PDE Based Motion Planning for Robotics - A Case Study

Ashwin Nedungadi[1]

Technische Universitat Dortmund, Germany
ashwin.nedungadi@tu-dortmund.de

Abstract. Motion planning is a widely studied topic in robotics due to it's relevance for obstacle avoidance and path planning in autonomous robots. In this case study, we implement a novel PDE based motion planning technique and illustrate using a robot arm as example. This work also builds upon the previous works by proposing a parallel computing approach to the problems encountered from solving PDEs and discusses it's impact on the motion planning problem.

Keywords: Motion Planning · Partial Differential Equations · Robotics

1 Introduction

Motion planning or path planning for robotics has a long history that can be traced back to the late 60s during the early development of computer controlled robots. A classic thought experiment which demonstrates the motion planning problem is the piano mover's problem[1][2] in which a piano must be moved through an apartment cluttered by obstacles and in which the piano must clear tight corners. The problem is solved by considering the free configuration space of the piano and if a viable obstacle-free path can be drawn between the start and goal positions.
Similarly, the position or configuration of a robot is generally described by a number of variables, for mobile robots, this is the 6D pose[1] of the robot. For an articulated arm, these variables are usually positions of the different joints on the robot. Motion planning can also be split into two broad terms, when all degrees of freedom can be changed independently like a fully actuated robot arm, we call it Holonomic motion planning. In this case, the existence of a collision-free path is characterized by the connected component in the free configuration space. In this case the problem reduces down to simply finding a path between two points in this free space. In contrast, when the degrees of freedom of the robot are not independent[2], then it is known as Non-Holonomic motion planning. This is a fundamental issue for most types of mobile robots such as a boxcar or unicycle[3][6] and the motion planning for such systems turns out to be much

[1] 6D pose consists of (x,y,z,roll,pitch,yaw) and describes the position and orientation of a robot in 3D space.
[2] e.g. a unicycle cannot rotate around it's axis without also changing it's position

more challenging than for Holonomic systems.In this paper, we try to apply a novel PDE based method[3] to control an articulated 2DOF robot arm and improve upon it's limitations by parallelizing the PDE solver using multiple cores in MATLAB using the parallel computing toolbox[14][16].

1.1 State of the Art

Numerous methods employing mathematics[11], exist for the motion planning problem such as search based algorithms which use graphs (e.g. A* or Dijkstra) and computes paths or trajectories over a discrete representation of the problem[9], a probabilistic roadmap method[5] which takes random samples from the configuration space, testing them for whether they are free, and using a local planner to connect these configurations to other nearby configurations or Artificial potential field based methods in which the robot is regarded as a point under the influence of an artificial potential field and is then subjected to attractive forces and repulsive forces.[7][8].
In addition, using sinusoids for trajectories at integrally related frequencies to achieve motion[6] as well as using learning based planners which utilize neural networks and are based on the principles of imitation learning utilizing "expert paths" from models previously trained on graph based planers[10] have been proposed.
While the above methods are proven to be robust in dynamic environments and for linear systems that are holonomic, most of the methods above cannot be extended to non-holonomic systems. For this, a novel method that is based on motion planning for non-holonomic systems is presented[3] which uses parabolic partial differential equations that drives a system to move from a initial state x_i to the final state x_f avoiding obstacles and respecting the given input constraints for the system.

2 Mathematical Background & Problem Statement

In this section, we present some background on the underlying PDE problem[3]. We define the geometric heat flow equation (GHF):

$$\frac{\partial}{\partial s}v_i(t,s) = \frac{\partial^2}{\partial t^2}v_i(t,s) + \sum_{j,k} \Gamma^i_{jk}\frac{\partial v_j}{\partial t}\frac{\partial v_k}{\partial t} \tag{1}$$

where Γ^i_{jk} are the Christoffel symbols for the inner product.

