdir(output_folder)

    # Creation of the new K matrix, which will allow
    # display of scattering objects placed in the computational domain
    new_K = np.ones_like(K)
    new_K[2:-2,2:-2,2:-2] = K[2:-2,2:-2,2:-2]
    new_K[new_K==0] = 8
    new_K[new_K==6] = 0
    new_K[new_K==5] = 0

    # Creation of the simulation object
    sim_obj = FDTDSimulation(K, beta, source_pos_disc, excitation_signal,
courant_number, X, T, PML_damping=PML_damping)

    if ofname is not None:
        # Creation of the visualizatio window
        fig = mlab.figure(size=(1000,1000))
```

```
    # The main FDTD simulation loop:
    for frame_number in range(1,sim_length):
        try:
            print('%3i/%3i\r'%(frame_number+1,sim_length), end='')

            # Calculation of simulation step, and obtaining
            # of the data from the GPU
            sim_obj.step()

            if frame_skip is not None:
                if (not (frame_number%frame_skip==0)) and (frame_number !=
0):
                    continue

            p_distribution  = sim_obj.get_pressure_3D()

            if ofname is None:
                fig = mlab.figure(size=(1000,1000))

            # Display of the scalar field representing the "environment",
            # which are mainly the scaterring objects
            environ_field   = mlab.pipeline.scalar_field(new_K)
            env_plot        = mlab.pipeline.volume(environ_field)

            # Display of the pressure wave visualization
            pressure_wave   = mlab.pipeline.scalar_field(p_distribution,
colormap='blue-red')
            vol_plot        = mlab.pipeline.volume(pressure_wave,
vmin=0.01, vmax=np.max(p_distribution))

            # Value bar
            lo_bound    = np.min([np.min(p_distribution),-vis_halfspread])
            hi_bound    = np.max([np.max(p_distribution),vis_halfspread])
            mlab.colorbar(orientation='vertical').data_range = (lo_bound,
hi_bound)

            if display_time:
                mlab.title(f"time: {'%.2f'%(1000*frame_number*T)}
[ms]",height=0.01,size=0.3)

            if ofname is not None:
                # Saving of the output visualization frame
                if frame_skip is not None:
                    effective_frame_number = frame_number//frame_skip
                else:
                    effective_frame_number = frame_number

                save_path =
os.path.join(output_folder,f'{str(effective_frame_number).zfill(4)}.png')
                mlab.savefig(filename=save_path)
                # Clearing the frame
                mlab.clf()
            else:
                mlab.show()

        except KeyboardInterrupt as e:
            print('Rendering loop was terminated manually.\n')
            break

    print()
```

```
    if ofname is not None:
        input('\n-------------------------\nIn a moment, frames of
animation will be encoded into a film\nCheck the animation frames in a
folder, and press ENTER to continue \n')

        # Conversion of generated frames to a film
        if frame_skip is not None:
            fps = int(60/frame_skip)
        else:
            fps = 60

