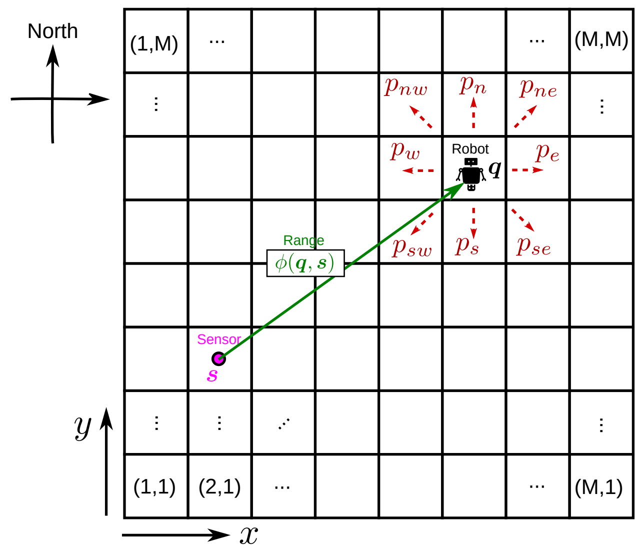
Homework 9 (due at 11pm November 23rd, 2022)

1 Problem: Grid-Based Recursive Bayesian Estimator

This problem should be completed using the data provided in the `rbe_data.mat` and accompanying MATLAB files contained in `h9.zip`.

Environment. Consider a robot that moves over a 2D grid as shown below. The horizontal and vertical axes are discretized into $M = 35$ intervals leading to a total of $M^2 = 1225$ grid cells.



The bottom left corner of the grid is denoted as the first state $\boldsymbol{q}_1 = (x, y) = (1, 1)$ and the top right corner as the last state $\boldsymbol{q}_{M^2} = (x, y) = (M, M)$ as shown below. The discrete state space of the robot is

$$\mathcal{Q} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_{M^2}\} = \{(1, 1), (2, 1), \cdots (M, 1), (1, 2), (2, 2), \cdots, (M, M)\} \tag{1}$$

After loading the `rbe_data.mat` file into your workspace you can access the discrete state space `Q` using the following syntax for a cell array:

```
>> Q{1}
ans =
     1
     1
>> Q{M}
ans =
    35
     1
>> Q{M^2}
ans =
    35
    35
```

Motion Model. A simple state transition model for the robot is assumed wherein it moves "north" (i.e., up one cell) with probability p_n, "northeast" (i.e., up and over one cell to the right) with probability p_{ne}, and so on for all cells that are immediately surrounding the robot's current cell [1]. Thus, the state transition model (for transitioning *from* state $\boldsymbol{q}_j$ *to* state $\boldsymbol{q}_i$ or equivalently the probability of $\boldsymbol{q}_i$ given $\boldsymbol{q}_j$) can be written as

$$\boldsymbol{\pi}_k(\boldsymbol{q}_i|\boldsymbol{q}_j;\boldsymbol{\theta}_k) = \begin{cases} p_n & \text{if } y_i = y_j + 1 \text{ and } x_i = x_j \\ p_{ne} & \text{if } y_i = y_j + 1 \text{ and } x_i = x_j + 1 \\ p_e & \text{if } y_i = y_j \text{ and } x_i = x_j + 1 \\ p_{se} & \text{if } y_i = y_j - 1 \text{ and } x_i = x_j + 1 \\ p_s & \text{if } y_i = y_j - 1 \text{ and } x_i = x_j \\ p_{sw} & \text{if } y_i = y_j - 1 \text{ and } x_i = x_j - 1 \\ p_w & \text{if } y_i = y_j \text{ and } x_i = x_j - 1 \\ p_{nw} & \text{if } y_i = y_j + 1 \text{ and } x_i = x_j - 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The subscript k is used to denote that the probabilities

$$\boldsymbol{\theta}_k = [p_n,\ p_{ne},\ p_e,\ p_{se},\ p_s,\ p_{sw},\ p_w,\ p_{nw}]^{\mathrm{T}} \tag{3}$$

may vary over time. In your assignment you will test your filter with two different models for the probability parameters.

Model 1 (True Model): The first model is the actual model used by the robot and is provided as the following function:

```
function [probParams] = probModel(k,Nmoves)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description: this function returns the true probability parameters
% of the state-transition model used when generating the data set.
%
% Input:
%   k : an integer indicating the current time step. 1 <= k <= Nmoves
%   Nmoves : the total number of time-steps in the data set
%
% Output: a vector with the following structure
%   probParams = [pn pne pe pse ps psw pw pnw];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Model 2 (Diffusion Model): The second model assumes no knowledge of the robot's motion and assigns an equal probability to all directions of motion.