$$v_i(t,s+ds) := \frac{\partial^2}{\partial t^2}v_i(t,s) + \sum_{j,k} \Gamma^i_{jk}(v(t,s))\frac{\partial v_j(t,s)}{\partial t}\frac{\partial v_k(t,s)}{\partial t} \tag{2}$$

with boundary conditions $v(0,s_0) = x_0$ and $v(1,s_0) = x_1$ and initial condition $v(t,0)$ which can be a customized initial curve which satisfies certain conditions. This is a system of partial differential equations, coupled by the second terms

and can be solved using an explicit iterative procedure using finite differences in t. The curves $v(t,.) : [0,1] \to R^n$ act as state variables and S is the time-variable for the curve evolution. For large values of C, we observe that the solution $v(t,\infty)$ tends to be an admissible curve joining v_0 and v_1.

$$\frac{d}{dt}v(t,\infty) \in \Delta_0(v(t,\infty)) \tag{3}$$

from which we can extract the controls $u(t)$.

3 Implementation

Planar Robot Arm Dynamics

Our goal is to plan the motion of the tip of the robot arm from an initial state x_i to a final state x_f in a planar space with coordinates $(x, y, \theta_1, \theta_2)$ without any obstacles ($b(x) \equiv 1$). The system has 2 degrees of freedom and we can obtain the holonomic constraints below and encode them into the $G(x)$ matrix:

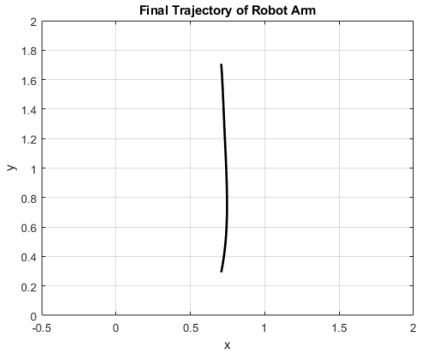
$$\begin{cases} q_1(\mathbf{x}) = L_1 \cos(\theta_1) + L_2 \cos(\theta_2) - x = 0 \\ q_2(\mathbf{x}) = L_1 \sin(\theta_1) + L_2 \sin(\theta_2) - y = 0 \end{cases} \tag{4}$$

$$G(x) = \begin{pmatrix} \sin^2\theta_1 + \sin^2\theta_2 + k & -\frac{\sin 2\theta_1}{2} - \frac{\sin 2\theta_2}{2} & (k-1)\sin\theta_1 & (k-1)\sin\theta_2 \\ -\frac{\sin 2\theta_1}{2} - \frac{\sin 2\theta_2}{2} & \cos^2\theta_1 + \cos^2\theta_2 + k & -(k-1)\cos\theta_1 & -(k-1)\cos\theta_2 \\ (k-1)\sin\theta_1 & -(k-1)\cos\theta_1 & k+1 & k\cos(\theta_1 - \theta_2) \\ (k-1)\sin\theta_2 & -(k-1)\cos\theta_2 & k\cos\theta_1 - \theta_2 & k+1 \end{pmatrix} \tag{5}$$

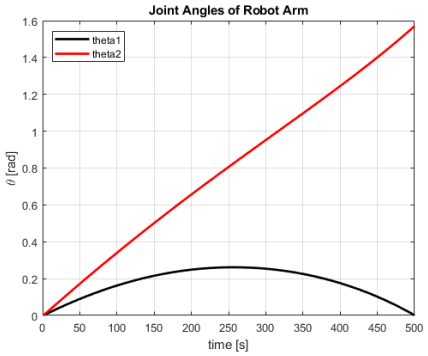
We then solve the above defined parabolic equation using MATLAB with the **pdepe** function. We set two different boundary conditions for the initial condition and obtain the graphs below which show the final trajectory of the robot tool tip and the changing joint angles.
The integration time for the PDE is taken as 5 sec with 500 discretization steps.

