

知能科学：確率的手法

平井 慎一

立命館大学 ロボティクス学科

講義の流れ

- 1 乱数
- 2 モンテカルロ法
- 3 移動ロボットの経路計画
- 4 まとめ

乱数

コンピュータ上でサイコロを模擬する.

$$D = \begin{cases} 1 & X \in (0, 1/6) \\ 2 & X \in [1/6, 2/6) \\ 3 & X \in [2/6, 3/6) \\ 4 & X \in [3/6, 4/6) \\ 5 & X \in [4/6, 5/6) \\ 6 & X \in [5/6, 1) \end{cases}$$

X は区間 $(0, 1)$ 内の一様乱数

$$X \sim U(0, 1)$$

一様乱数 $U(0, 1)$

関数 rand 区間 $(0, 1)$ 内の一様乱数

プログラム uniform.m

```
for i=1:10
    x = rand;
    s = num2str(x);
    disp(s);
end
```

uniform.m

```
>> uniform
```

```
0.81472
```

```
0.90579
```

```
0.12699
```

```
0.91338
```

```
0.63236
```

```
0.09754
```

```
0.2785
```

```
0.54688
```

```
0.95751
```

```
0.96489
```

```
>> uniform
```

```
0.15761
```

```
0.97059
```

```
0.95717
```

```
0.48538
```

dice.m

サイコロのシミュレーション

```
function [k] = dice()
%   simulating a dice
    x = rand;
    if x < 1/6.00           k = 1;
    elseif x < 2/6.00      k = 2;
    elseif x < 3/6.00      k = 3;
    elseif x < 4/6.00      k = 4;
    elseif x < 5/6.00      k = 5;
    else                    k = 6;
    end
end
```

dice_run.m

サイコロを 100 回振る

```
for i=1:10
    for j=1:10
        s(j) = dice();
    end
    disp(join(string(s), ' '));
end
```

dice_run.m

```
>> dice_run
```

```
5 6 4 4 5 6 2 4 2 2
```

```
2 6 2 4 5 2 3 6 1 5
```

```
6 2 5 6 1 6 6 6 2 3
```

```
6 6 5 1 1 3 2 4 1 4
```

```
3 4 1 4 4 2 3 4 4 1
```

```
5 1 3 2 6 1 5 3 1 4
```

```
2 1 1 1 6 5 2 2 1 4
```

```
5 2 5 5 6 2 6 6 1 5
```

```
1 1 6 6 5 1 2 4 2 5
```

```
2 3 2 6 2 4 5 6 1 3
```

```
>> dice_run
```

```
5 5 2 1 2 4 2 6 5 3
```

```
3 4 2 2 6 3 4 2 5 1
```

```
6 5 6 4 5 4 6 5 2 5
```

```
2 6 6 5 2 5 1 1 6 3
```


dice_validation.m

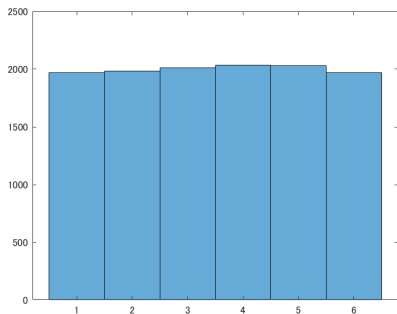
```
n = 12000;

for k=1:n
    a(k) = dice();
end

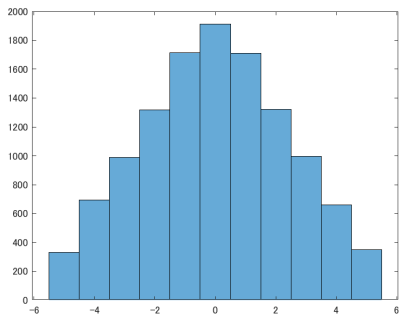
histogram(a);
fprintf("サイコロの目の頻度\n");
fprintf("一時停止：何かのキーを押してください\n");
pause;

d = a(2:n) - a(1:n-1);
histogram(d);
fprintf("差の頻度\n");
fprintf("一時停止：何かのキーを押してください\n");
pause;
```

