Proceedings of the 1st international Symposium on AI and Fluid Mechanics
Paper No S1 P5

Position Control of Acoustic Cavitation Bubbles by Reinforcement Learning Using GPU-Accelerated Environment Simulations

[1]K. Klapcsik*

[1]*Department of Hydrodynamic Systems, Faculty of Mechanical Engineering, Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary*

Abstract. Reinforcement Learning (RL) is used to develop control methods that allow precise positioning of acoustic cavitation bubbles by acoustic manipulation. A problem-specific RL is developed which utilizes the capabilities of GPUs to vectorize the environmental simulations. Via vectorization, the framework can manage hundreds or even thousands of environmental replicas for single and dual bubble control problems. The performance is demonstrated on two benchmark problems. The implemented RL algorithm is the proximal policy optimization (PPO) method. The results showed that the agent consistently learns effective policy across a wide range of parallel environments. The learned policy effectively exposes the nonlinear nature of translation bubble dynamics. Although the benchmark problems are simple, the framework allows to easily implement and refine problems for different scenarios with small number of bubbles; for example, influence the radial dynamics to shift the reaction kinetics in the direction of the formation of the required chemical product or delay the bubble break-up due to shape instability.

Keywords: Reinforcement Learning; GPU-acceleration; vectorized environments, acoustic cavitation

1. Introduction

In liquids, exposed to high-frequency and high-intensity ultrasound, microbubbles are generated, which exhibit radial oscillations. Over the last few decades, numerous promising applications have emerged that exploit the physical and chemical effects of collapsing cavitation bubbles. The bubbles act as microreactors where chemical reactions occur, producing highly reactive free radicals that can be utilized to enhance chemical reactions [1]. Ultrasonic irradiation can also be utilized in biomedical applications to enhance drug delivery by breaking down biological barriers between tumors or central nervous system [2]. The efficacy of applications depends on the dynamics of cavitation bubbles. However, controlling the cavitation process is challenging due to the complex physics. Depending on the application, various tasks need to be solved. For example, in sonochemistry, it may be necessary to: create the optimal bubble distribution (cluster structure), influence the radial dynamics to shift the reaction kinetics in the direction of the formation of the required chemical product [1], or delay the bubble break-up due to shape instability [3].

The present study focuses on positioning the cavitation bubbles by manipulating the acoustic field. The objective is to control the randomly initialized bubbles to a randomly specified target position within the shortest possible time. The position control is achieved by applying dual-frequency excitation. The previous study [4] proved that the RL agent is capable of learning a policy for controlling a single bubble that is also robust against the variation of uncertainties of the environment. It was also revealed that the bottleneck of the training is the computationally expensive simulation of the environment within each algorithm step due to ODE solution to simulate the microbubble oscillations. To speed-up policy generation a *Gym-style* environment API [5] is developed that utilizes the capabilities of GPU-s to vectorize environmental simulations and run hundreds or even thousands of environment replicas on the same GPU. The present study investigates the impact of parallelization on the speed-up of policy generation on two benchmark problems. The training was carried out using the proximal policy optimization (PPO) algorithm [6].

2. Bubble Position Control

The sketches of the control problems are given in Figure 1. In the first scenario (A), a single bubble is initialized with a random initial position and the task of the agent is to drive the bubble to a randomly prescribed target position. The translational bubble motion is governed by the net force, which is the sum of the Primary Bjerknes force F_{B1} and the drag force F_D. This problem was proposed in the previous paper of the author and solved using the DDPG algorithm [4]. An additional benchmark problem is proposed (B), where a secondary bubble is added to the system, which doubles the observation space size and increases

*Corresponding Author, K. Klapcsik kklapcsik@hds.bme.hu

the problem complexity via introducing the Secondary Bjerknes force F_{B2}. In the case of (B), the initial and the final distance of the bubbles are prescribed symmetrically to the node of the acoustic field. Note that the present problems are restricted to one-dimensional translational bubble motion.