Initial conditions	x	y	θ_1	θ_2
values of u_0	1.0	1.2	$\frac{\pi}{2}$	$-\frac{\pi}{2}$
Boundary condition 1	x	y	θ_1	θ_2
Initial	$\frac{\sqrt{2}}{2}$	$1 - \frac{\sqrt{2}}{2}$	$\frac{\pi}{2}$	$-\frac{\pi}{4}$
Final	$\frac{\sqrt{2}}{2}$	$1 + \frac{\sqrt{2}}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{4}$
Boundary condition 2	x	y	θ_1	θ_2
Initial	1.5	0.2	$\frac{\pi}{2}$	$\frac{\pi}{4}$
Final	0.0	2.0	$\frac{\pi}{2}$	$-\frac{\pi}{4}$

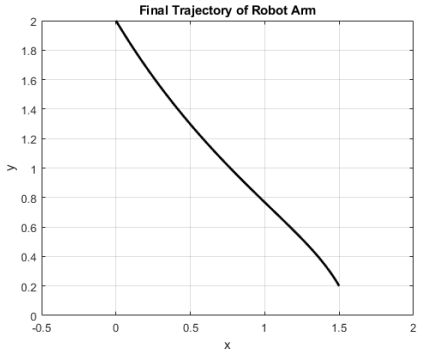
Table 1: Parameter Table



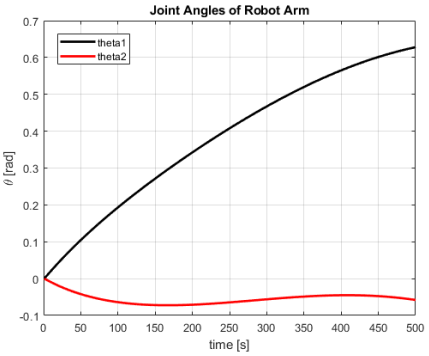
(a) Trajectory, boundary condition 1



(b) Joint angles, boundary condition 1



(c) Trajectory, boundary condition 2



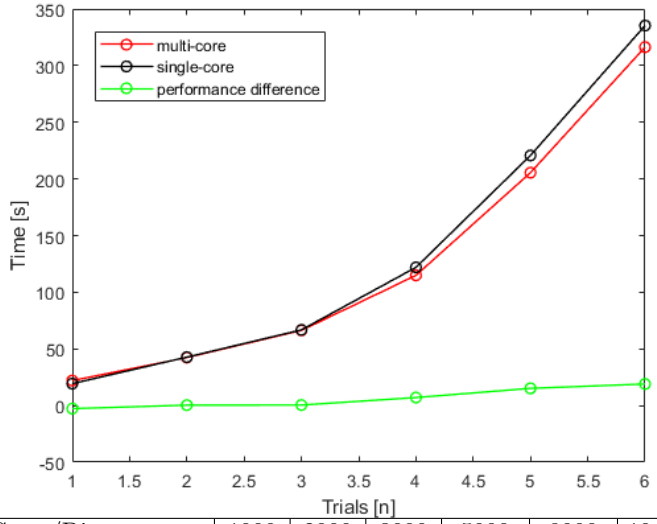
(d) Joint angles, boundary condition 2

4 Implementation with Parallelism & Conclusion

The numerically intensive part of the above method lies in solving the geometric heat flow equation which is a system of parabolic partial differential equations using MATLAB. However, MATLAB does not use multiple cores by default to solve PDEs or for computing loops[4]. Hence, with the Parallel Computing Toolbox, we can parallelize the computation of the code directly without the need for sophisticated parallel algorithms. starting a parallel pool with the **parpool** function and running the code leads to better computational time for the above mentioned PDE problem. This is equivalent to using **parfor** within the script to compute for-loops.

Since changing the PDE problem definition to be a more computationally intensive PDE problem (e.g. simulating a 6DOF robot arm) is outside the scope of this case study, the integration time steps were modified to be longer to simulate more computational load and the results from the single-core and multi-core experiments are compared. The CPU used was an intel i5-10300H with 4 cores operating at 2.50 Ghz with 16 GB of RAM.