dice_validation.m



ヒストグラム

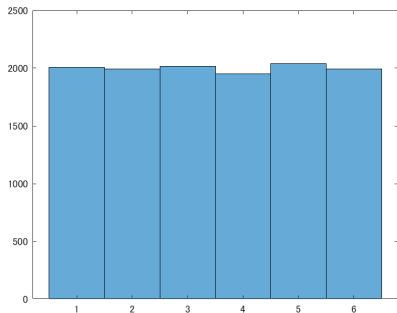


差のヒストグラム

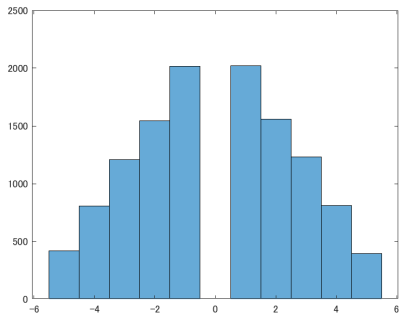
dice_false_run.m

```
>> dice_false_run
3 2 5 3 4 5 1 5 6 5
2 5 3 6 4 3 4 3 5 6
3 6 3 4 2 1 2 4 3 4
6 4 2 5 4 3 4 2 3 1
2 6 3 2 5 6 1 3 1 6
5 6 5 2 4 5 4 5 3 5
2 4 1 2 1 4 6 3 4 3
5 6 5 1 6 1 4 6 4 6
2 6 2 1 4 2 4 3 4 3
4 3 5 3 5 1 2 1 2 6
>> dice_false_run
4 1 6 4 2 3 6 5 2 1
6 2 3 1 4 6 5 3 2 3
5 1 3 1 2 6 4 2 1 4
1 4 3 4 5 2 3 4 1 6
```