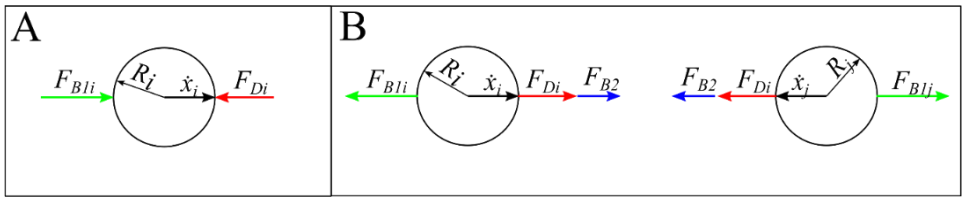

Figure 1. A schematic illustration of the Single Bubble (A) and Dual Bubble (B) control tasks.

2.1. Mathematical Model

The applied mathematical model is derived by Doinikov [7], where the radial and translational motion of bubble i is described by a coupled ordinary differential equation system that consist of the Keller—Miksis equation [8]

$$(1-\frac{\dot{R}_i}{c_L})R_i\ddot{R}_i+(1-\frac{\dot{R}_i}{3c_L})\frac{3}{2}\dot{R}_i^2=(1+\frac{\dot{R}_i}{c_L}+\frac{R_i}{c_L}\frac{d}{dt})\frac{(p_{Li}-p(x_i,t))}{\rho_L}+\frac{\dot{x}_i^2}{4}+G_{ij}^{(rad)}, \quad (1)$$

and the equation of translational bubble motion

$$R_i\ddot{x}_i+3\dot{R}_i\dot{x}_i=\frac{3F_{ex}(x_i,t)}{2\pi\rho_L R_i^2}+3G_{ij}^{(trn)}, \quad (2)$$

where R_i and x_i are the time-dependent bubble radius and position of bubble i, respectively and F_{ex} is the sum of the external forces. The dots stand for derivatives with respect to time. $G_{ij}^{(rad)}$ and $G_{ij}^{(trn)}$ are the coupling terms between the adjacent bubbles and their formulas can be found in [7]. The liquid pressure at the bubble wall is

$$p_{L,i}=(\frac{2\sigma}{R_{0i}}+P_\infty)(\frac{R_{0i}}{R_i})^{3\gamma}-\frac{2\sigma}{R}+4\mu_L\frac{R_i}{R_i}. \quad (3)$$

In the above equations, $\rho_L=998\,kg/m^3$ is the liquid density, $c_L=1500\,m/s$ is the liquid sound speed, $\sigma=0.0725\ N/m$ is the surface tension, $\mu_L=0.001\ Pas$ is the liquid dynamic viscosity and $\gamma=1.4$ is the polytrophic exponent. In addition, $R_{0i}=60\ \mu m$ is the equilibrium bubble radius of bubble i.

2.2. Acoustic manipulation

In the present study, the acoustic field is prescribed as a sum of two standing waves:

$$p(x,t)=P_\infty+P_{A0}\cos(k_0x+\theta_1)\sin(\omega_0t+\theta_0)+P_{A1}\cos(k_1x+\theta_1)\sin(\omega_1t+\theta_1). \quad (4)$$

Here $P_\infty=1\ bar$ is the ambient pressure, while P_{A0} and P_{A1} denote the pressure amplitudes. Similarly, $\omega_0=2\pi f_0$ and $\omega_1=2\pi f_1$ are the angular frequencies, with $k_0=2\pi/\lambda_0$ and $k_1=2\pi/\lambda_1$ representing the wave numbers. $f_0=25\ kHz$ and $f_1=50\ kHz$ are the excitation frequencies, λ_0 and λ_1 are the corresponding wavelengths. The acoustic manipulation is performed by controlling the pressure amplitude and the phase shift values. Since the reinforcement learning agent interacts with its environment in discrete timesteps, the controllable parameters are discrete functions of time. The algorithmic timestep is prescribed as $\Delta t_{RL}=N\tau_0$, where τ_0 denotes the acoustic period. $N=10$ and $N=5$ for scenario (A) and (B), respectively. To properly model the bubble dynamics and the translational motion within each control interval the above governing equations need to be solved using an initial value problem solver.