        # Encoding of the animation to a film in an .avi format
        cmd = f"ffmpeg  -r {fps} -i _output_renders/%04d.png  -c:v libx264
-vf \"crop=trunc(iw/2)*2:trunc(ih/2)*2,fps={fps},format=yuv420p\" {ofname}"
        if encode_video:
            print(cmd)
            os.system(cmd)
        else:
            print(f'If you want to encode video to a final form, use the
following command: \n\n{cmd}')
```

APPENDIX C: PROGRESS OF DPG AGENTS WITH RANDOM INPUT

This Appendix contains visualizations of the obtained autocorrelation diffusion coefficient and rewards of agents with a random starting diffuser design was fed into. In such circumstances, an agent has a harder task to perform, as it has to be prepared to tackle very varied diffuser designs, which also are characterized by varying diffusion coefficient. This fact is visible in the form of low scores gained by agents and lack of or poor improvement of generated diffuser designs over time.

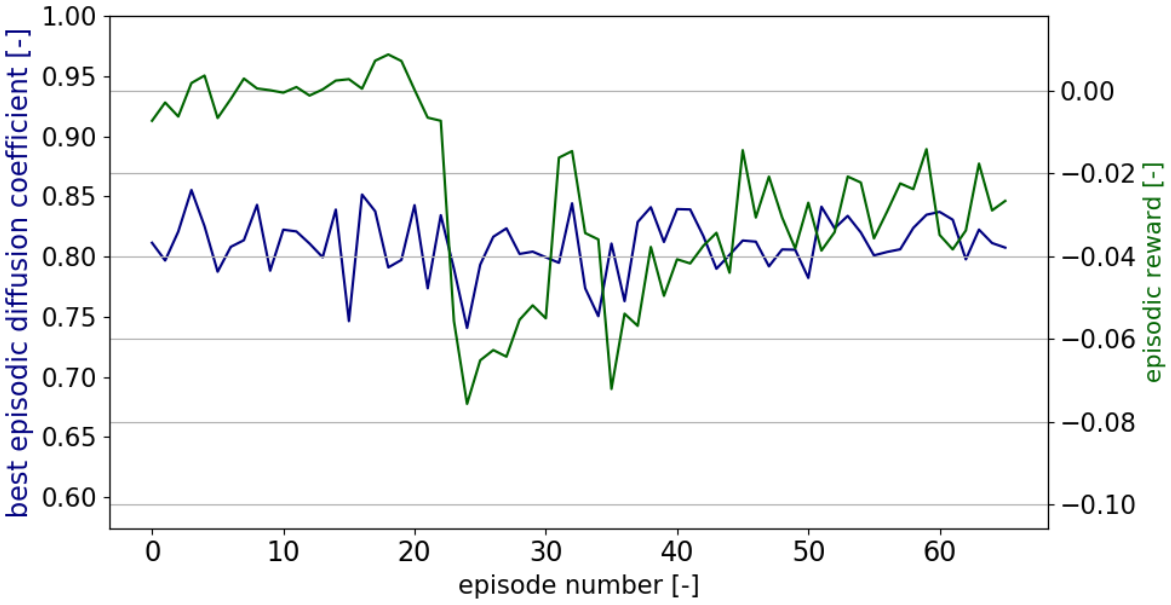


Fig. C.1 Results obtained from agent no. 1, which was fed with random input diffuser designs to be improved over each episode.

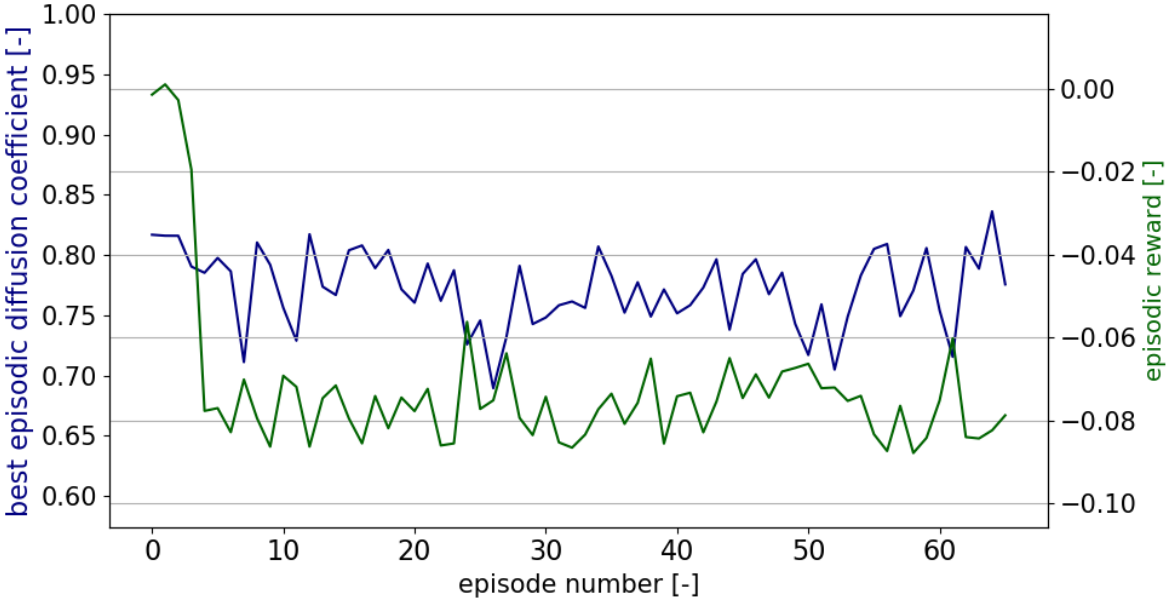