dice_false_validation.m



ヒストグラム



差のヒストグラム

一様乱数 $U(a, b)$

$U(a, b)$

区間 (a, b) 内の一様乱数

$$X \sim U(0, 1)$$

$$Y \sim U(a, b)$$

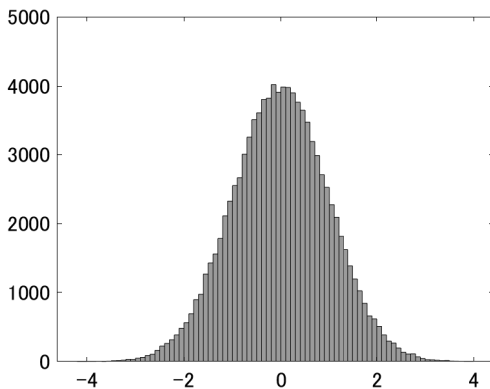
$$Y = (b - a)X + a$$

正規乱数 $N(0, 1)$

関数 randn

平均 0, 標準偏差 1 の正規乱数

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$



正規乱数 $N(\mu, \sigma)$

$N(\mu, \sigma)$

平均 μ , 標準偏差 σ (分散 σ^2) の正規分布に従う乱数

$$X \sim N(0, 1)$$

$$Y \sim N(\mu, \sigma)$$

$$Y = \sigma X + \mu$$

モンテカルロ法

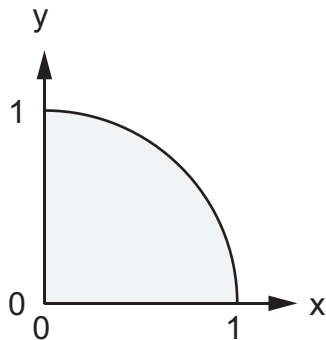
計算に時間や手間を要する問題を，乱数を用いて近似的に解く手法

定積分の計算

定積分

$$S_1 = \int_0^1 \sqrt{1-x^2} dx$$

被積分関数 $y = \sqrt{1-x^2}$

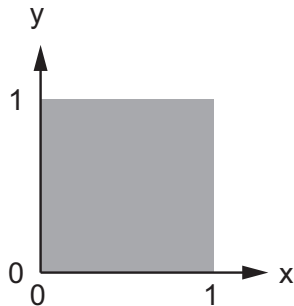


定積分の計算

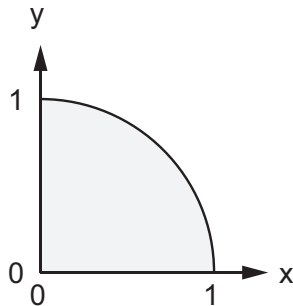
$$x \in [0, 1], \quad y \in [0, 1]$$

点 (x, y) を正方形領域内でランダムに発生させる。

$$x \sim U(0, 1), \quad y \sim U(0, 1)$$



N : 発生させた点の個数



M : $y \leq \sqrt{1 - x^2}$ を満たす
点の個数

定積分の計算

点の個数は、面積に比例すると仮定
正方形領域の面積：1

$$N : M = 1 : S_1$$

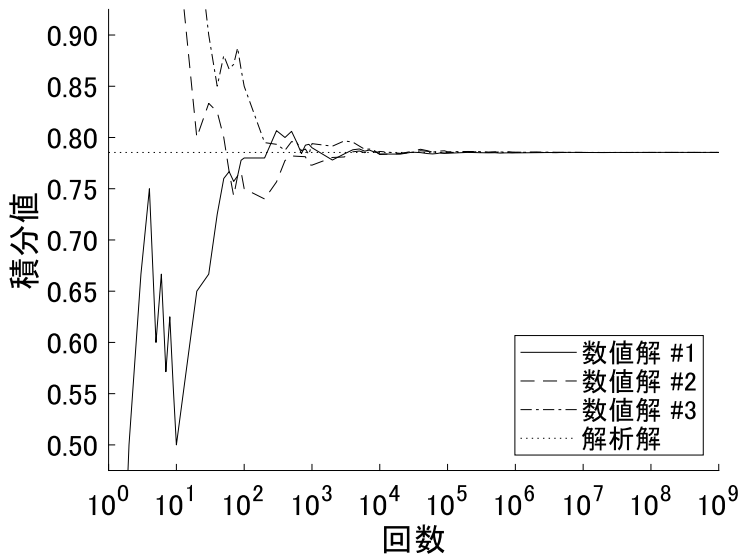
⇓

$$S_1 = \frac{M}{N}$$

integral_S1.m

```
N = 1000000000; M = 0; k = 1; step = 10;
for i=1:N
    x = rand;    y = rand;
    if x*x + y*y <= 1
        M = M+1;
    end
    if rem(i,step) == 0
        s = [num2str(i), ' ', num2str(M/i)];
        disp(s); k = k+1;
    end
    if k == 10
        step = step * 10; k = 1;
    end
end
end
```

定積分の計算



多重定積分の計算

多重定積分

$$S_n = \int \int \cdots \int_{D_n} \sqrt{1 - x_1^2 - x_2^2 - \cdots - x_n^2} \, dx_1 \, dx_2 \cdots dx_n$$

積分領域

$$D_n = \{(x_1, x_2, \dots, x_n) \mid x_1^2 + x_2^2 + \cdots + x_n^2 \leq 1, \\ x_1, x_2, \dots, x_n \geq 0\}$$

被積分関数

$$y = \sqrt{1 - x_1^2 - x_2^2 - \cdots - x_n^2}$$

多重定積分の計算

$$x_1, x_2, \dots, x_n \in [0, 1], \quad y \in [0, 1]$$

点 $(x_1, x_2, \dots, x_n, y)$ をランダムに発生させる.