3. Reinforcement Learning Framework

In the present study, a Reinforcement Learning (RL) Framework is implemented leveraging the parallelization capabilities of GPUs to run massive amounts of environments simultaneously on a single machine. The Framework consists of three interconnected layers. The bottom layer is a GPU-accelerated ordinary differential equation solver, where each GPU thread solves environmental simulations initialized

*Corresponding Author, K. Klapcsik kklapcsik@hds.bme.hu

with different parameters. The ODE solver is implemented in *Python* using the *Numba.Cuda* library following the implementation details of MPGOS [9]. The solver solves independent ordinary differential equations in the form of

$$\dot{x} = f(\boldsymbol{x}, t, \boldsymbol{p_{cp}}, \boldsymbol{p_{sp}}, \boldsymbol{p_{dc}}) \tag{5}$$

where x is the state variable, t is the time, $\boldsymbol{p_{cp}}$ is the vector thread-specific parameters, such as system constants, $\boldsymbol{p_{dc}}$ is the vector of adjustable parameters modified by the RL in discrete steps to enable adaptive control and $\boldsymbol{p_{sp}}$ is the vector of shared parameters, which is common for all instances of ODEs. The applied algorithm is the explicit, adaptive and fourth-order Runge—Kutta—Cash—Karp method with fifth-order error estimation.

The middle layer is a Gym-Style environment API [5] designed to seamlessly interface between the ODE solver and the RL algorithms. To mimic a realistic experimental setup, the state is assumed to be partially observable, i.e., the observation vector includes only positional values, reflecting the limited available information. The observation vector includes the target position $x_{T,i}$ for each bubble, the actual position $x_{i,0}$ of each bubble, and one previous stacked position value $x_{i,-1}$.The preliminary simulations revealed that encoding two position values per bubble into the observation is sufficient to achieve good performance. The reward function is designed to penalize the distance from the target positions.

$$r_t = \frac{1}{N_B} \sum \left(\frac{d_i}{d_{max}}\right)^{0,5} \tag{6}$$

where $d_i = |x_{T,i} - x_i|$ is the distance between the target position $x_{T,i}$ and the actual bubble position $x_{i,0}$, d_{max} is the maximal possible distance, N_B is the number of bubbles. Upon successful control (i.e. when the distances are lower than the tolerance $E_x/\lambda_0 = 0.01$), the agent receives an additional positive reward $R_t = N_B \cdot 10$. Conversely, if the environment terminates due to bubble collision or selects parameters where integration error occurs, the agent is rewarded by an additional penalty $R_t = -200$. To prevent agent from learning to terminate episodes instead of exploring the parameter space, the negative terminal reward is significantly larger in magnitude than the cumulative reward achievable by an untrained model.

The top layer stands for the RL algorithms written by using the *PyTorch* library. For the present study, the Proximal Policy Optimization (PPO) algorithm [6] was implemented. PPO optimizes the surrogate loss function

$$L^{CLIP}(\theta) = \mathbb{E}[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)], \tag{7}$$

where $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{old}}(a_t|s_t)$ is the policy ratio, $\epsilon = 0.2$ is the clipping limit. $\hat{A}_t$ is the advantage function calculated according to the Generalized Advantage Estimation (GAE)

$$\hat{A}_t = \sum_l (\gamma\lambda)^l \delta^V_{t+l}, \qquad \delta^V_t = r_t + \gamma V(s_{t+1}) - V(s_t), \tag{8}$$

where $\gamma = 0.99$ is the discount factor, $\lambda = 0.95$ controls the bias-variance tradeoff, and $V(s)$ is the value function estimator. The value function is trained simultaneously via the minimization of the squared error