Fig. C.2 Results obtained from agent no. 2, which was fed asrandom input diffuser

designs to be improved over each episode.

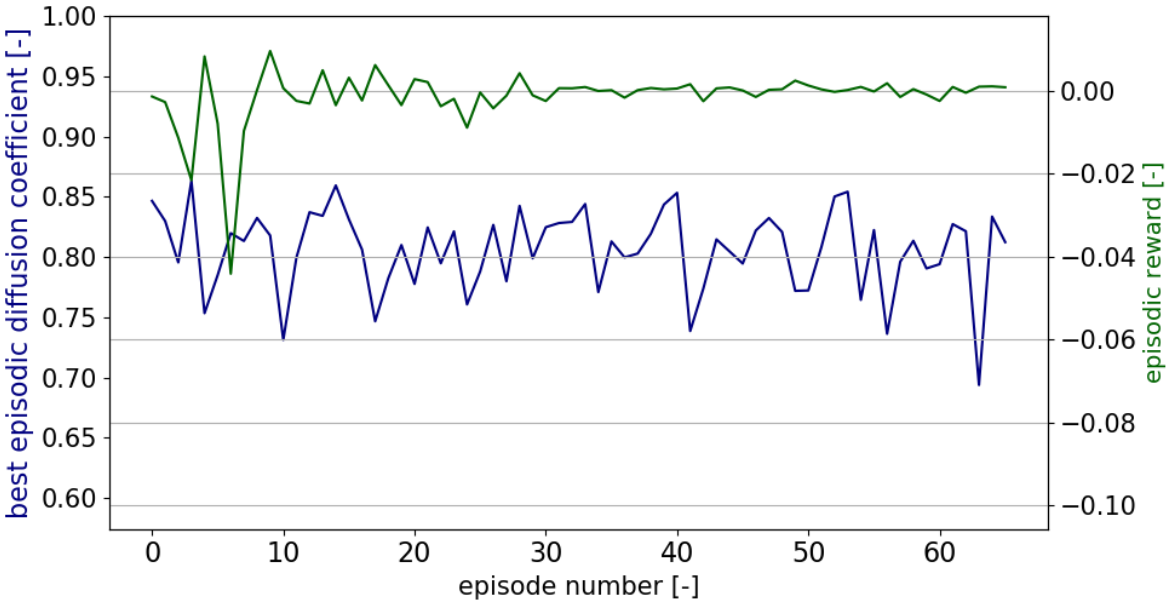


Fig. C.3 Results obtained from agent no. 3, which was fed with random input diffuser designs to be improved over each episode.

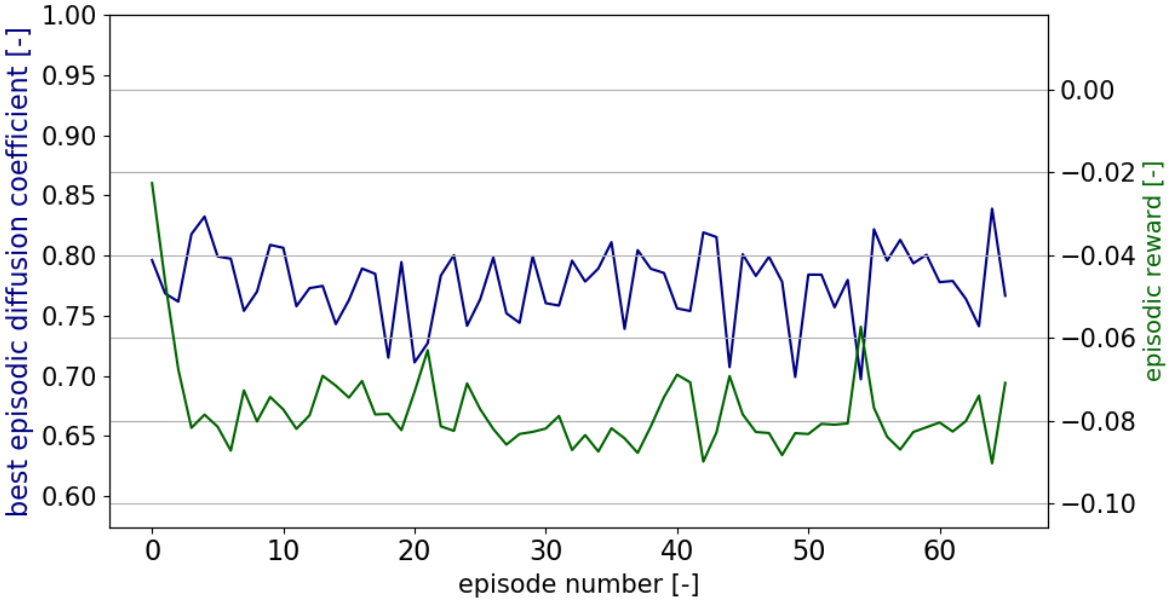


Fig. C.4 Results obtained from agent no. 4, which was fed with random input diffuser designs to be improved over each episode.

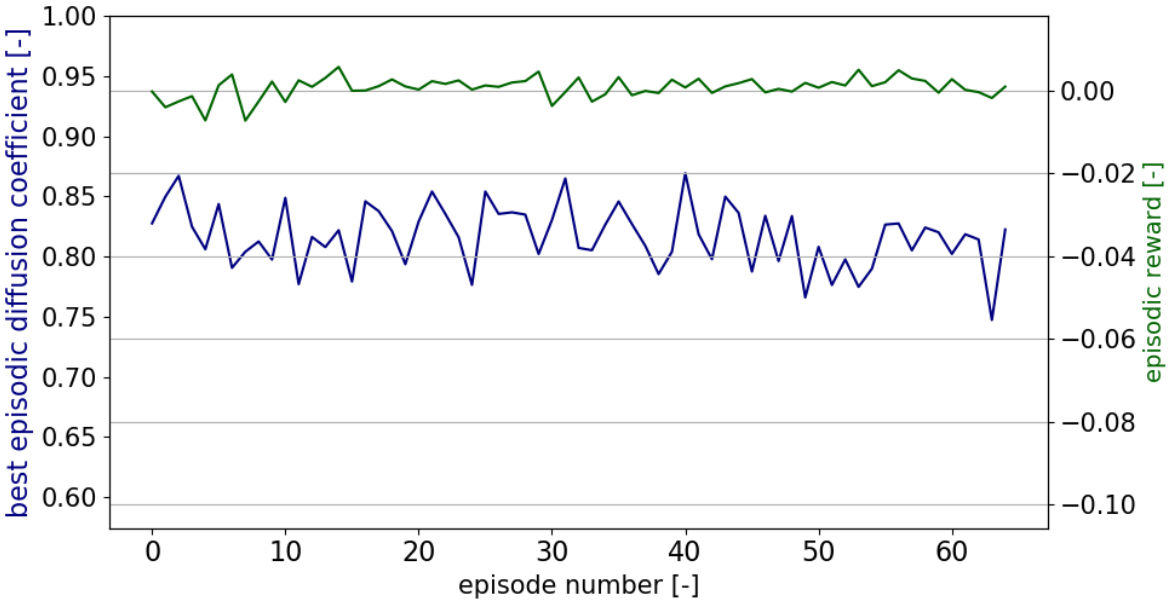


Fig. C.5 Results obtained from agent no. 5, which was fed with random input diffuser designs to be improved over each episode.

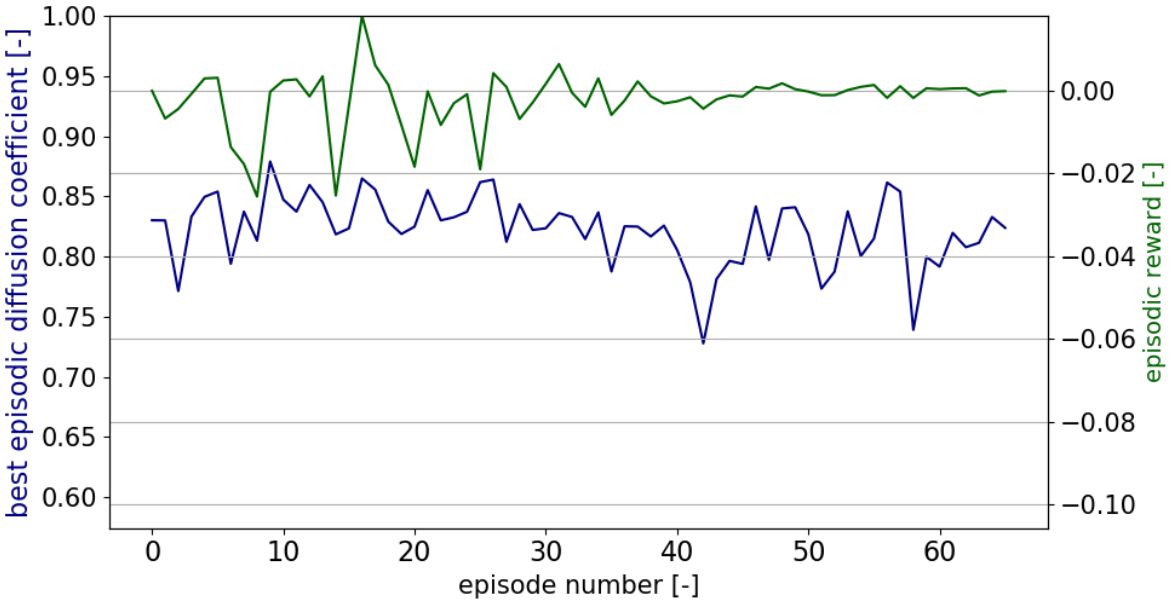


Fig. C.6 Results obtained from agent no. 6, which was fed with random input diffuser designs to be improved over each episode.

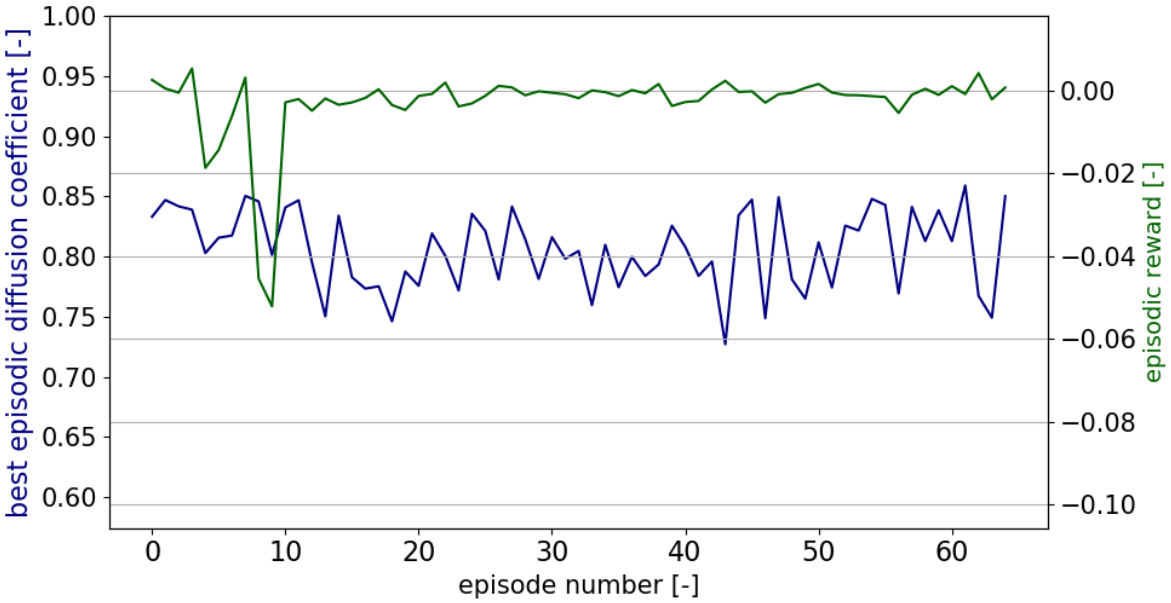


Fig. C.7 Results obtained from agent no. 7, which was fed with random input diffuser designs to be improved over each episode.

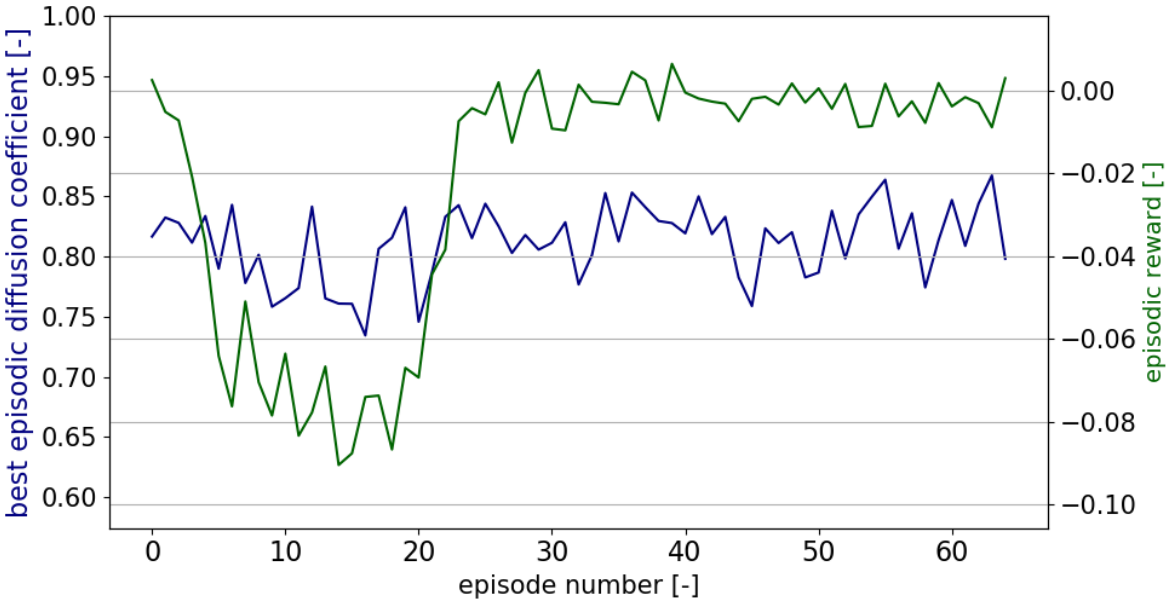


Fig. C.8 Results obtained from agent no. 8, which was fed with random input diffuser designs to be improved over each episode.

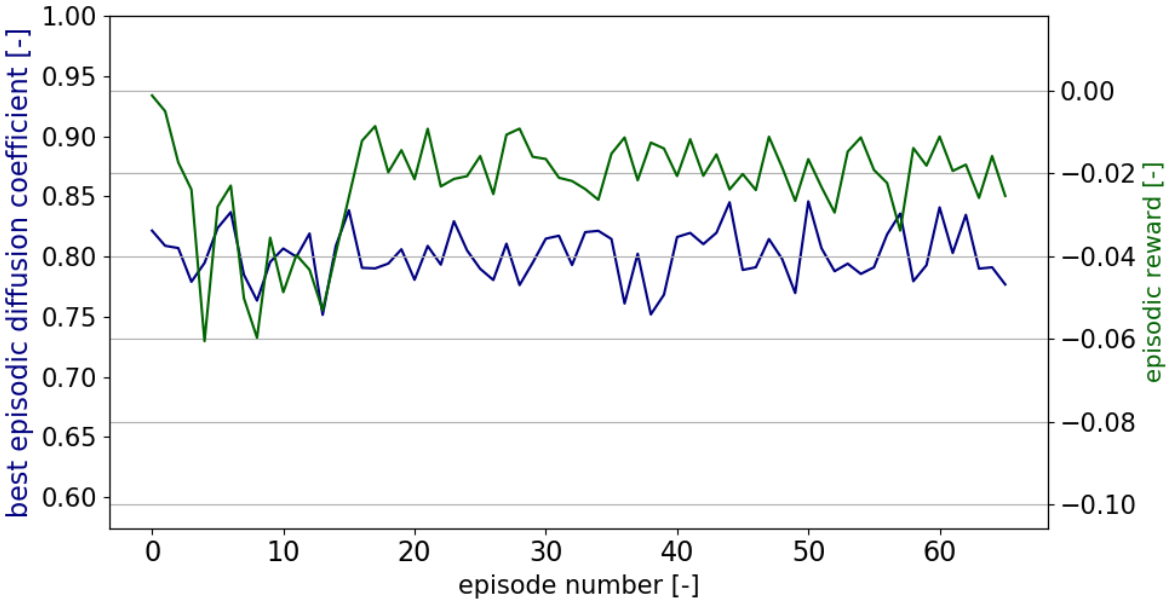