$$(0 \leq x_1, x_2, \dots, x_n, y \leq 1)$$

$$x_1 \sim U(0, 1), x_2 \sim U(0, 1), \dots, x_n \sim U(0, 1), y \sim U(0, 1)$$

N : 発生させた点の個数

M : $y \leq \sqrt{1 - x_1^2 - x_2^2 - \dots - x_n^2}$ を満たす点の個数

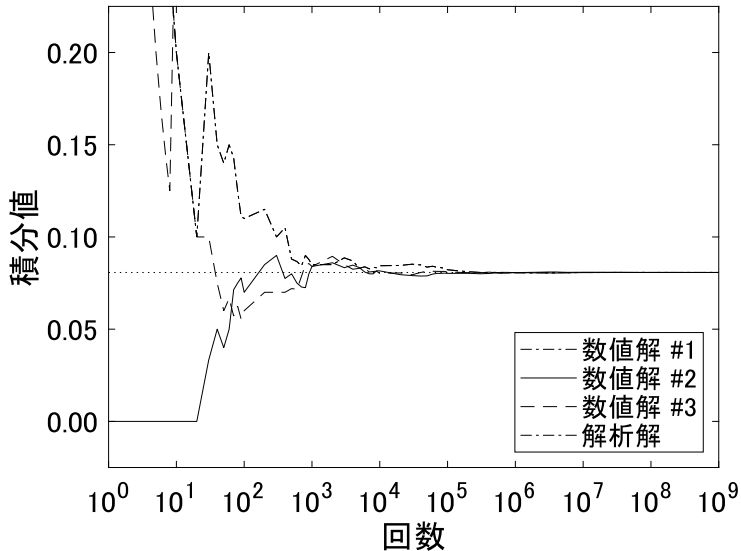
多重定積分の計算

$$M : N = S_n : 1$$

⇓

$$S_n = \frac{M}{N}$$

多重定積分 S_5 の計算



同じ誕生日

問題

n 人のグループで、誕生日が同じ人たちがいる確率を求める。

問題を簡単にするため、2月29日は考えない。

誕生日の確率分布は、一様であると仮定する。

同じ誕生日

問題

n 人のグループで、誕生日が同じ人たちがいる確率を求める。

問題を簡単にするため、2月29日は考えない。

誕生日の確率分布は、一様であると仮定する。

モンテカルロシミュレーションで解く。

- 1月1日, 1月2日, 1月3日, \dots , 12月31日に, 1, 2, 3, \dots , 365 を対応させる。
- 一様に値 1, 2, 3, \dots , 365 を取る乱数を, n 個発生させる。
- n 個の値の中に同じ値があるか否かを調べる。
- 以上の試行を複数回繰返し, 同じ値がある頻度を数える。

same_birthday_simulation.m

```
n = 23; same = 0; diff = 0;
for iteration=1:1000
    birthday = ceil(365*rand(1,n));
    found = 0;
    for k=1:n
        if ismember(birthday(k), birthday(k+1:n))
            found = 1;
            break;
        end
    end
    if found
        same = same + 1;
    else
        diff = diff + 1;
    end
end
```

同じ誕生日

$n = 5$ で、頻度を 10 回数えた結果の例

5 人	同じ人がいる	24 回	すべて異なる	976 回
5 人	同じ人がいる	21 回	すべて異なる	979 回
5 人	同じ人がいる	28 回	すべて異なる	972 回
5 人	同じ人がいる	39 回	すべて異なる	961 回
5 人	同じ人がいる	26 回	すべて異なる	974 回
5 人	同じ人がいる	24 回	すべて異なる	976 回
5 人	同じ人がいる	24 回	すべて異なる	976 回
5 人	同じ人がいる	27 回	すべて異なる	973 回
5 人	同じ人がいる	29 回	すべて異なる	971 回
5 人	同じ人がいる	33 回	すべて異なる	967 回