$$L^{VF}(\theta) = \left(V_\theta(s_t) - V_t^{target}\right)^2. \tag{9}$$

Combining the above terms and adding the $S[\pi_\theta](s_t)$ entropy bonus, the following objective function needs to be maximized

$$L(\theta) = \mathbb{E}[L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S[\pi_\theta](s_t)], \tag{10}$$

where $c_1 = 0.5$, $c_2 = 0.01$ are coefficients. The PPO algorithm trains two neural networks: namely, an actor (policy) and a value function estimator. Both the actor and the value networks have two hidden layers. For single and dual bubble cases 128 and 256 neurons per layer were used, respectively. The actor employs

*Corresponding Author, K. Klapcsik kklapcsik@hds.bme.hu

a beta distribution which offers improved convergence for bounded action spaces compared to the Gaussian distribution [10]. The learning rate $\alpha = 2.5 \cdot 10^{-4}$ was constant and the minibatch size was 512. This study aims to investigate the effect of environment parallelization on the speed-up of policy generation. To isolate this effect, hyperparameters and neural network architectures were kept constant, while the number of environments and the rollout length were adjusted as experimental variables. The maximum steps (environment-agent interaction) were 5 million per trial. The numerical experiments were carried out on a standard PC equipped with an Intel(R) Core (TM) i7-3770K 3.50GHz CPU, and a NVIDIA GeForce GTX 1070 GPU. After the training, the performance of each trained policy was evaluated by running 10 thousand episodes and collecting the episode lengths and the cumulated episode rewards.

4. Training Performance and Efficiency Analysis of the Reinforcement Learning Agent

Figure 2 shows the episode reward and the episode length for different number of environments. The panels on the left and right show the results for single and dual bubble position control, respectively. For each environment configuration, simulations were repeated using different numbers of rollouts (agent-environment interactions between training phases). Both the boxplots of the episode reward and episode length show consistent results. However, increasing the number of environments above 2048 and using 32 rollout steps decreases the performance. The best performance is observed using configurations with 1024-2048 environments and 8 rollout steps. These configurations result in high episode rewards with the least outliers, and short episode lengths. The maximum timesteps (128 denoted by the red dashed lines) never reached in these cases meaning 100% success rate during evaluation. Figure 3 presents the total runtime. The increase of the number of environments decreases the runtime; however, above 2048 environments the runtime does not decrease significantly.

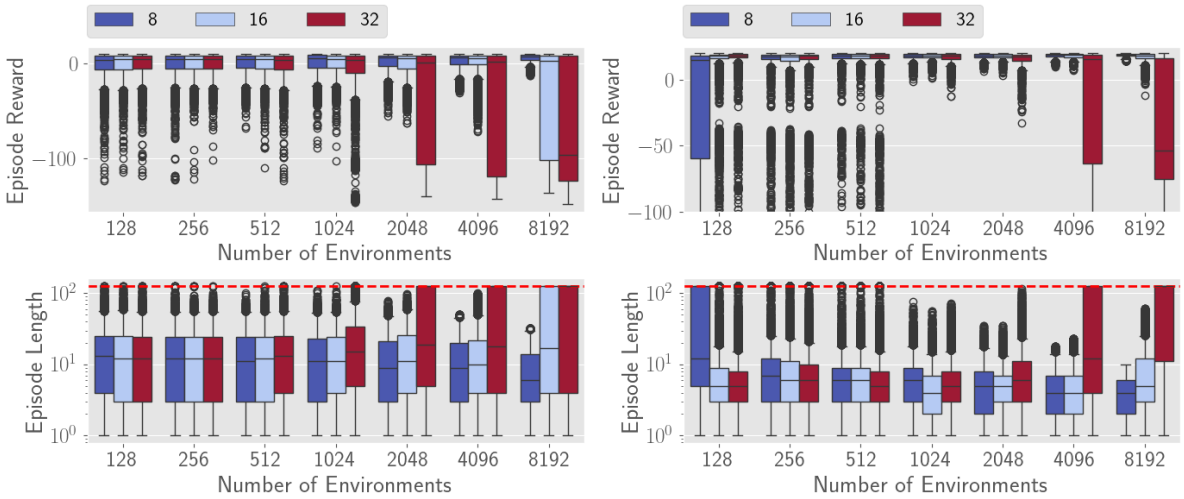

Figure 2. The performance of trained agents measured over 10 thousand evaluation episodes.

