# 知能情報システム創成（人工知能演習）

担当教員：西　竜志

2025 年 4 月 1 日

## 概要

知能情報システム創成（人工知能演習）では，講義科目「人工知能」で学んだ内容に関する演習を行う．この演習では，人工知能の代表的なアルゴリズムを実装し，実行できるようになることを目的とする．人工知能の教科書『イラストで学ぶ 人工知能概論』[1] を必ず持参すること．また，演習に関わる内容は，教科書やサポートページ (`http://ai.tanichu.com/`) のスライドなどで復習しておくこと．演習課題についてわからないことがあれば積極的に TA に質問すること．この演習では，Python を用いて，迷路環境を題材にした各種アルゴリズムの実装を行う．Python の実装方法，及び numpy，matplotlib についてわからないことがあれば，書籍や Web サイトなどで調べること．

また，**知能情報システム創成（人工知能演習）のサポートサイト**にて，演習課題に関する補足情報を掲載している．各自でサポートサイトを確認しながら課題を進めると良い．

`https://sites.google.com/site/souseiai/`

### スケジュール

授業スケジュールの詳細はシラバスを参照すること．manaba+R にて随時，授業に関する連絡を行うことがあるので，各自で確認すること．

**事前準備**　PC への開発環境の構築
**第 1 回目**　シミュレーション環境の構築：概要説明，迷路の作成
**第 2 回目**　基本的な探索手法（教科書 第 2 章）：幅優先探索，深さ優先探索
**第 3 回目**　最短経路探索（教科書 第 3 章）：最適探索，最良優先探索，A*アルゴリズム
**第 4〜7 回目**　選択課題

- 強化学習（教科書 第 7 章）：Q 学習
- 位置推定（ベイズフィルタ）（教科書 第 8 章）
- 位置推定（粒子フィルタ）（教科書 第 9 章）

### 成績評価について

課題の提出，達成状況，学習態度などで総合評価を行う．また，**すべての必須課題を完了していることを単位付与の条件**とする．オプション課題は加点項目となる．「発展」に関しては，成績評価の範囲ではないが，実装した手法のより深い理解のためにやってみることを勧める．

4〜7 回目の課題は**選択課題**とする．強化学習・ベイズフィルタ・粒子フィルタの課題のどれか一単元の必須課題を完了していれば，選択課題の単位要件は満たす．選択課題については，遅延の減点は無い．複数の選択課題を達成していた場合は，オプション課題と同様に加点対象となる．

人工知能演習では，教員・TA が課題のチェックを行う．チェック時には，実装したプログラムを実行し，実行結果と実装したプログラムについて説明をすること．その際，課題の要件を満たさなければ，課題完了とはみなさない．また，教員・TA によるチェックは演習時間内のみとする．

## 参考文献

[1] 谷口忠大. イラストで学ぶ 人工知能概論. 講談社, 2014.

## 事前準備　PC への開発環境の構築

人工知能演習の課題に取り組む前の準備として，各自のパソコンに Python の開発環境を構築すること．ただし，既に開発環境を構築済みの場合は，事前準備をとばしても構わない．自らのパソコンに Python の環境を持たない者に関しては，これを機会に自らの PC で開発環境を構築することを推奨する．

環境構築については，環境構築ガイド（`https://www.python.jp/install/install.html`）が参考になる．課題に必要なライブラリは，python 3 系，numpy，matplotlib である．参考までに，情報教室 PC のバージョン情報はサポートサイトにて提供する．必ずしも上記のバージョンに合わせる必要はないが，バージョンにより若干，仕様が変更されている場合がある．インストールされているバージョンを確認の上，課題を進めること．

PC を忘れたり，開発環境が構築されていない場合は、演習を進めれないため注意して準備する事.