同じ誕生日

$n = 23$ で、頻度を 10 回数えた結果の例

23 人	同じ人がいる	493 回	すべて異なる	507 回
23 人	同じ人がいる	496 回	すべて異なる	504 回
23 人	同じ人がいる	511 回	すべて異なる	489 回
23 人	同じ人がいる	514 回	すべて異なる	486 回
23 人	同じ人がいる	476 回	すべて異なる	524 回
23 人	同じ人がいる	507 回	すべて異なる	493 回
23 人	同じ人がいる	530 回	すべて異なる	470 回
23 人	同じ人がいる	519 回	すべて異なる	481 回
23 人	同じ人がいる	501 回	すべて異なる	499 回
23 人	同じ人がいる	531 回	すべて異なる	469 回

同じ誕生日

$n = 30$ で、頻度を 10 回数えた結果の例

30 人	同じ人がいる	708 回	すべて異なる	292 回
30 人	同じ人がいる	684 回	すべて異なる	316 回
30 人	同じ人がいる	717 回	すべて異なる	283 回
30 人	同じ人がいる	712 回	すべて異なる	288 回
30 人	同じ人がいる	713 回	すべて異なる	287 回
30 人	同じ人がいる	708 回	すべて異なる	292 回
30 人	同じ人がいる	701 回	すべて異なる	299 回
30 人	同じ人がいる	690 回	すべて異なる	310 回
30 人	同じ人がいる	716 回	すべて異なる	284 回
30 人	同じ人がいる	724 回	すべて異なる	276 回

同じ誕生日

$n = 50$ で、頻度を 10 回数えた結果の例

50 人	同じ人がいる	974 回	すべて異なる	26 回
50 人	同じ人がいる	968 回	すべて異なる	32 回
50 人	同じ人がいる	972 回	すべて異なる	28 回
50 人	同じ人がいる	964 回	すべて異なる	36 回
50 人	同じ人がいる	967 回	すべて異なる	33 回
50 人	同じ人がいる	962 回	すべて異なる	38 回
50 人	同じ人がいる	970 回	すべて異なる	30 回
50 人	同じ人がいる	981 回	すべて異なる	19 回
50 人	同じ人がいる	967 回	すべて異なる	33 回
50 人	同じ人がいる	982 回	すべて異なる	18 回

同じ誕生日

2月29日を考慮

- ⇒ 2月29日に値366を対応させる
乱数が366となる確率は、他の値の確率の $1/4$

誕生日の確率分布を考慮

- ⇒ 確率分布を乱数の発生確率に反映させる



モンテカルロ法は、様々な条件に対応することができる

二人零和ゲーム

$G_A^* > 0$ の例

$$\begin{bmatrix} g_{PP} & g_{PQ} \\ g_{QP} & g_{QQ} \end{bmatrix} = \begin{bmatrix} +5 & -8 \\ -2 & +5 \end{bmatrix}, \quad \begin{bmatrix} x^* \\ y^* \end{bmatrix} = \begin{bmatrix} 0.35 \\ 0.65 \end{bmatrix}$$

$G_A^* = 0$ の例

$$\begin{bmatrix} g_{PP} & g_{PQ} \\ g_{QP} & g_{QQ} \end{bmatrix} = \begin{bmatrix} +4 & -8 \\ -2 & +4 \end{bmatrix}, \quad \begin{bmatrix} x^* \\ y^* \end{bmatrix} = \begin{bmatrix} 1/3 \\ 2/3 \end{bmatrix}$$