Fig. C.9 Results obtained from agent no. 9, which was fed with random input diffuser designs to be improved over each episode.

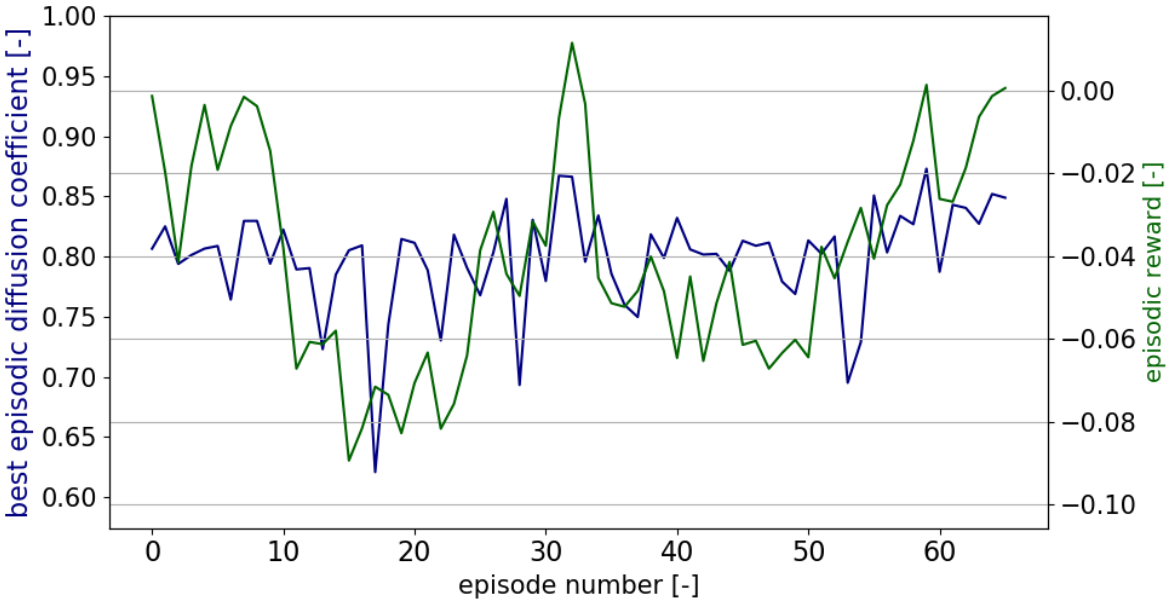


Fig. C.10 Results obtained from agent no. 10, which was fed with random input diffuser designs to be improved over each episode.

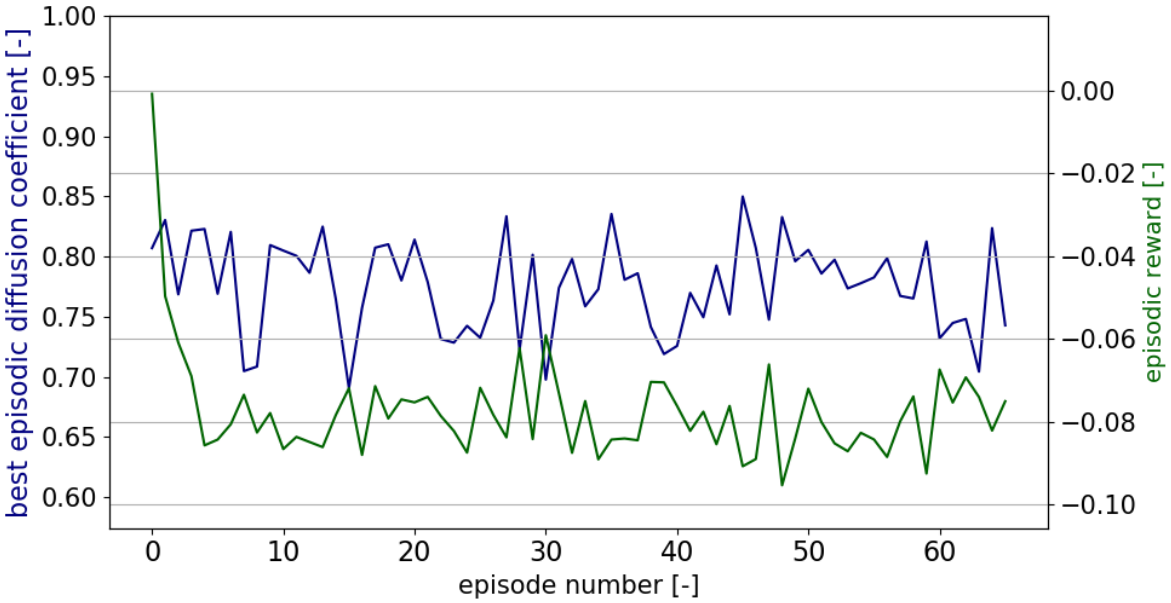


Fig. C.11 Results obtained from agent no. 11, which was fed with random input diffuser designs to be improved over each episode.

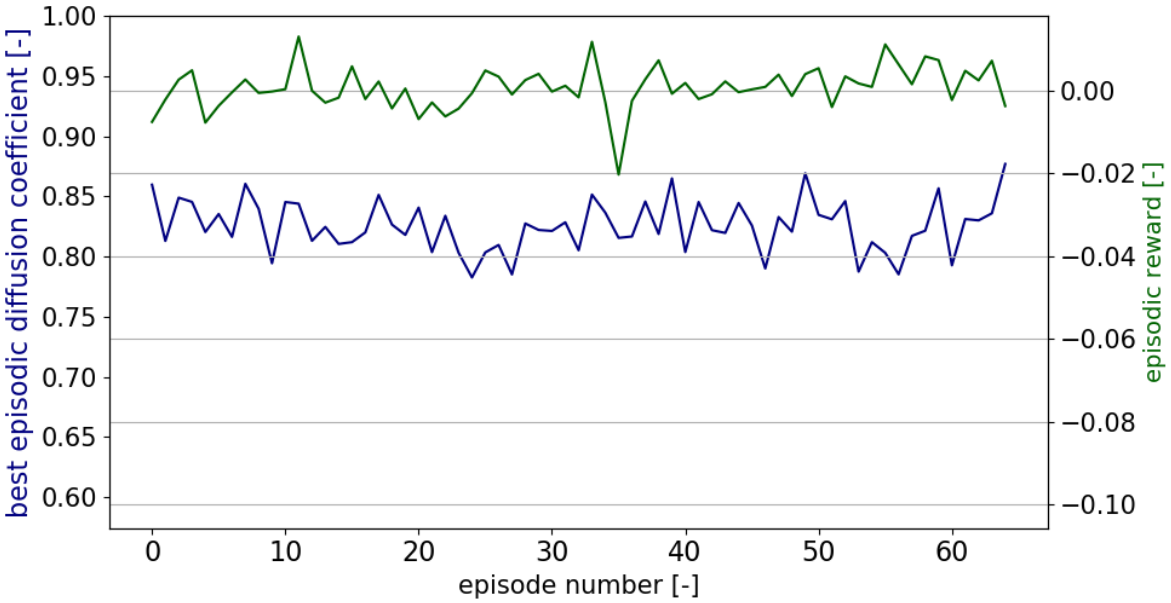


Fig. C.12 Results obtained from agent no. 12, which was fed with random input diffuser designs to be improved over each episode.

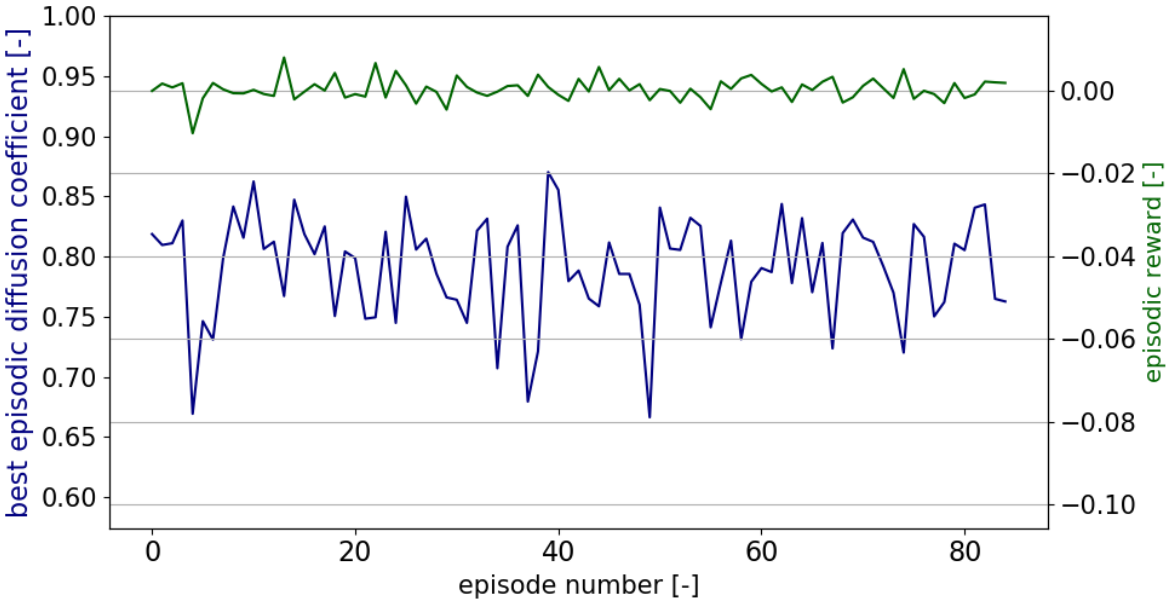


Fig. C.13 Results obtained from agent no. 13, which was fed with random input diffuser designs to be improved over each episode.

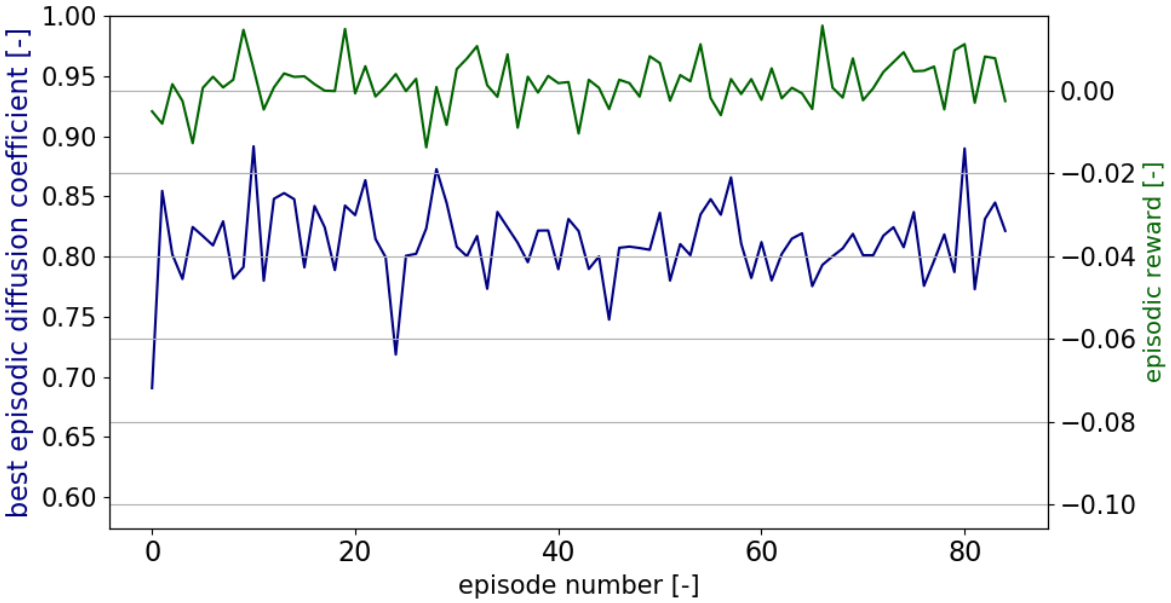


Fig. C.14 Results obtained from agent no. 14, which was fed with random input diffuser designs to be improved over each episode.

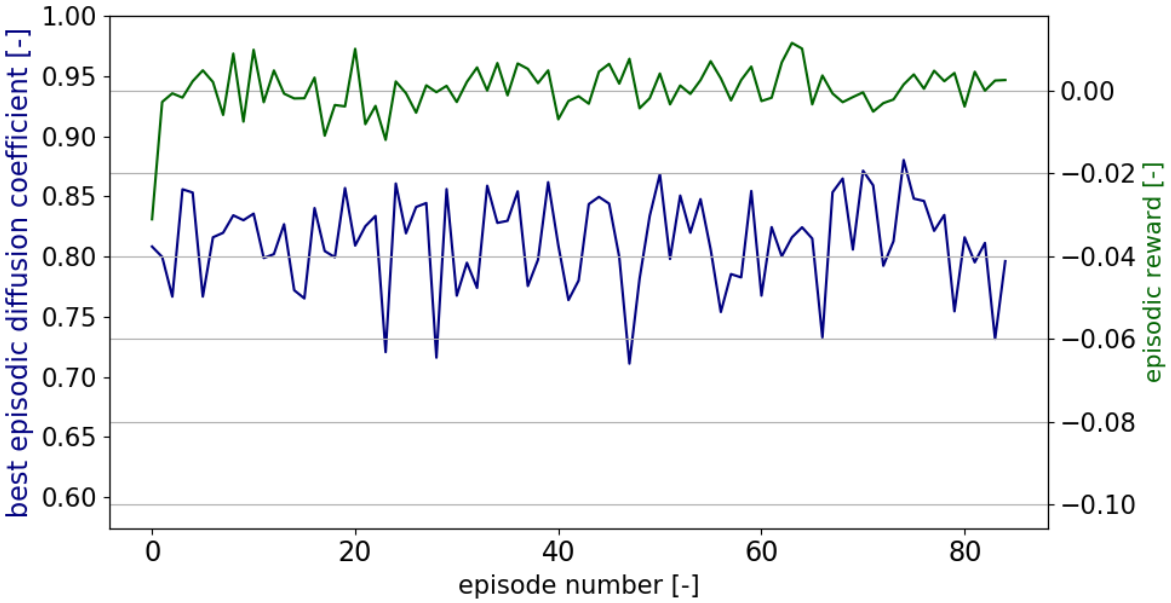


Fig. C.15 Results obtained from agent no. 15, which was fed with random input diffuser designs to be improved over each episode.

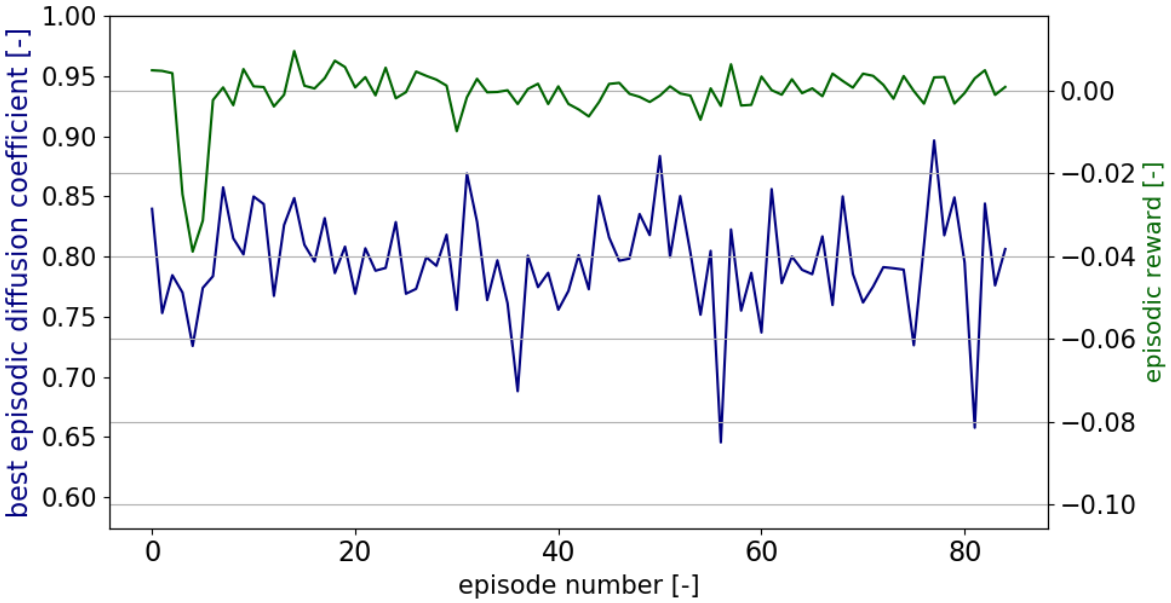