# 1　迷路の作成

今後の演習で使用するためのシミュレーション環境（迷路環境）を構築する．迷路の描画用の関数に関する情報は，知能情報システム創成（人工知能演習）のサポートサイト `https://sites.google.com/site/souseiai/` を参照すること．演習で使用する配布プログラムについてもサポートサイトからダウンロードできる．

図 1 に matplotlib で描画した迷路の一例を示す．また，図 2 に図 1 の迷路のテキスト表現（プログラム上では二次元配列で表される）を示す．

**必須課題 1-1**　自分のオリジナルの迷路（9 × 9 サイズ）をノートに書いた上で，それを壁を 1，通路を 0 とし，数字の間を半角スペースで区切った**二次元配列**として sample_maze.txt に保存せよ．必ずスタートとゴールまでの経路が存在し，道の幅はすべて 1 であること．また，必ず外周はすべて壁で囲むこと．スタート位置・ゴール位置はそれぞれ**左上と右下**に固定とする．

**必須課題 1-2**　課題 1-1 で作った sample_maze.txt を numpy の loadtxt 関数を用いて numpy 配列 (numpy.ndarray) として読み込み，この迷路を上下逆転させた迷路を作成する処理を行った後，sample_maze_reverse.txt として出力するプログラムを作れ．保存の際は numpy の savetxt を用いて構わない（fmt=”%d”のオプションを付けることで整数部分のみが保存できる）．

**必須課題 1-3**　matplotlib の imshow 関数を用いて迷路を可視化せよ．例えば，迷路を記憶している ndarray 変数が maze であった場合 plt.imshow(maze, cmap=”binary”) で表示可能である．

**必須課題 1-4**　迷路作成プログラム（make_maze.py）を用いて迷路のテキストファイルを作成せよ．

**必須課題 1-5**　迷路編集プログラム（edit_maze.py）を用いて課題 1-4 で作成された迷路をスタートからゴールまで 2 種類の経路が存在するように編集し，保存せよ．

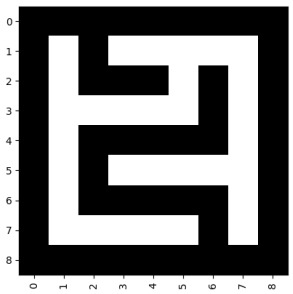


図 1　迷路（9 × 9 サイズ）の一例

```
1 1 1 1 1 1 1 1 1
1 0 1 0 0 0 0 0 1
1 0 1 1 1 0 1 0 1
1 0 0 0 0 0 1 0 1
1 0 1 1 1 1 1 0 1
1 0 1 0 0 0 0 0 1
1 0 1 1 1 1 1 0 1
1 0 0 0 0 0 1 0 1
1 1 1 1 1 1 1 1 1
```

図 2　迷路（9 × 9 サイズ）のテキスト表現

# 2　状態空間と基本的な探索（教科書 第 2 章）

探索問題は初期状態から目標状態へ至る行動の系列を求めることである．状態空間中のすべての状態を探索する最も基本的な探索手法が深さ優先探索と幅優先探索である．深さ優先探索，幅優先探索をそれぞれ実装せよ．探索を行うためには，これから探索しようとする候補の状態の集合を**オープンリスト** (open list)，探索し終えた状態の集合を**クローズドリスト** (closed list) を管理することが重要である（教科書 p.24 図 2.9 参照）．

## 2.1　深さ優先探索 (DFS, Depth-First Search)

深さ優先探索は縦型探索とも呼ばれる．データ構造の**スタック** (stack) と深い関係にあり，探索に用いるオープンリストをスタックで表現することで自然と実現できる．

深さ優先探索のアルゴリズム

1. 初期状態をオープンリストに入れる．クローズドリストを空に初期化する．
2. **while** オープンリストが空ではない． **do**
3. 　オープンリストから先頭の要素 $s$ を取り出す．クローズドリストに $s$ を追加する（$s$ を探査することに相当）．
4. 　$s$ が目標状態ならば，解は発見されたとして探索を終了．
5. 　$s$ から接続していてまだ追加していない状態をすべてオープンリストの**先頭**に追加する（**スタック**に**プッシュ**する）．
6. **end while** 探索を終了．