$G_A^* < 0$ の例

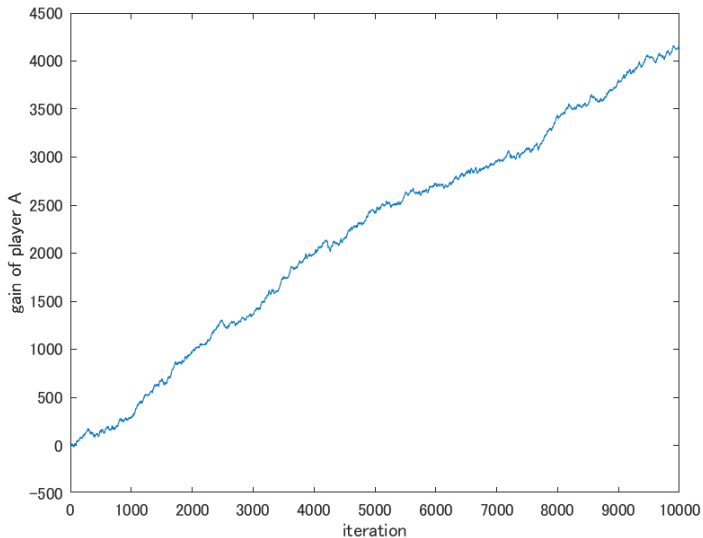
$$\begin{bmatrix} g_{PP} & g_{PQ} \\ g_{QP} & g_{QQ} \end{bmatrix} = \begin{bmatrix} +3 & -8 \\ -2 & +3 \end{bmatrix}, \quad \begin{bmatrix} x^* \\ y^* \end{bmatrix} = \begin{bmatrix} 5/16 \\ 11/16 \end{bmatrix}$$

zero_sum_game_simulation.m

```
g = [ 5, -8; -2, 5 ];  
x = 0.35; y = 0.65;  
  
gainA = 0; result = [];  
for iteration=1:10000  
    if rand < x A = 1; else A = 2; end  
    if rand < y B = 1; else B = 2; end  
    gainA = gainA + g(A,B);  
    result = [ result; iteration, gainA ];  
end  
  
plot(result(:,1), result(:,2));
```

