# Exercise 11A – Harmonic Potential Fields

## Q1 Potential Field Planning

In this exercise, we are going to implement motion planning functionality based on the Harmonic Potential Field approach [1] in the lecture video. Starting from a 2D obstacle map (a discrete grid), we will implement an iterative numeric approximation to the solution of the Laplace equation. Finally, we will extract the solution path between the robot position and the goal location.

In this exercise sheet we will walk through the steps of the process, these must then be implemented in the code.

### Q1.1 Initialisation

Figure 1 shows the obstacle map we will use for this exercise. Black cells represent obstacles, and white cells are freely traversable. We define the map's origin at the top left corner, corresponding to cell (1,1) (or (0,0) with 0-indexing like Python). The first dimension faces downward, and the second dimension towards the right. The robot and goal locations are marked by "R" and "G", respectively. Note that the map is assumed to have been inflated by the robot's radius already. It thus represents the configuration space, and planning can proceed directly bu considering the robot as a point.

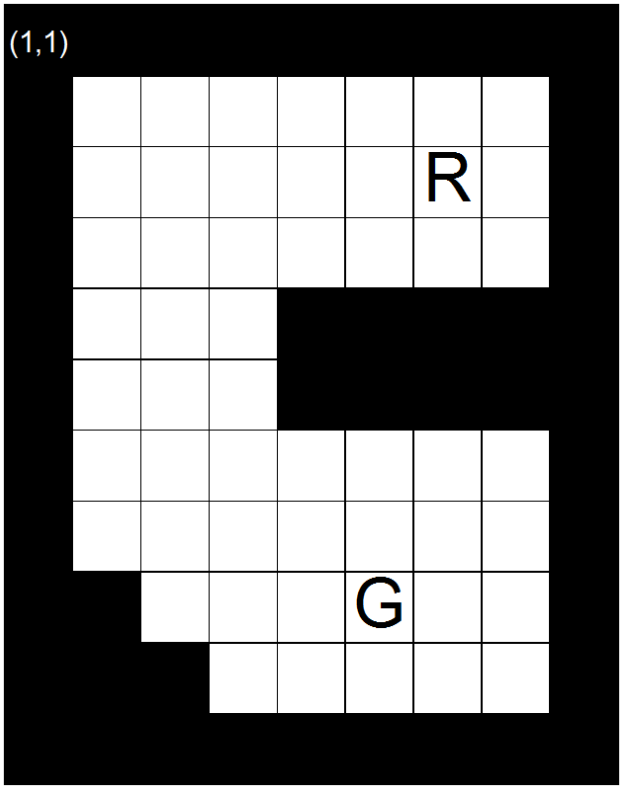


Figure 1: Obstacle map for harmonic potential field exercise.

The first step is to construct this space as a grid representation. In the code, a simple 2D binary array suffices, with 1 for traversable cells and 0 for non-traversable (obstacle) cells.

Next, we need to create the scalar array $U$ that will store the potential field. This will contain the values of the potential at each location $q$ in the configuration space $\Omega$. We will use the Laplace approach, so values in the field are in the interval $U(q) \in [0, 1]\ \forall q \in \Omega$.

1. What values should these cells take at initialisation:
   (a) obstacle cells?
   (b) goal cell?
   (c) other cells?
2. What is the effect of the initialisation value of the other (non-obstacle) cells? Try different values to see the effect in the code in terms of stability, solve time and solution.

**Answer:**

1. What values should these cells take at initialisation:
   (a) Obstacle cells: The maximum potential (1)
   (b) Goal cell: Minimum potential (0)
   (c) Other cells: Any value $\in (0, 1)$ would be valid here.
2. All valid values should always converge to the same solution, as long as the obstacles and goal are held to their respective initial values. The effect is that the potentials have to propagate from high (obstacle) or low (goal) potential values across the field.

## Q1.2 Iterative updates

The harmonic potential field method uses an iterative solution to calculate the potential field. This means that (non goal or obstacle) cells are repeatedly updated as a function of the values of their neighbouring cells until convergence. In this exercise, we will iteratively numerically solve the Laplace equation ($\nabla^2 U(q) = 0$) with Dirichlet boundary conditions on the potential field map $U$. Recall from the lecture that the iterative discrete Laplace scheme takes the following form for a discrete, square grid:

$$U^{k+1}(q) = \frac{1}{2n} \sum i = 1^n \left( U^k(q - e_i) + U^k(q + e_i) \right) \tag{1}$$

where $k \in \mathcal{Z}^+$ is a time index or iteration step, $n$ is the number of dimensions, and $e_i$ represents a shift of 1 neighbour in the respective dimension. Thus, if we have a 1D grid ($n = 1$), this would amount to the average of the left and right neighbours, in a 2D grid ($n = 2$) this is the average of the four direct neighbours left, right, above and below. All non-boundary, non-goal cells are iteratively updated until convergence (usually a tolerance on the maximum change of any individual cell in each full map update).

1. Implement this update scheme in the associated code. NOTE: Be careful how you update your data. Pay attention to the fact that the update rule is a function of the old potential values ($U^{k+1} = f(U^k)$), so make sure you aren't overwriting old values as you update the potential field!
2. Bonus question: This operation looks a lot like an image convolution. If we were to perform this using a convolution filter, what would the convolution matrix be? (Note: assume that the obstacle cells and goal would be masked after the convolution operation to keep their values at 1 and 0 respectively)

**Answer:**

1. (See code). This essentially amounts to running a loop over all valid cells, and updating them with the specified function until convergence.
2. We could convolve the map with the matrix $\begin{bmatrix} 0 & 0.25 & 0 \\ 0.25 & 0 & 0.25 \\ 0 & 0.25 & 0 \end{bmatrix}$.

### Q1.3 Backtrack solution (output path)

Finally, we will extract the solution path. As solutions to the Laplace equation only exhibit a single minimum at the goal location (and no saddle points), one way of extracting the solution path is via a greedy search along the steepest negative gradient starting from the robot's starting position.

1. Write a function to extract the path (cell sequence) given the final converged potential field $U$. In this exercise, assume that the robot can move from the current cell to any neighbouring cell including diagonals (8-connected grid).

## References

[1] J.-O. Kim and P.K. Khosla. Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, 8(3):338–349, 1992.