```
probParams = ones([8 1])*1/8;
```

[1] This arrangement is called the Moore neighborhood

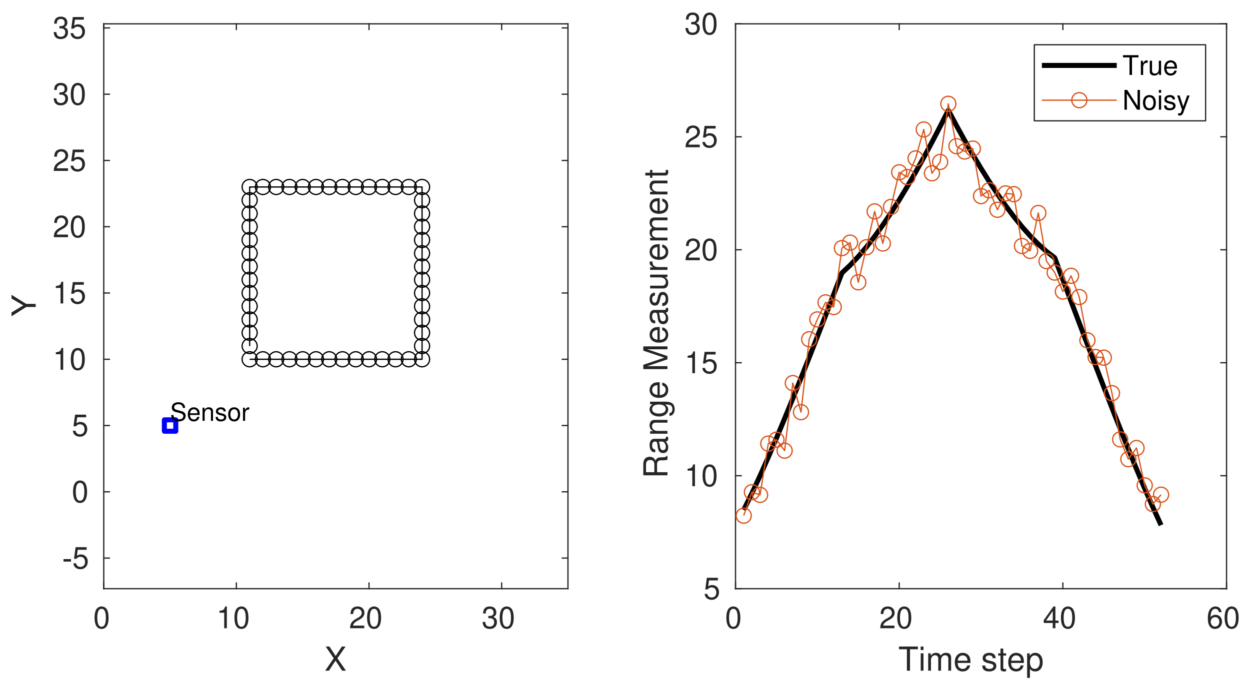
Measurement model. Suppose there is one range sensor located at the state $\boldsymbol{s} = (x_s, y_s) = (5, 5)$ in the lower left corner. The sensor reports the range to the robot corrupted by noise with $\sigma_v = 3$ meters. That is, the sensor reports

$$y(\boldsymbol{q}_j) = \phi(\boldsymbol{q}_j, \boldsymbol{s}) + v \tag{4}$$

where $\phi(\boldsymbol{q}_j, \boldsymbol{s}) = \sqrt{(x_s - x_j)^2 + (y_s - y_j)^2}$ is a nonlinear output and $v \sim \mathcal{N}(0, \sigma_v^2)$ is the measurement noise. The likelihood function of the data y_k given the state $\boldsymbol{q}_j$ can be formulated as

$$L(y_k|\boldsymbol{q}_j) = p(y_k|\boldsymbol{q}_j) = \frac{1}{\sigma_v\sqrt{2\pi}}\exp\left(-(y_k - \phi(\boldsymbol{q}_j, \boldsymbol{s}))^2/2\sigma_v^2\right) \ , \tag{5}$$

Ground Truth. The actual motion of the robot consists of a square pattern with `Nmoves = 52` time steps as shown below. The robot begins in state $\boldsymbol{q}_{361} = (11, 11)$ and moves north. The ground truth is provided in the variable `x_true`.



Homework: Recursive Bayesian Estimator. Your homework is to design a recursive Bayesian estimator by modifying the `for` loop within the function `RBE_template.m` for the robot to predict its position on the grid by processing the `y_noisy` range measurments provided. The probability of each cell state is initialized with a uniform distribution (i.e., $1/M^2$ for each cell). Keep track of all the probabilities by inserting each vector as a new column into a matrix `p`. At the end of the filter's execution you should have a matrix of probabilities at each time-step:

```
>> size(p)
ans =
          52        1225
```

To browse through the frames of a movie showing the posterior you can uncomment the last two lines of the `RBE_template.m`.

Note: The zip file contains four additional files: `rbe_data.mat` , `probModel.m`, `printSnapshots.m` and `playMovie.m` that are needed for the assignment. Ensure you are running MATLAB with the contents of the zip file in your working directory.

Required submission The code will automatically produce a file `snapshots.pdf` in your working directory. You should submit two such snapshot pdf files—one for the true motion model and one for the diffusion model—along with your code to demonstrate you've completed the assignment.