Fig. C.16 Results obtained from agent no. 16, which was fed with random input diffuser designs to be improved over each episode.

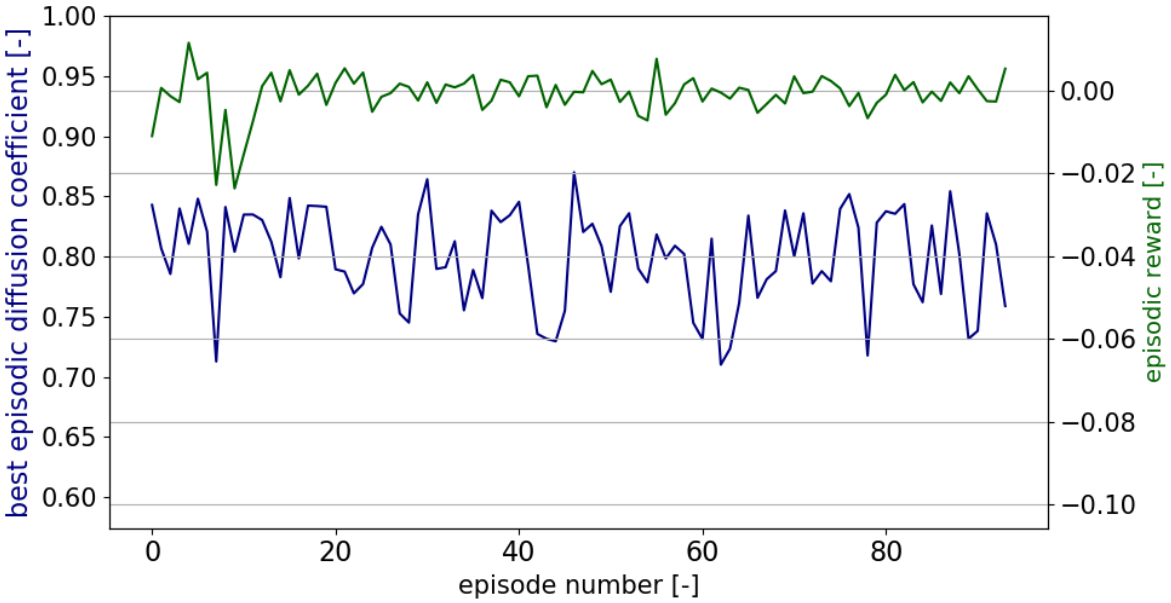


Fig. C.17 Results obtained from agent no. 17, which was fed with random input diffuser designs to be improved over each episode.

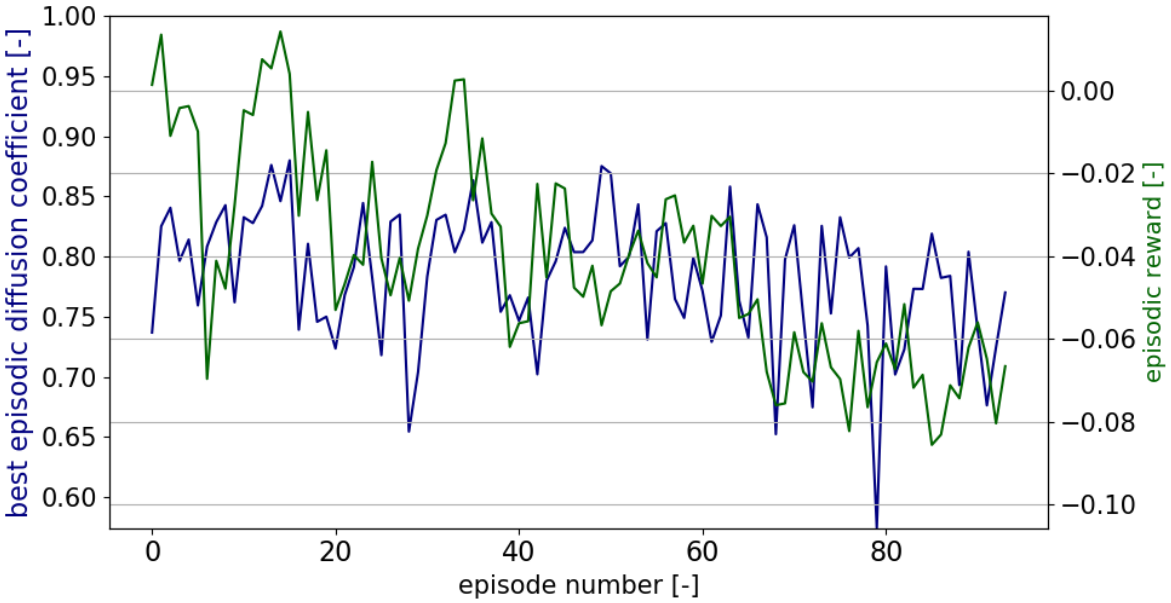


Fig. C.18 Results obtained from agent no.18, which was fed with random input diffuser designs to be improved over each episode.

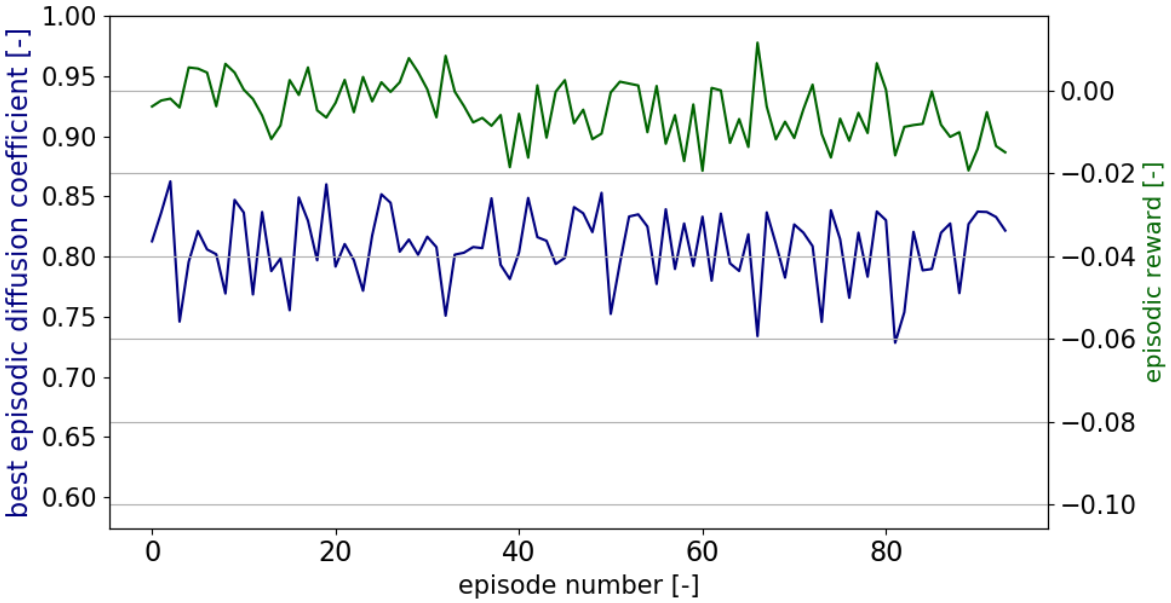


Fig. C.19 Results obtained from agent no. 19, which was fed with random input diffuser designs to be improved over each episode.

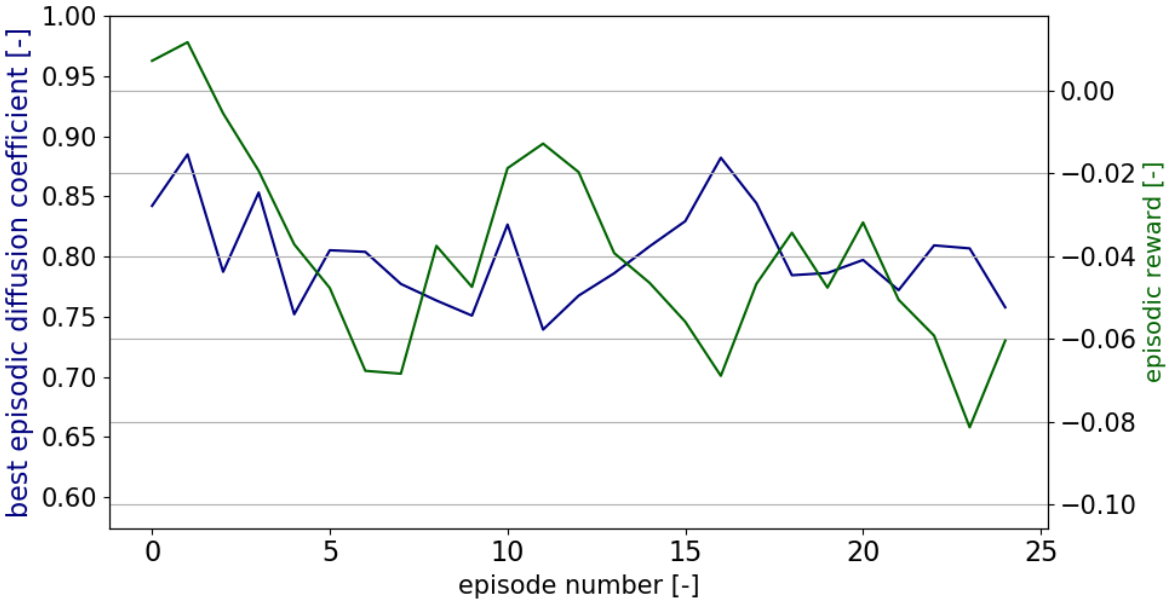


Fig. C.20 Results obtained from agent no. 20, which was fed with random input diffuser designs to be improved over each episode.

APPENDIX D: PROGRESS OF THE DPG AGENTS WITH INPUT SELECTED FROM BEST DESIGNS

This Appendix contains visualizations of the obtained autocorrelation diffusion coefficient and rewards of agents to whom the best design chosen from 10 best designs encountered by a group of agents over the course of training was selected as a starting design to be improved. It can be seen that for some agents, it was possible to get better over the course of training; however, it was not true for all agents. This suggests that successful training of the agent is dependent on the starting weights of an agent.