## 2.2　幅優先探索 (BFS, Breadth-First Search)

幅優先探索は横型探索とも呼ばれる．データ構造の**キュー** (queue) と深い関係にあり，探索に用いるオープンリストをキューで表現することで自然と実現できる．

幅優先探索のアルゴリズム

1. 初期状態をオープンリストに入れる．クローズドリストを空に初期化する．
2. **while** オープンリストが空ではない． **do**
3. 　オープンリストから先頭の要素 $s$ を取り出す．クローズドリストに $s$ を追加する（$s$ を探査することに相当）．
4. 　$s$ が目標状態ならば，解は発見されたとして探索を終了．
5. 　$s$ から接続していてまだ追加していない状態をすべてオープンリストの**末尾**に追加する（**キュー**に**エンキュー**する）．
6. **end while** 探索を終了．

## 2.3 課題

チェックポイント

- オープンリストとクローズドリストの管理ができているか.
- 深さ優先探索において，スタックの処理ができているか.
- 幅優先探索において，キューの処理ができているか.

**必須課題** 2-1　各通路上のセル（迷路上の 0 の部分）を経路探索における「状態」として，スタートからゴールへの経路を深さ優先探索するプログラムを作成し，実行結果を出力せよ．迷路の左上をスタート，迷路の右下をゴールとする．迷路の大きさは $15 \times 15$ サイズとする．このとき，各状態は迷路を表現する二次元配列の縦と横の添字のタプルで表現できる．また，オープンリストとクローズドリストの実装には，Python のリストをそのまま用いるとよい.

**オプション課題** 2-2　課題 2-1 と同条件の探索課題を幅優先探索で実現せよ.

# 3　最適経路の探索（教科書 第 3 章）

最適経路を探索するためのアルゴリズム（最適探索，最良優先探索，A*アルゴリズム）を実装する．最適経路の探索問題では，グラフの辺にコストを定義し，そのゴールまでの和を最小化する問題を考える．

探索アルゴリズムの実装に必要な関数記号を表 1 に示す．状態の予測評価値 $\hat{h}(s)$ とは，各状態 $s$ の持つ「ゴールまでの予測される遠さ」である．予測評価値は実際に辺を移動することでかかるコストから計算されるのではなく，発見的な知識として外部から与えられるものである．

表 1　最適経路探索に用いる関数の定義

| 関数記号 | 説明 |
|---|---|
| $g(s)$ | 初期状態から状態 $s$ までの最適経路上のコストの総和 |
| $h(s)$ | 状態 $s$ からゴールまでの最適経路上のコストの総和 |
| $f(s)$ | $s$ を経由した場合の最適経路のコスト $(f(s) = h(s) + g(s))$ |
| $\hat{h}(s)$ | $h(s)$ の推定値，状態 $s$ の予想評価値として用いる |
| $\hat{g}(s)$ | $g(s)$ の推定値 |
| $\hat{f}(s)$ | $f(s)$ の推定値，$(\hat{f}(s) = \hat{h}(s) + \hat{g}(s))$ |

## 3.1　最適探索 (optimal search)

ヒューリスティックな知識（予測評価値）を用いず，コストの和を最小化する最適経路を確実に発見するための手法が最適探索である．

最適探索のアルゴリズム

1. 初期状態のコスト値を 0 としてオープンリストに追加する．クローズドリストを空に初期化する．
2. **while** オープンリストが空ではない． **do**
3. 　オープンリストから先頭の要素 $s$ を取り出す．クローズドリストに $s$ を追加する．
4. 　$s$ が目標状態ならば，解は発見されたとして探索を終了．
5. 　$s$ から接続していてまだ探査していない状態をすべてオープンリストに追加する．オープンリスト内の状態の累積コストの推定値 $\hat{g}(s)$ を再計算し，**累積コストの推定値が小さい順に並べ替える**．
6. **end while** 探索を終了．