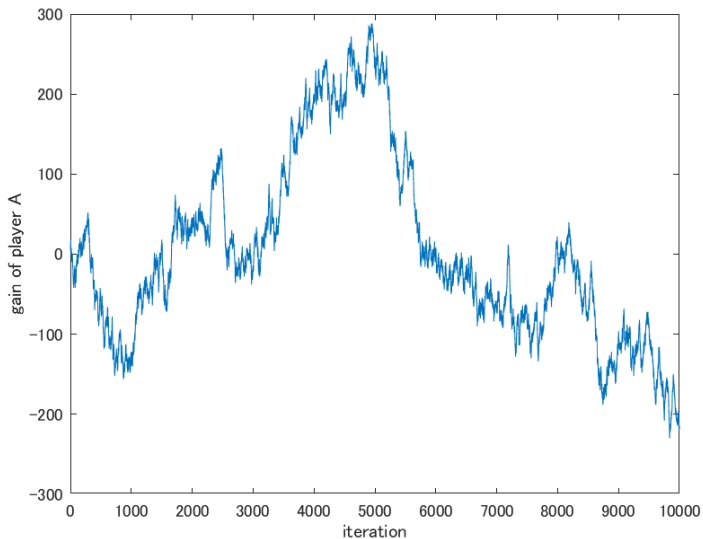
二人零和ゲーム

$G_A^* > 0$ の例



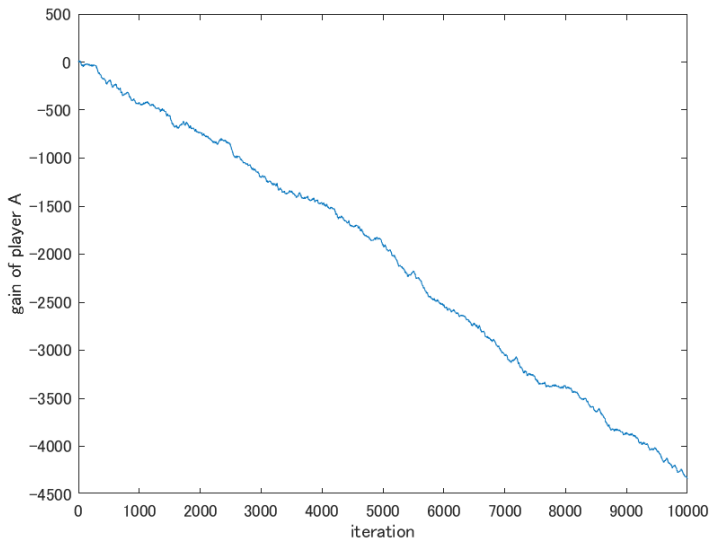
二人零和ゲーム

$G_A^* = 0$ の例

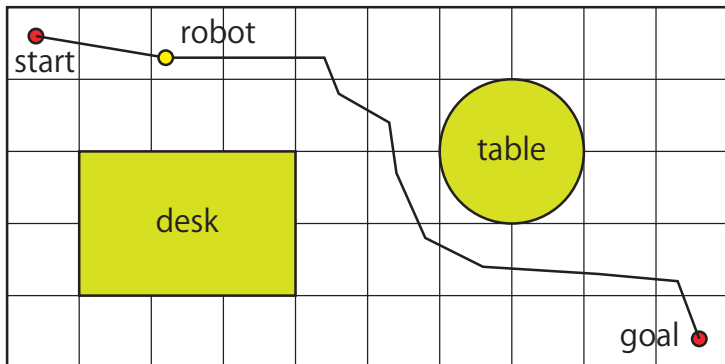


二人零和ゲーム

$G_A^* < 0$ の例

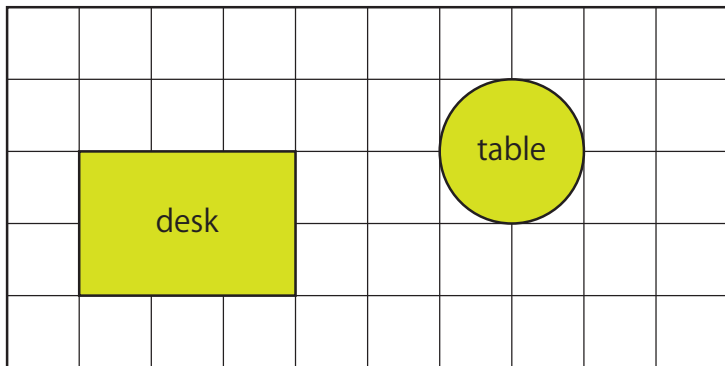


移動ロボットの経路計画



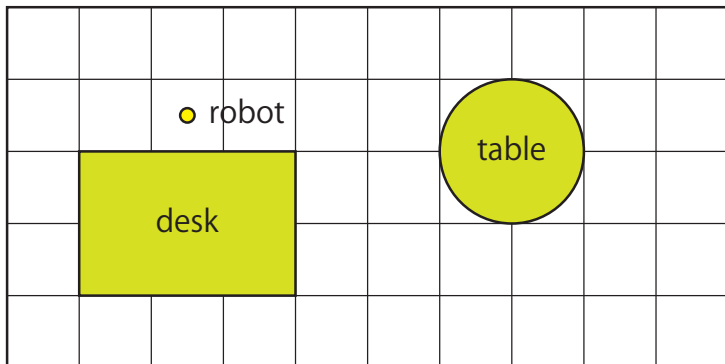
障害物（机，テーブル）と干渉や接触をすることなく
スタートからゴールに至る経路を見つける

移動ロボットの経路計画



部屋の大きさ： 横 10m 縦 5m
机 (長方形 横 3m 縦 2m)
テーブル (円形 半径 1m)

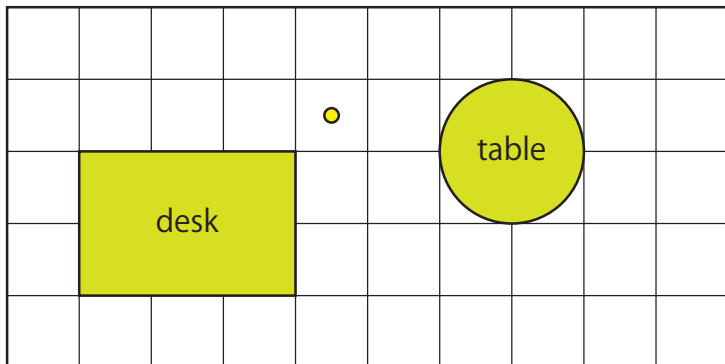
移動ロボットの経路計画



移動ロボット： 大きさを無視する → 点とみなす
部屋の左下角に原点を設定

→ 移動ロボットの位置 座標 (x, y)