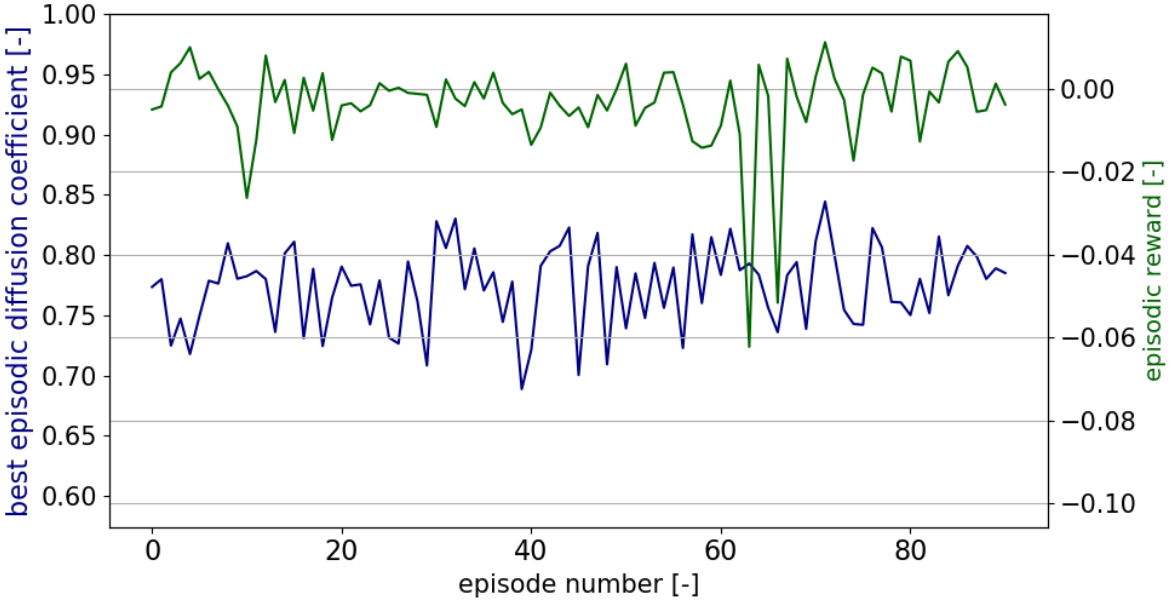


Fig. D.1. Results obtained from agent no. 1, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

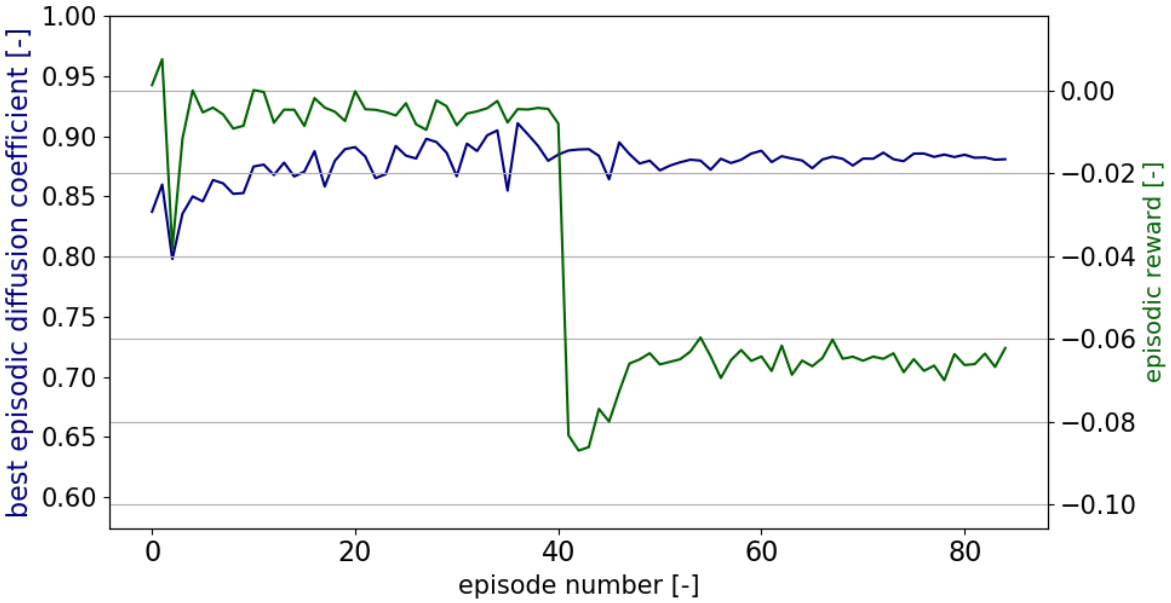


Fig. D.2. Results obtained from agent no. 2, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

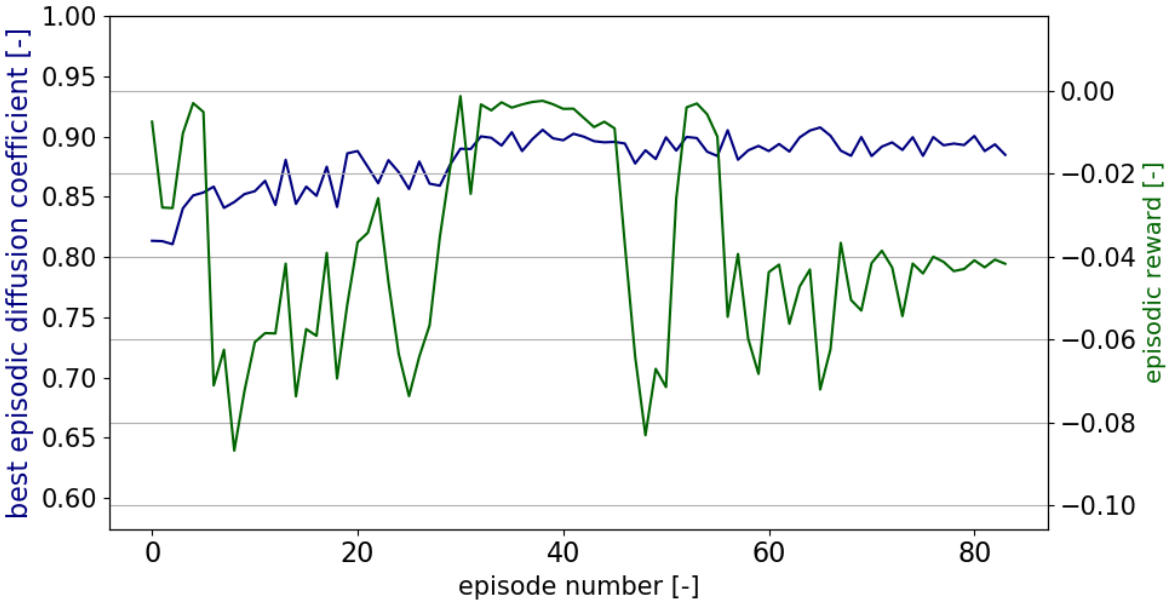


Fig. D.3. Results obtained from agent no. 3, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

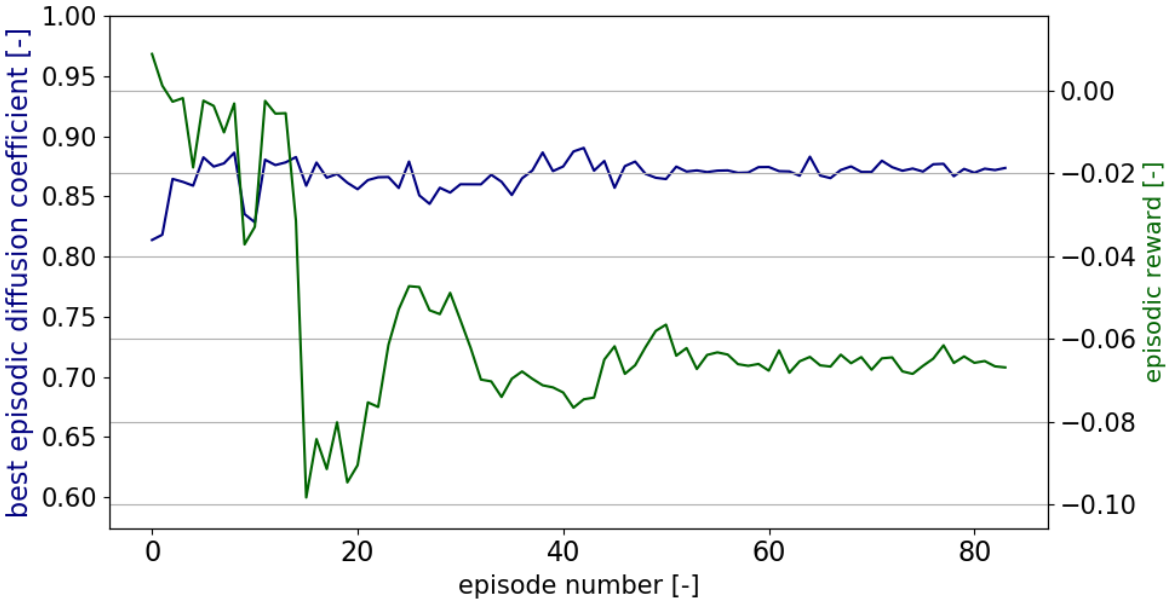


Fig. D.4. Results obtained from agent no. 4, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

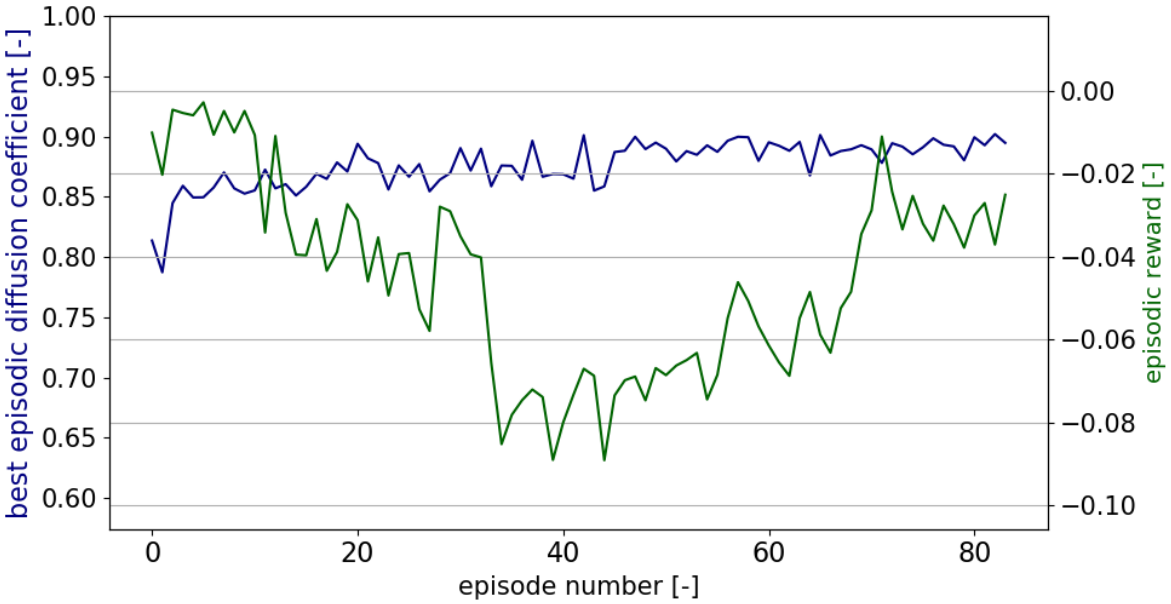


Fig. D.5. Results obtained from agent no. 5, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

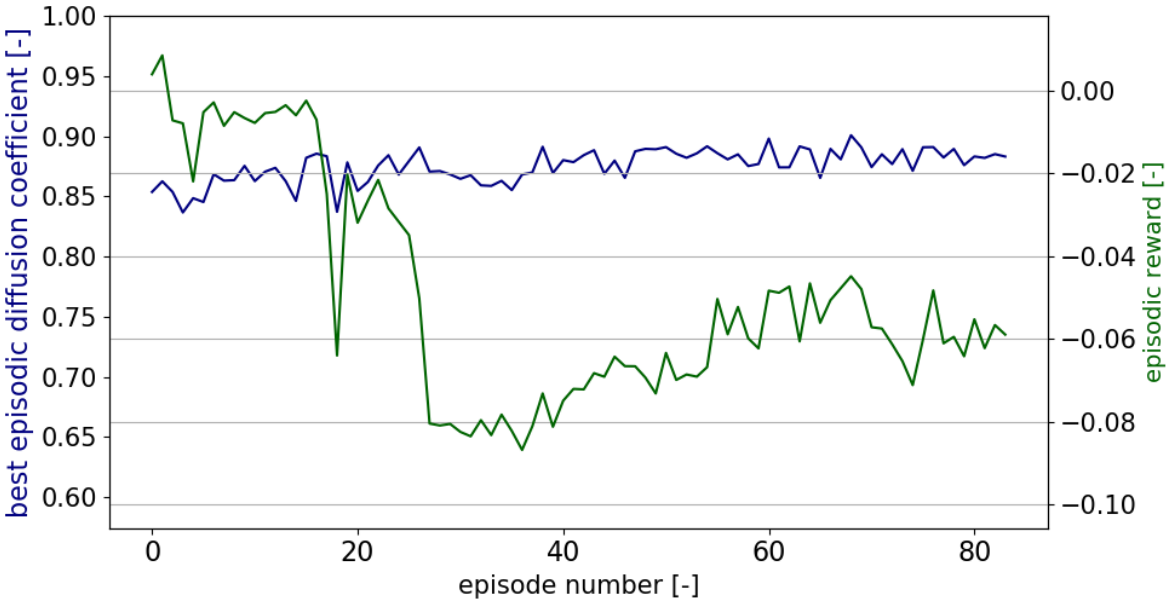


Fig. D.6. Results obtained from agent no. 6, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

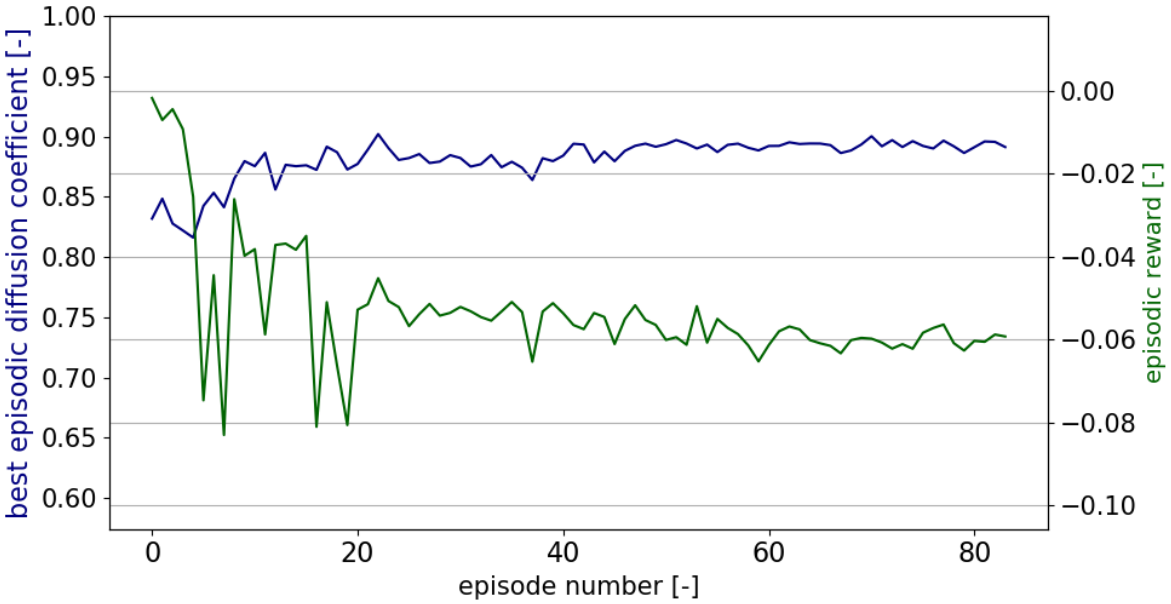


Fig. D.7. Results obtained from agent no. 7, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

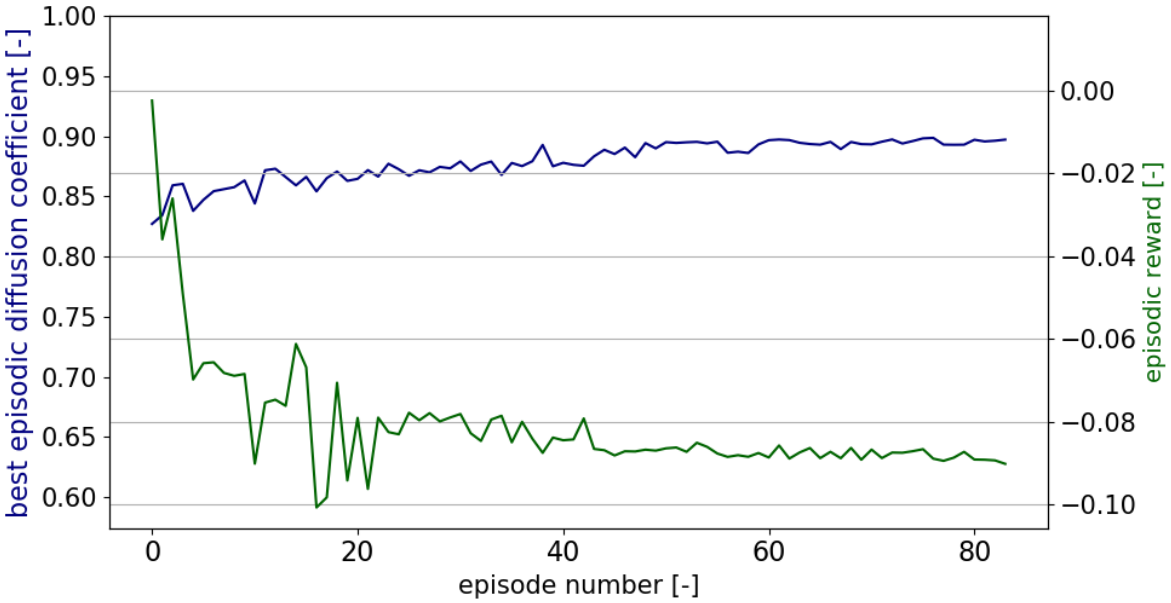