## 3.2 最良優先探索 (best-first search)

ヒューリスティックな知識としての予想評価値を頼りに探索を進めるのが最良優先探索である．

最良優先探索のアルゴリズム

1. 初期状態のコスト値を 0 としてオープンリストに追加する．クローズドリストを空に初期化する．
2. **while** オープンリストが空ではない． **do**
3. 　オープンリストから先頭の要素 $s$ を取り出す．クローズドリストに $s$ を追加する．
4. 　$s$ が目標状態ならば，解は発見されたとして探索を終了．
5. 　$s$ から接続していてまだ探査していない状態をすべてオープンリストに追加する．オープンリスト内の状態を，**予測評価値 $\hat{h}(s)$ が小さい順に並べ替える**．
6. **end while** 探索を終了．

## 3.3 A*アルゴリズム (A-star algorithm)

現在の状態までにかかった累積コストの推定値 $\hat{g}(s)$ と，ゴールまでに将来かかるであろう予測評価値 $\hat{h}(s)$ の二つをバランスよく用いて，探索を効率化する手法が A*アルゴリズムである．探索中に，よりよい経路が見つかればクローズドリストからオープンリストに状態を戻す操作が存在することに注意すること．

A*アルゴリズム

1. 初期状態のコスト値を 0 としてオープンリストに追加する．クローズドリストを空に初期化する．
2. **while** オープンリストが空ではない． **do**
3. 　オープンリストから先頭の要素 $s$ を取り出す．クローズドリストに $s$ を追加する．
4. 　$s$ が目標状態ならば，解は発見されたとして探索を終了．
5. 　$s$ から接続されているすべての状態 $s'$ に関して累積コストの推定値 $\hat{g}(s')$ と予測評価値 $\hat{h}(s')$ から $\hat{f}(s')$ を計算する．
6. 　5. の状態のうちで，オープンリストにもクローズドリストにも含まれていないものは，オープンリストに加える．
7. 　5. の状態のうちで，オープンリストかクローズドリストに含まれていたものについては，既に入っているものより $\hat{f}(s)$ の値が小さければ，**元のものを消去し，新しいものをオープンリストに追加する**．
8. オープンリスト内の状態を，**累積コストの推定値と予測評価値の和 $\hat{f}(s)$ が小さい順に並べ替える**．
9. **end while** 探索を終了．

## 3.4 課題

最適探索のチェックポイント

- 累積コストの推定値 $\hat{g}(s)$ の再計算を正しく実装できているか.
- 累積コストの推定値の小さい順にソートできているか.

最良優先探索のチェックポイント

- 予測評価値の値を適切に設定できているか.
- 予測評価値の小さい順にソートできているか.

A*アルゴリズムのチェックポイント

- $\hat{f}(s)$ が正しく計算できているか.
- $\hat{f}(s)$ の小さい順にソートできているか.
- アルゴリズム 7 行目の処理を正しく実装できているか.

**必須課題** 3-1　各通路上のセル（迷路上の 0 の部分）を経路探索における「状態」として，スタートからゴールへの最短経路を求める最適探索のプログラムを作成し，実行結果を出力せよ．迷路の左上をスタート，迷路の右下をゴールとする．迷路の大きさは $15 \times 15$ サイズとする．隣り合うセルに移動するときのコストは 1 であるとする.