As we can observe from the graph and the table showing the single-core vs. multi-core performance with respect to time in seconds, the performance dif-



Cores/Discrete steps	1000	2000	3000	5000	8000	10,000
single-core time	22.051	42.719	66.782	121.975	220.724	335.344
multi-core time	19.340	42.362	66.338	114.912	205.553	316.325

Table 2: single vs. multi core performance with increasing step size

ference that parallelization provides at smaller discretization steps are almost negligible. However, as we increase the step size into thousands, we start to notice a slight difference. Due to the memory limitations of the local computer, 10,000 steps was the maximum possible step size that could be simulated. But it is easy to imagine what this graph would look like if we extrapolate it into the tens of thousands and hundreds of thousands discrete steps.
We also observe that the performance increase from using 4 cores is not that significant, this could be because of Amdahl's[15] law[3] and the fact that our code is not optimized for parallel computing or due to the limitations of the local computer and that there exists a "cold start" penalty for using the parallel pools. We must also consider that MATLAB itself is not a great program for writing optimized code and better performance can be achieved using parallel algorithms[16] and dedicated implementation with software such as Julia or FEniCS.

[3] The theoretical speedup from using more cores is limited by the serial part of the program.

References

1. Wilson, D., Davenport, J. H., England, M., & Bradford, R. (2013). A "Piano Movers" Problem Reformulated. 2013 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing. https://doi.org/10.1109/synasc.2013.14
2. Schwartz, J.T. and Sharir, M. (1983), On the "piano movers'" problem I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. Comm. Pure Appl. Math., 36: 345-398. https://doi.org/10.1002/cpa.3160360305
3. M. A. Belabbas and Shenyu Liu, "New method for motion planning for nonholonomic systems using partial differential equations," 2017 American Control Conference (ACC), 2017, pp. 4189-4194, https://doi.org/10.23919/ACC.2017.7963599
4. S. Liu and M.A. Belabbas, "A homotopy method for motion planning" Arxiv 1901.10094, 2019
5. L. E. Kavraki, P. Svestka, J. -. Latombe and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," in IEEE Transactions on Robotics and Automation, vol. 12, no. 4, pp. 566-580, Aug. 1996, doi: 10.1109/70.508439.
6. R. M. Murray and S. S. Sastry, "Nonholonomic motion planning: steering using sinusoids," in IEEE Transactions on Automatic Control, vol. 38, no. 5, pp. 700-716, May 1993, https://doi.org/10.1109/9.277235.
7. Hamidreza Heidari, Martin Saska, Collision-free trajectory planning of multirotor UAVs in a wind condition based on modified potential field, Mechanism and Machine Theory, Volume 156, 2021, 104140, ISSN 0094-114X, https://doi.org/10.1016/j.mechmachtheory.2020.104140.
8. Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. Proceedings. 1985 IEEE International Conference on Robotics and Automation, 2, 500–505.
9. P. C. Chen and Y. K. Hwang, "SANDROS: a dynamic graph search algorithm for motion planning," in IEEE Transactions on Robotics and Automation, vol. 14, no. 3, pp. 390-403, June 1998, doi: 10.1109/70.678449.
10. R. Reinhart, T. Dang, E. Hand, C. Papachristos and K. Alexis, "Learning-based Path Planning for Autonomous Exploration of Subterranean Environments," 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 1215-1221, doi: 10.1109/ICRA40945.2020.9196662.
11. Zhang, J. (2021). AI based Algorithms of Path Planning, Navigation and Control for Mobile Ground Robots and UAVs.
12. S. M. LaValle, Planning Algorithms. Cambridge, U.K.: Cambridge University Press, 2006.
13. J. Laumond, Robot motion planning and control, ser. Lecture notes in control and information sciences. Springer, 1998.
14. Jianping Zhu. Mississippi State University. Solving Partial Differential Equations on Parallel Computers. https://doi.org/10.1142/2190
15. Bryant, Randal E.; David, O'Hallaron (2016), Computer Systems: A Programmer's Perspective (3 ed.), Pearson Education, p. 58, ISBN 978-1-488-67207-1
16. S. Pschenica, (1994) "Parallel Algorithms for Solving Partial Differential Equations," Journal of the Iowa Academy of Science: JIAS, 101(2), 70-72. Available at: https://scholarworks.uni.edu/jias/vol101/iss2/11