Fig. D.8. Results obtained from agent no. 8, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

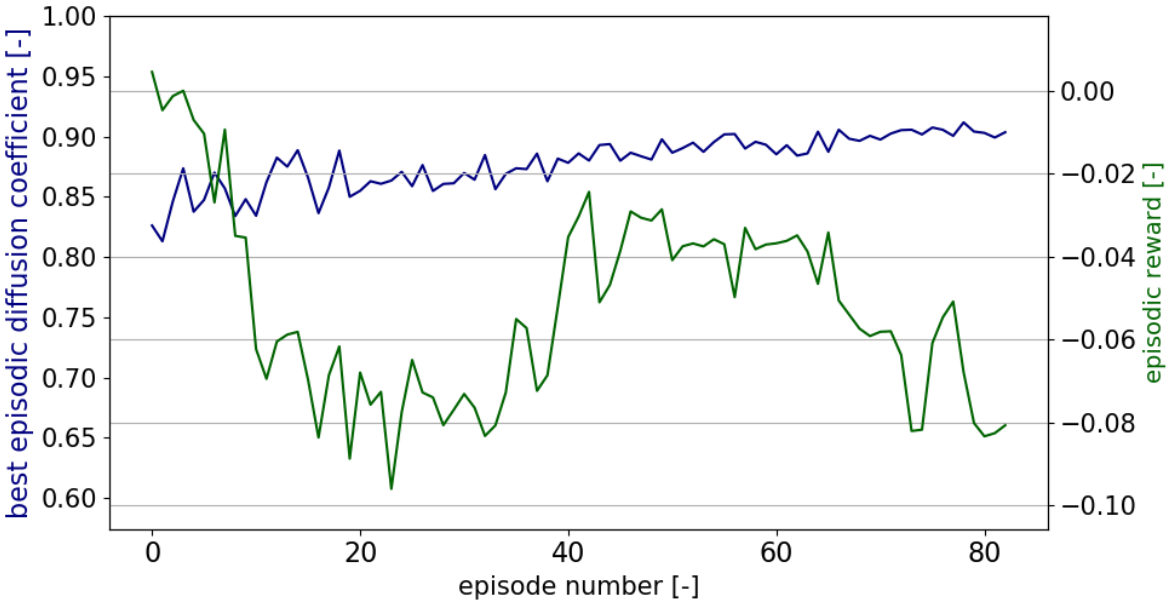


Fig. D.9. Results obtained from agent no. 9, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

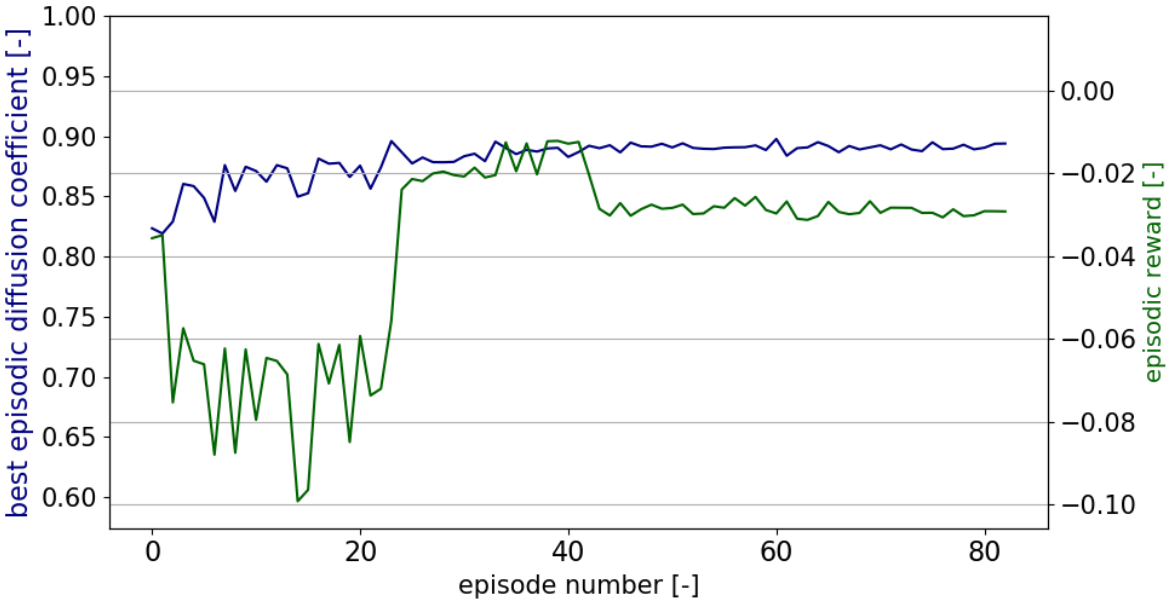


Fig. D.10. Results obtained from agent no. 10, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

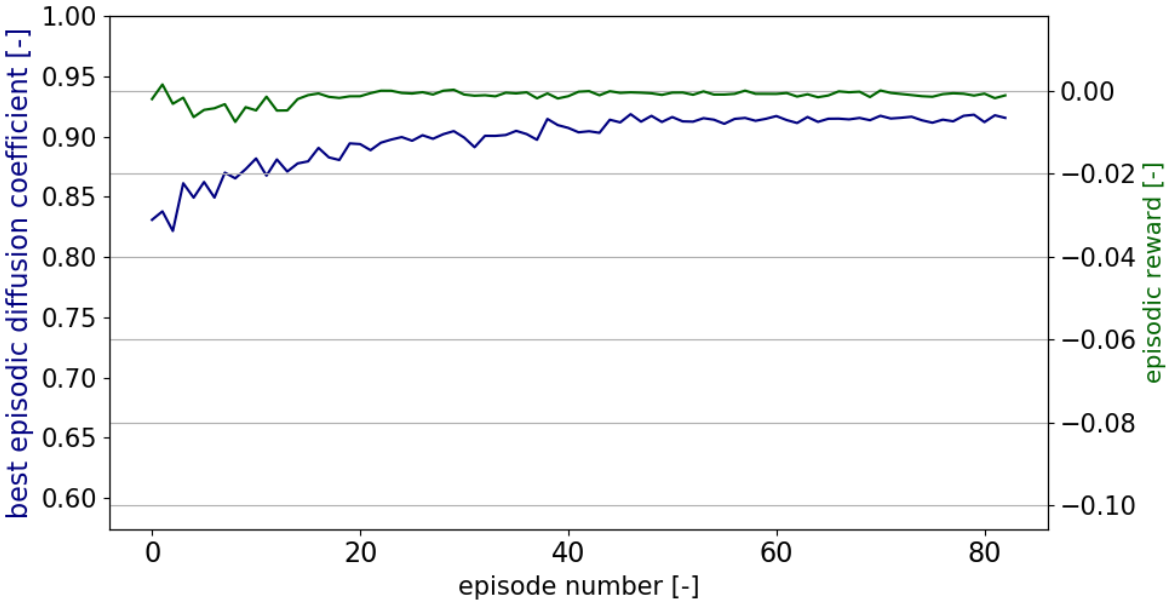


Fig. D.11. Results obtained from agent no. 11, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

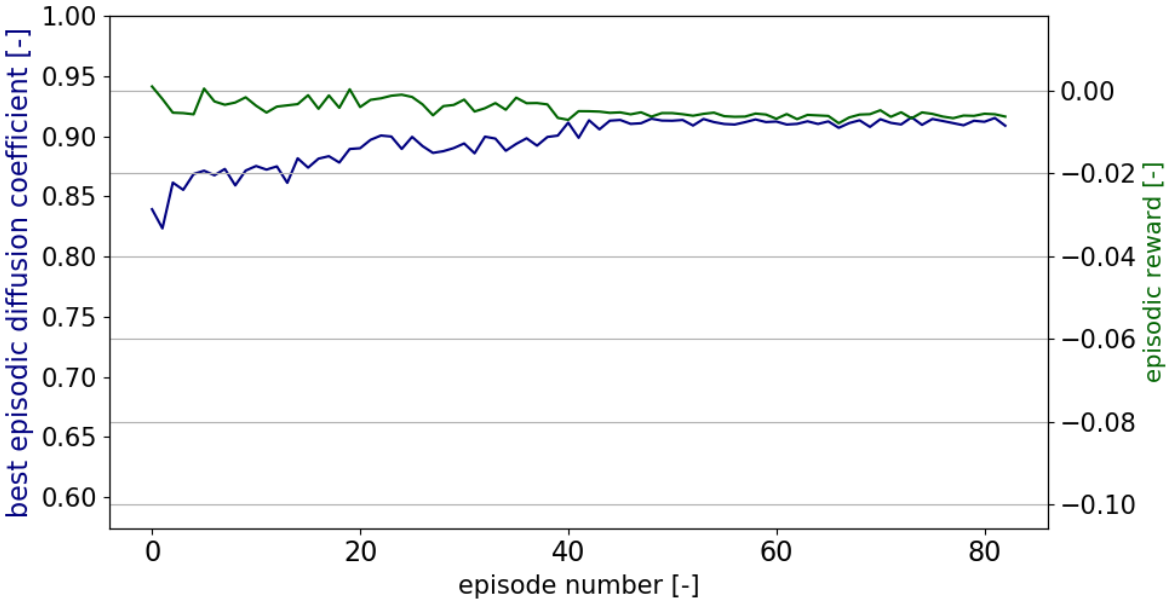


Fig. D.12. Results obtained from agent no. 12, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

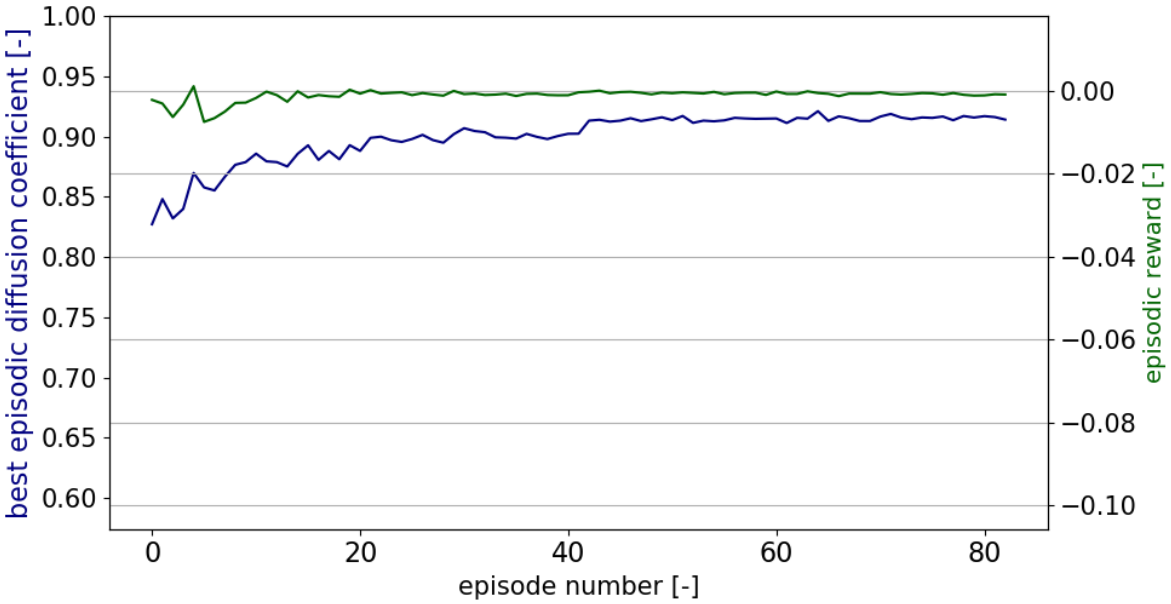


Fig. D.13. Results obtained from agent no. 13, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

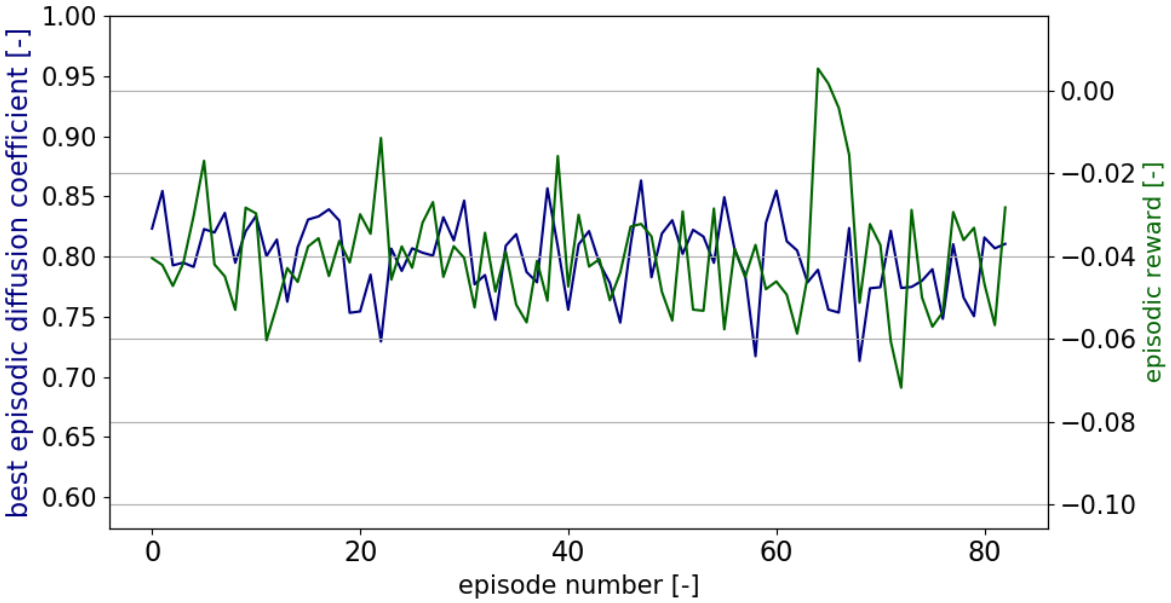


Fig. D.14. Results obtained from agent no. 14, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

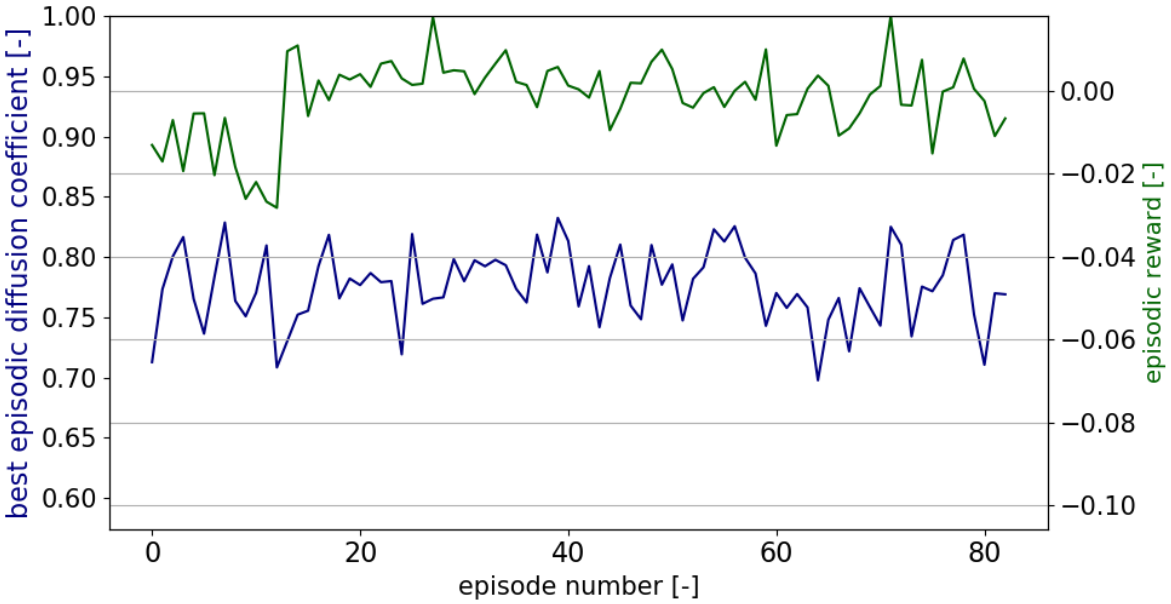


Fig. D.15. Results obtained from agent no. 15, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

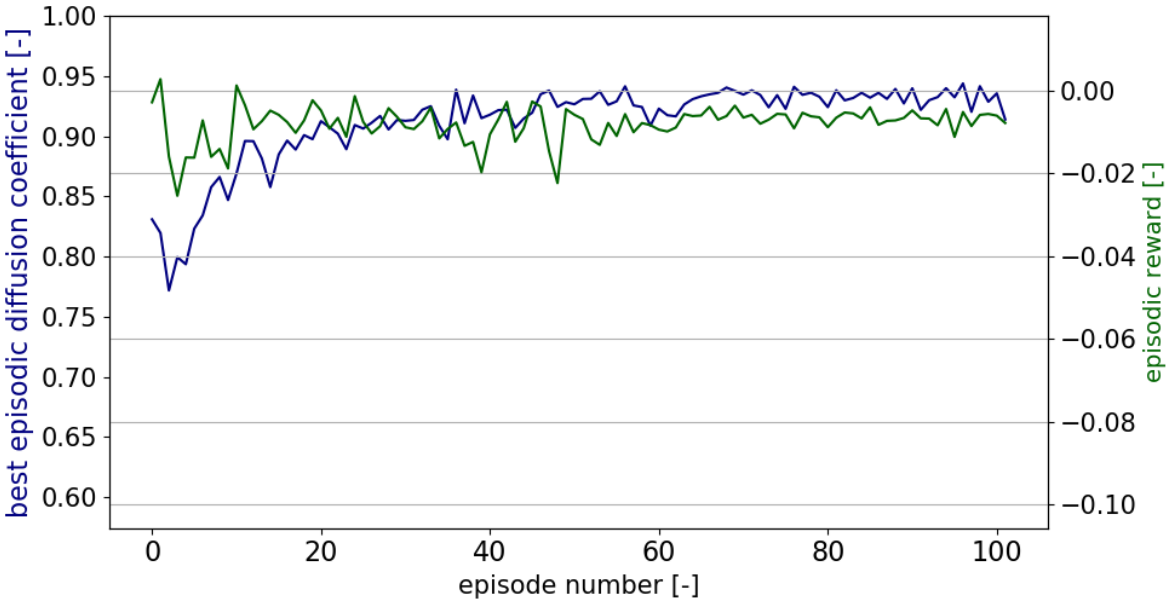