**オプション課題** 3-2　課題 3-1 と同条件の経路探索課題を最良優先探索で実現せよ．予測評価値 $\hat{h}(s)$ は現在位置からゴールまでの壁が存在しなかった場合での到達に必要な移動回数（マンハッタン距離）とする.

**オプション課題** 3-3　課題 3-2 と同条件の経路探索課題を A*アルゴリズムで実現せよ.

**オプション課題** 3-4　スタートからゴールまで複数の経路が存在する迷路を作り，最適探索，最良優先探索，A*アルゴリズムで求まる経路がすべて異なるようにせよ．この際，隣り合うセル間の移動コストが異なる部分を設定してもよい.

発展

- 迷路の形状を変化させると結果がどう変化するか確認してみよう.
- コストや予測評価値を変化させると結果がどう変化するか確認してみよう.

# 4 強化学習（教科書 第 7 章）

強化学習 (reinforcement learning) のアルゴリズムの一つである Q 学習を実装する．

## 4.1 Q 学習 (Q-learning)

最適行動価値関数 $Q^*(s,a)$ の Q 値を推定することで強化学習を実現する学習を Q 学習と呼ぶ．Q 学習では，まずすべての $(s,a)$ の組み合わせに対して $Q(s,a)$ の値を記録するテーブルを用意する．この値を徐々に更新して真の Q 値である $Q^*(s,a)$ に近づけることを目指す．Q 学習の更新式を式 (1) に示す．ここで，$\gamma$ $(0 \leq \gamma < 1)$ は割引率 (discount rate)，$\alpha$ $(0 < \alpha \leq 1)$ は学習率である．

$$Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) + \alpha \underbrace{\left(r_{t+1} + \gamma \max_{a_{t+1} \in A} Q\left(s_{t+1}, a_{t+1}\right) - Q\left(s_t, a_t\right)\right)}_{\text{TD 誤差}} \tag{1}$$

Q 学習のアルゴリズム

1. Q 値を初期化する．
2. **for** $i = 1$ to $L$ **do**
3. 時刻 $t = 1$ として，$s_0$ を観測する．
4. **repeat**
5. 方策 $\pi$ に従って行動 $a_t$ を選択して移動する．
6. 環境から報酬 $r_{t+1}$ と状態 $s_{t+1}$ を観測する．
7. Q 学習の更新式に従って $Q(s_t, a_t)$ の値を更新する．
8. 時刻 $t \leftarrow t + 1$ とする．
9. **until** ゴールに到達する．もしくは，終了条件に達する．
10. **end for**

## 4.2 方策 (policy)

Q 学習では Q 値を学習していくが，方策 $\pi$ の決定方法は別途決定する必要がある．

**グリーディ法** (greedy method)　最も Q 値の高い行動を必ず選択する．

**ランダム法** (random method)　あらゆる行動を等確率で選択する．

**$\epsilon$-グリーディ法** (epsilon-greedy method)　確率 $\epsilon$ で全行動からランダムによる行動選択を行い，確率 $1 - \epsilon$ でグリーディ法による行動選択を行う．

**ボルツマン選択** (Boltzmann exploration policy)　ボルツマン分布と呼ばれる確率分布を行動価値関数を用いてつくることで方策とする．

## 4.3 課題

チェックポイント

- 確率的な状態遷移を実装できているか.
- TD 誤差を正しく計算できているか.
- 方策を適切に実装できているか.

**必須課題 4-1**　迷路の上でエージェント（ホイールダック 2 号）が { 停止，右，左，上，下 } の五種類の行動をランダムに選び，移動することで，報酬を観測しながら，ゴールに辿り着くプログラムを作成せよ．壁にぶつかる方向に移動した場合は $-1$ の報酬が得られ，ゴールに辿り着いた際には $+10$ の報酬が得られるものとする．それ以外の場合の報酬は 0 とする．

状態遷移についてはホイールダック 2 号が移動しようとした方向に確率 0.8 で移動できるものとする（停止の場合はその場に留まる）．残りの 0.2 の確率でホイールダック 2 号は移動できず，その場に留まるものとする．移動しようとした際に壁があれば，それ以上は進まず止まるものとする（教科書 p.113 8.4 節の条件と同一）.