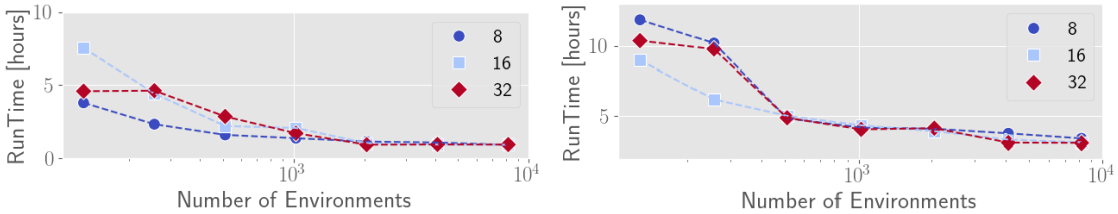

Figure 3. The training time for different number of environments and rollout steps.

5. Position Control

Figure 4 shows two test cases for scenario (A). The panels depict the normalized bubble position (top panels) and pressure amplitude (bottom panels) as the function of acoustic periods. The target positions with a tolerance of $E_x/\lambda_0 = 0.01$ are denoted by the green dashed lines. In the first case (left panels), the

*Corresponding Author, K. Klapcsik kklapcsik@hds.bme.hu

bubble is driven away from the antinode located at $x/\lambda_0 = 0.25$. This control is achieved by choosing an appropriate combination of high- and low-pressure amplitude values. The second case (right panels) demonstrates control in the opposite direction. Again, the problem is solved by choosing a combination of high- and low-pressure amplitude values to attract the bubbles towards an intermediate equilibrium position located around $x/\lambda_0 = 0.1$ then trigger rapid translational motion. In both cases, the agent leverages the nonlinear nature of translational bubble motion at this parameter set. Overall, the PPO algorithm identified a similar solution to the one found by DDPG in the previous paper [4]. Naturally, the trained agent follows a stochastic policy resulting in slight variation in the selected actions.

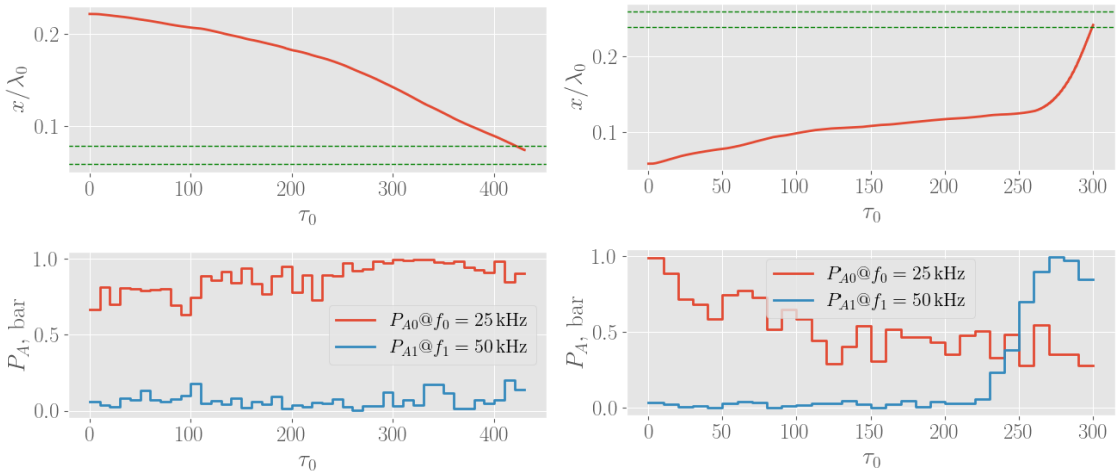

Figure 4. Testcases for single bubble position control.