Fig. D.16. Results obtained from agent no. 16, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

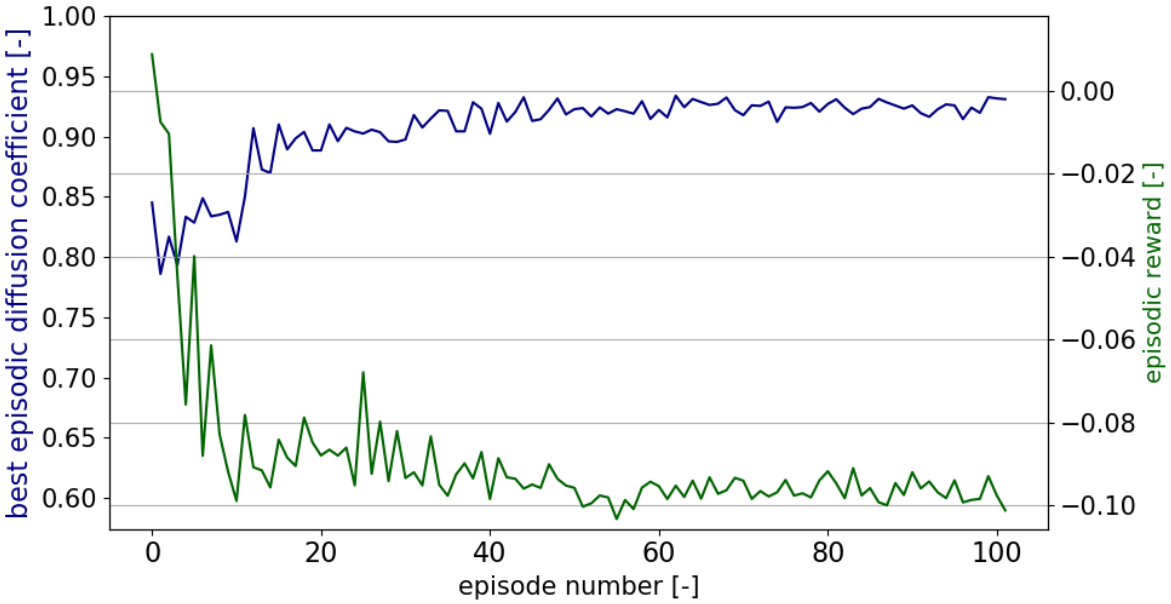


Fig. D.17. Results obtained from agent no. 17, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

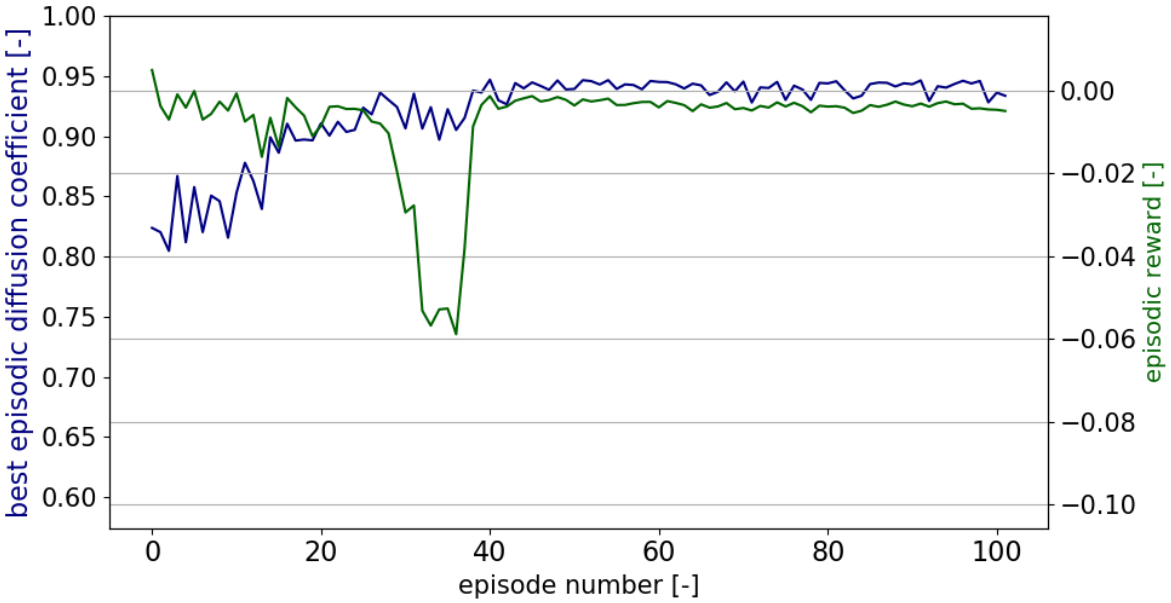


Fig. D.18. Results obtained from agent no. 18, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

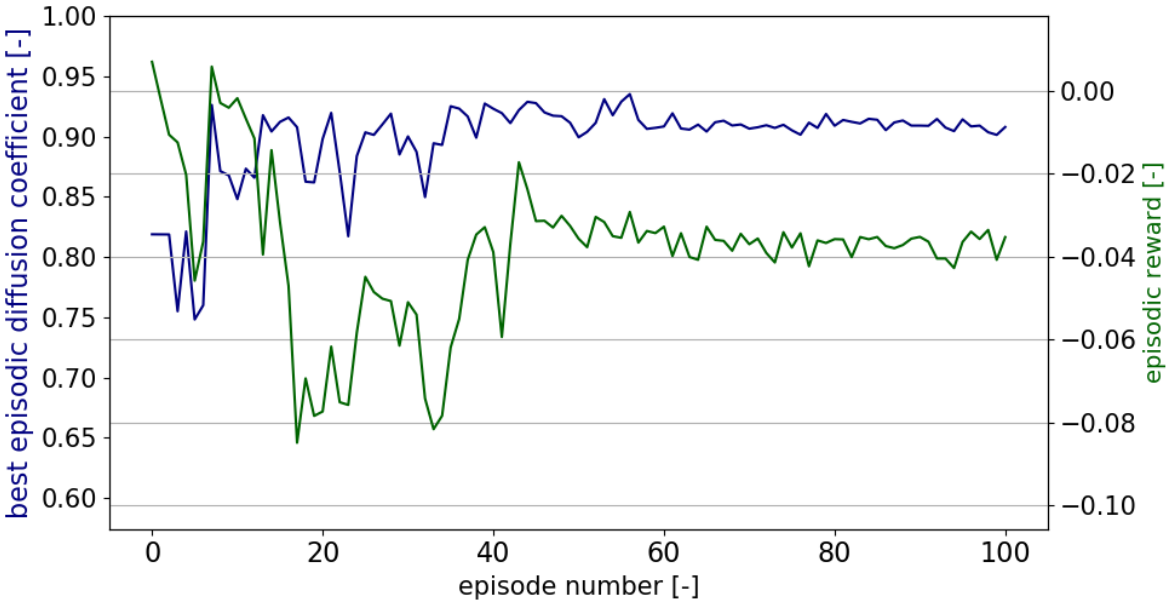


Fig. D.19. Results obtained from agent no. 19, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

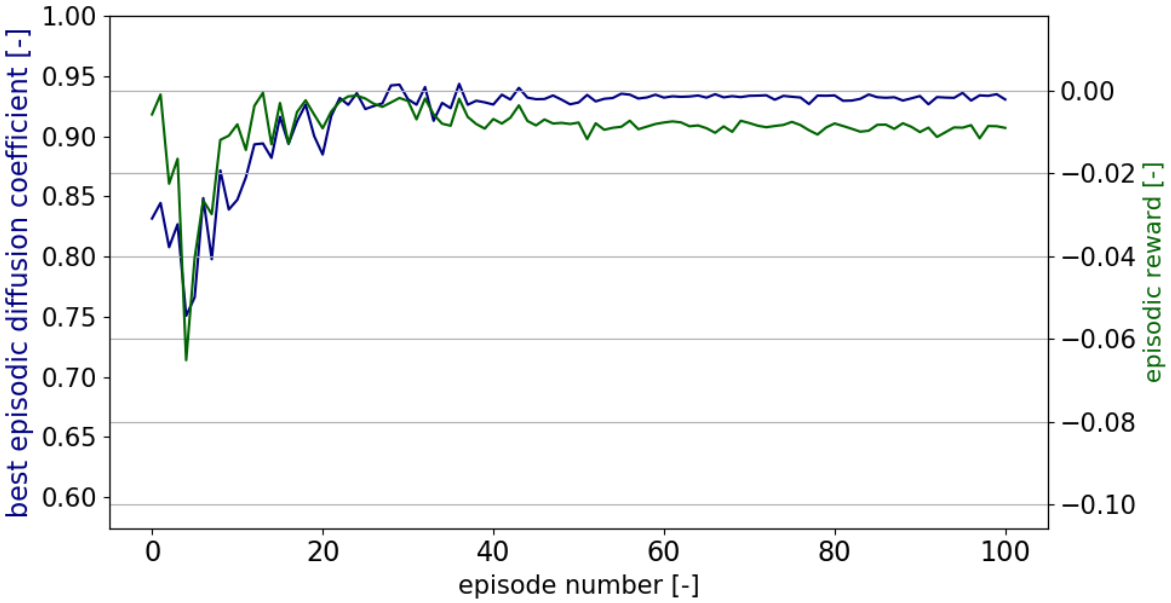


Fig. D.20. Results obtained from agent no. 20, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

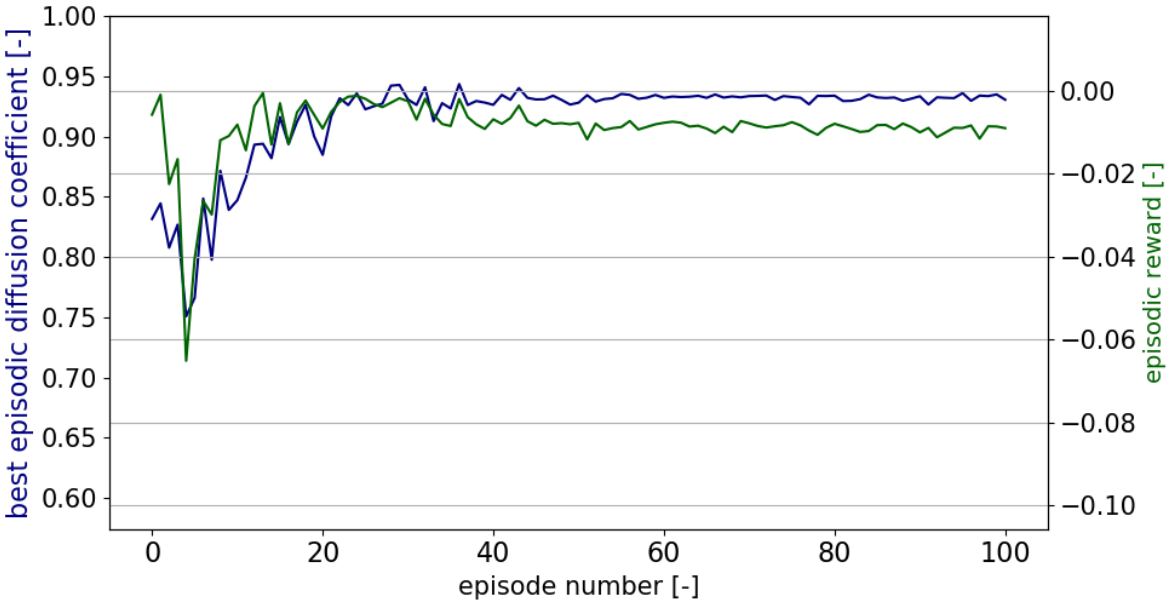


Fig. D.21. Results obtained from agent no. 21, for which a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process was selected as a starting design.

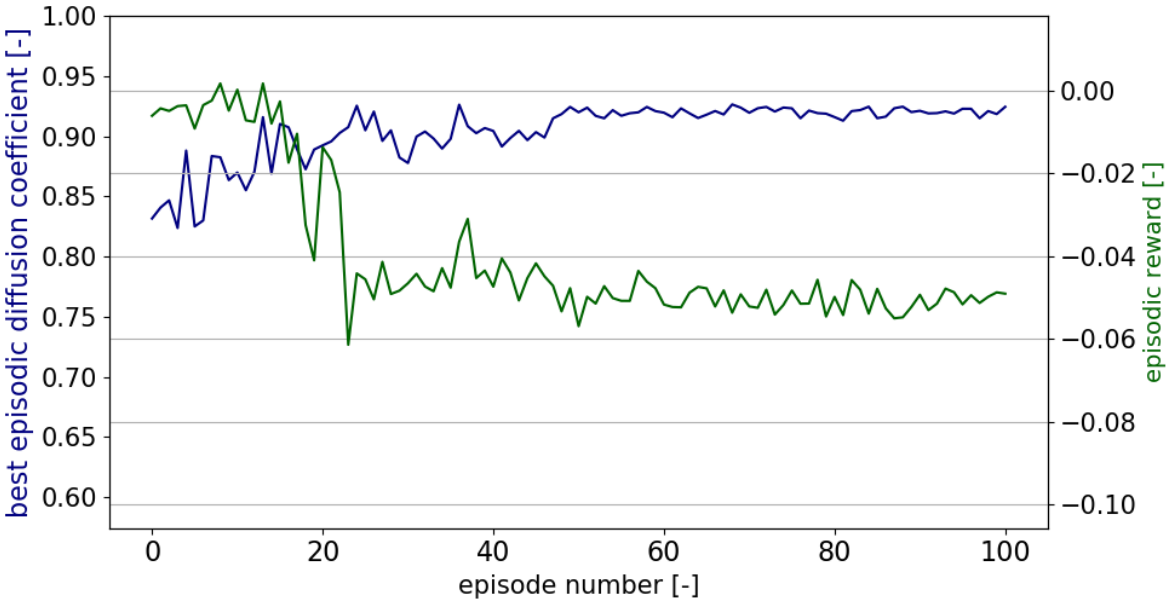


Fig. D.22. Results obtained from agent no. 22, which was fed a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process.

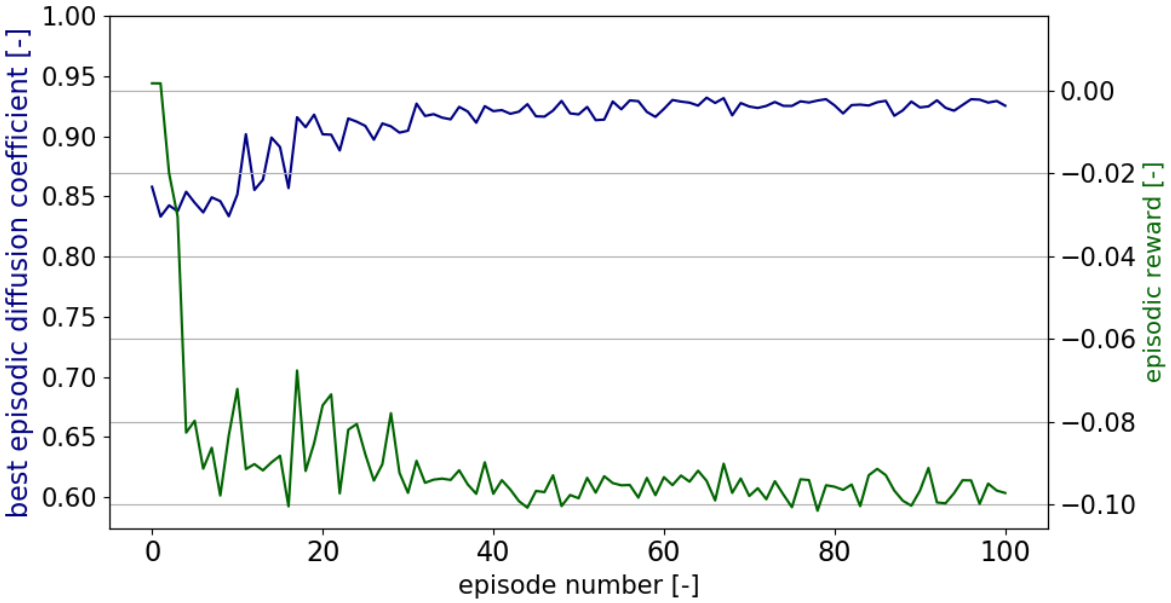


Fig. D.23. Results obtained from agent no. 23, which was fed a diffuser chosen from 10 best designed encountered by a group of agents in the optimization process.