**必須課題 4-2**　$Q(s,a)$ を記録するテーブルを準備し，Q-learning のアルゴリズムに基づき，更新するプログラムを作れ．価値関数 $V(s) = \max_a Q(s,a)$ をプロットし，スタートからゴールに向かって徐々に値が大きくなっていることを確認せよ．迷路の大きさは $9 \times 9$ 程度で構わない．学習の試行回数（トライアル数）は $L = 100$ とし，割引率 $\gamma = 0.9$，学習率 $\alpha = 0.1$ として実装せよ．また，Q 値の初期値は任意の値で良い（一般的には，すべての要素の値を 0.0 として初期化する）．方策は，グリーディ法でもランダム法でもどちらでも構わない.

**オプション課題 4-3**　Q-learning を実装し，$\epsilon$-グリーディ法で，ゴールへ向かう方策が実現されることを確認せよ.

**オプション課題 4-4**　Q-learning を実装し，ボルツマン選択で，ゴールへ向かう方策が実現されることを確認せよ.

発展

- $\alpha$ や $\gamma$ の値を変更すると結果がどう変化するか確認してみよう.
- 報酬の値を変更すると結果がどう変化するか確認してみよう.
- 方策を変更すると結果がどう変化するか確認してみよう.

# 5　位置推定（ベイズフィルタ）　（教科書 第 8 章）

ベイズフィルタによるロボットの自己位置推定 (self-localization) アルゴリズムを実装する．

## 5.1　ベイズフィルタ (Bayes filter)

自らの得た観測（センサ情報）$o_{1:t}$ と，自らが行ってきた行動 $a_{1:t}$ から自己位置 $s_t$ を推定する手法として最も基本的な手法がベイズフィルタである．$F_0(s_0)$ の初期値は**無情報** (uniformative) であり，すべての場所にそれぞれ等確率で存在するとする．$T$ は最大ステップ数である．

ベイズフィルタのアルゴリズム

1. $F_0(s_0)$ を初期化する．$F_0(s_0) = P(s_0)$
2. **for** $t = 1$ to $T$ **do**
3. 　$a_{t-1}$ で移動し，$o_t$ を観測する．
4. 　すべての $s_t$ に対して下記の $G_t$ を計算する．

$$G_t\left(s_t\right) \leftarrow P\left(o_t|s_t\right) \sum_{s_{t-1}} P\left(s_t|s_{t-1}, a_{t-1}\right) F_{t-1}\left(s_{t-1}\right) \tag{2}$$

5. 　$F_t\left(s_t\right) \leftarrow G_t\left(s_t\right) / \sum_s G_t(s)$
6. **end for**

## 5.2　課題

ベイズフィルタのチェックポイント

- 確率的な状態遷移を実装できているか．
- 観測，センサ情報の計算の処理を実装できているか．
- すべての状態に対して確率値の正規化を行えているか．

**準備**　教科書 p.113 の 8.4 節の例と同様の横幅 5 マスの一本道の通路環境（一次元）を作成せよ．実装では周囲の壁も 1 マスに含まれるため，高さ 3・幅 7 の大きさの通路を作成すればよい．

**必須課題 5-1**　教科書の例と同様の横幅 5 マスの一本道においてベイズフィルタを実装せよ．無情報の状態で中央を初期位置とし，右，右，左と移動できたときの各時刻での各セルにエージェント（ホイールダック 2 号）が存在する確率を出力せよ．プログラムは任意の幅の一本道の迷路に適用できなければならない．

状態遷移については，ホイールダック 2 号が移動しようとした方向に確率 0.8 で移動できるものとする．残りの 0.2 の確率でホイールダック 2 号は移動できず，その場に留まるものとする．移動しようとした際に壁があれば，それ以上は進まず止まるものとする．観測は教科書の

図 8.2 の 16 通りの観測の中から一つが得られる．0.7 の確率で正しい観測が得られ，0.3 の確率で誤った観測結果が得られるものとする．誤った場合の観測結果は，15 通りの中から等確率で発生するものとする．