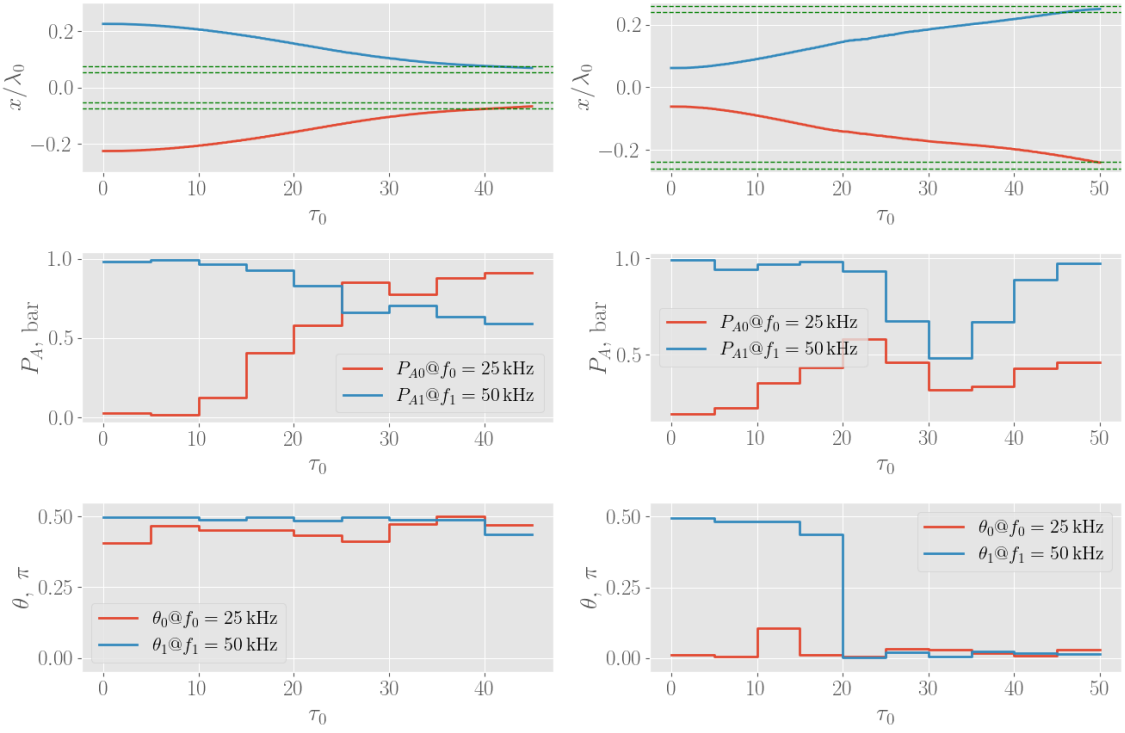

Figure 5. Testcases for dual bubble position control.

In the dual bubble case, the emerging secondary Bjerknes force increases the problem's complexity. The primary Bjerknes force dominates the bubble motion; thus, bubbles initialized close to the antinodes cannot be driven away from them. Conversely, the bubbles initialized close to each other and collided within a few

*Corresponding Author, K. Klapcsik kklapcsik@hds.bme.hu

acoustic periods due to the attractive secondary Bjerknes force. On a wide range of bubble positions, no mathematical solutions exist by controlling only the pressure amplitude values. Therefore, the agent was allowed to control the phase shift as well. Figure 5 shows two test cases for dual bubble control, which are unsolvable based on the linear theory. In the first case (left panels), the bubbles are initialized close to the antinodes and the agent was able to drive the bubble in the opposite direction expected from linear theory (see Figure 1. B). The second test case (right panels) shows a solution where the bubbles are initialized close to each other and the task was to increase the distance between the bubbles. Note that the minimum distance was empirically set to $d_{min} = 0.1 \cdot \lambda_0$, as bubble collision becomes unavoidable below this threshold. The agent was capable to drive the bubbles to their destination. Notably, the bubbles exhibit faster translational motion; thus, the control tasks were solved under fewer acoustic periods compared to the single bubble cases.