**オプション課題 5-2**　ホイールダック 2 号をキーボード入力で操作できるようにして，課題 5-1 のプログラムをインタラクティブにせよ．このとき，ロボットは課題 5-1 の状態遷移の確率に従い移動するものとする．

**オプション課題 5-3**　任意の形状の二次元の迷路に対応できるようにベイズフィルタを拡張して，実装せよ．

# 6 位置推定（粒子フィルタ） （教科書 第 9 章）

粒子フィルタ（パーティクルフィルタ）によるロボットの自己位置推定 (self-localization) アルゴリズムを実装する．迷路の形状や状態遷移，観測に関する条件はベイズフィルタの際と同じとする．

## 6.1 粒子フィルタ (particle filter, パーティクルフィルタ)

粒子フィルタはベイズフィルタを基にしつつ，モンテカルロ近似とベイズフィルタの更新式における Sampling Importance Resampling (SIR) を導入することで実現されるベイズフィルタの近似手法である．各粒子を状態遷移確率に従って移動させた後に，各粒子について観測確率で重みを付けてリサンプリングするだけの非常に単純なアルゴリズムになっている．

粒子フィルタのアルゴリズム

1. 粒子の分布を初期化させる．$t \leftarrow 1$
2. **repeat**
3. ロボットの行動 $a_{t-1}$ に従い，粒子 $s_{t-1}^{(i)}$ $(1 \leq i \leq N)$ ごとに次状態 $s_t^{(i)}$ を状態遷移確率を用いてサンプリングする．

$$\overline{s}_t^{(i)} \sim P\left(s_t \mid s_{t-1} = s_{t-1}^{(i)}, a_{t-1}\right) \tag{3}$$

4. $o_t$ を観測し，各粒子についてセンサ情報の観測確率 $w_i = P(o_t \mid \overline{s}_t^{(i)})$ を計算し，それぞれの粒子の重みとする．
5. 粒子の重みの比率 $\dfrac{w_i}{\sum_j w_j}$ に従って，粒子をリサンプリングする．これらを $s_t^{(k)}$ $(1 \leq i \leq N)$ とする．

$$s_t^{(k)} = \sum_i w_i \, \delta\left(s_t, \overline{s}_t^{(i)}\right) \tag{4}$$

6. $t \leftarrow t + 1$
7. **until** 停止させられるまで．

## 6.2 課題

粒子フィルタのチェックポイント

- 粒子ごとに状態遷移確率を用いてサンプリングできているか．
- 粒子の重みの比率に従って，粒子のリサンプリングができているか．

**必須課題 6-1** 教科書の例と同様の横幅 5 マスの一本道において粒子フィルタを実装せよ．無情報の状態で中央を初期位置とし，右，右，左と移動できたときの各時刻での各セルにパーティクル

の数を表示し，また，エージェント（ホイールダック 2 号）が存在する確率を出力せよ．プログラムは任意の幅の一本道の迷路に適用できなければならない．迷路の形状や状態遷移，観測に関する条件はベイズフィルタの際と同じとする．粒子の数は $N = 10$ として実装せよ．

**オプション課題** 6-2　ホイールダック 2 号をキーボード入力で操作できるようにして，課題 6-1 のプログラムをインタラクティブにせよ．このとき，ロボットは課題 5-1 の状態遷移の確率に従い移動するものとする．

**オプション課題** 6-3　任意の形状の二次元の迷路に対応できるように粒子フィルタを拡張して，実装せよ．

発展

- 粒子（パーティクル）の量を変化させると結果がどう変化するか確認してみよう．
- ベイズフィルタと粒子フィルタで推定結果にどのような違いがでるか確認してみよう．
- 壁の無い広い空間で自己位置推定した場合，推定結果はどうなるか確認してみよう．