6. Conclusions

The present study investigates the overall performance of the utilization of GPU-accelerated parallel environments on the proposed two benchmark problems. The result showed that PPO consistently learns effective policy across a wide range of parallel environments. The optimal training performance was achieved using 2048 and 1024 environments for single and dual bubble position control, respectively. It was also found that the algorithm performs better using a low number of rollout steps (8 or 16) for these specific problems. This low number of rollouts is justified by the typical episode length, which is approximately 10-15 depending on the initial and target positions. Although the present study considers a fixed bubble size, the framework allows to generalize policy for various bubble sizes, which could be an additional benefit of using parallel environments. Moreover, one can investigate different scenarios, or improve the underlying governing equations by implementing more accurate mathematical model. This study does not focus on low-level optimization but there is significant room for improvement in optimizing the kernel code through careful profiling on different hardware. The Environment API could be further optimized by leveraging Cuda Streams to overlap copy and compute operations. Furthermore, utilizing multiple GPUs would allow to overlap the training and rollout phases. These improvements could help in reducing performance bottlenecks, which arise as problem complexity increases.

7. Acknowledgement

The author acknowledges financial support from the Hungarian National Research, Development and Innovation Office via NKFIH Grants OTKA PD 142254 and the Ministry of Culture and Innovation of Hungary under project TKP-6-6/PALY-2021, funded by the TKP2021-NVA scheme.

References

1. Cs. Kalmár, T. Turányi, I. Gy. Zsély, M. Papp, F. Hegedűs (2022). *The importance of chemical mechanisms in sonochemical modelling*, Ultrason. Sonochem. 83:105925
2. E. Perolina, S. Meissner, B. Raos, B. Harland, S. Thakur, D. Svirskis (2024). *Translating ultrasound-mediated drug delivery technologies for CNS applications*, Adv. Drug. Deliv. Rev. 208:115274
3. F. F. Arifi, M. L. Calvisi (2021). *Optimal Control of Encapsulated Microbubble for Biomedicine*, 11th International Symposium on Cavitation, Daejon, Korea, 10-13 May 2021
4. K. Klapcsik, B. Gyires-Tóth, J. M. Rosselló, F. Hegedűs (2023) *Position control of an acoustic cavitation bubble by reinforcement learning,* arXiv:2312.05674
5. G, Brockman, V. Cheung, L. Petterson, J. Scheider, J. Schulman, J. Tang, W. Zaremba (2016) *OpenAI Gym*, arXiv:1606.01540
6. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov (2017) *Proximal Policy Optimization Algorithms*, arXiv:1707.06347
7. A. A. Doinikov (2004). *Mathematical model for collective bubble dynamics in strong ultrasound fields*, J. Acoust. Soc. Am. 116: 821–827.
8. J. B. Keller, M. Miksis (1980) *Bubble oscillations of large amplitude*, J. Acoust. Soc. Am., 68(2):628-633
9. F. Hegedűs (2021) *Program package MPGOS: Challenges and solutions during the integration of a large number of independent ODE systems using GPUs,* Commun. Nonlinear Sci. Numer. Simul., 97:105732
10. I. G. B. Petrazzini, E. A. Antonelo (2021) *Proximal Policy Optimization with Continuous Bounded Action Space via the Beta Distribution*, arXiv:2111.02202

*Corresponding Author, K. Klapcsik kklapcsik@hds.bme.hu