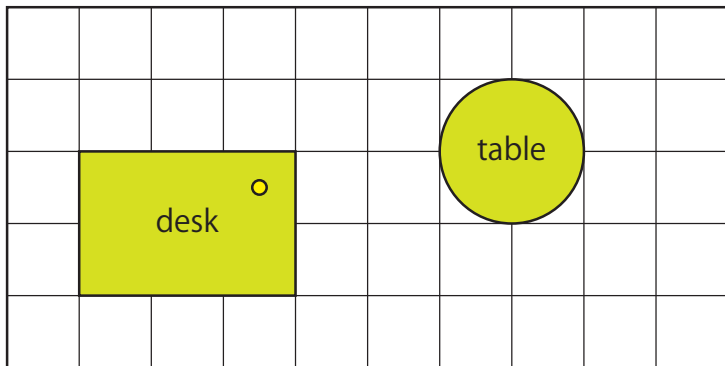
移動ロボットの経路計画



ロボットの位置 (4.50, 3.50)

自由 (free)

移動ロボットの経路計画

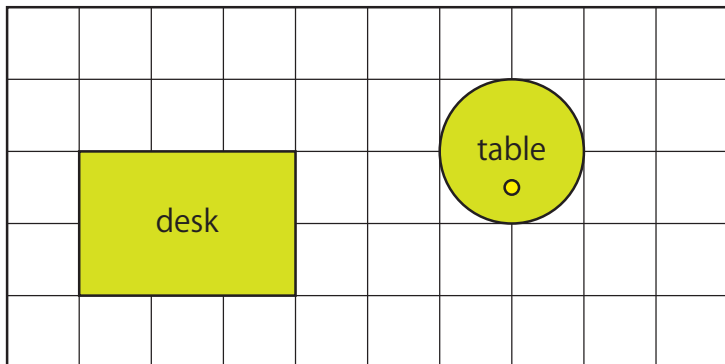


ロボットの位置 (3.50, 2.50)

干渉 (interfered)

→ 物理的に起こらない

移動ロボットの経路計画

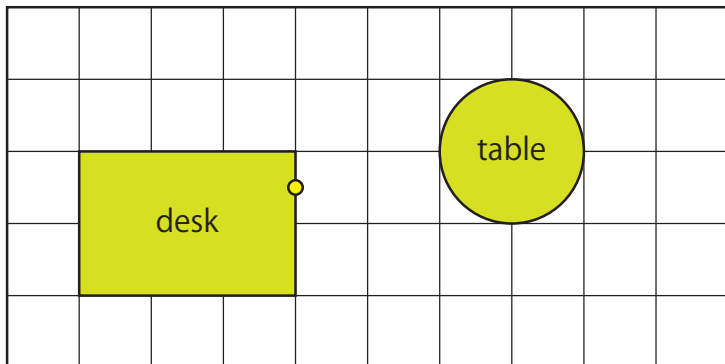


ロボットの位置 (7.00, 2.50)

干渉 (interfered)

→ 物理的に起こらない

移動ロボットの経路計画



ロボットの位置 (4.00, 2.50)

接触 (contact)

→ 起こりうるが望ましくない

干渉チェック

仮定：部屋や障害物の幾何情報はすべて既知

移動ロボットの座標 (x, y)

仮定： $0 \leq x \leq 10$ ならびに $0 \leq y \leq 5$

入力 (x, y) 出力：自由 or 干渉（接触を含む）

$1 \leq x \leq 4$ かつ $1 \leq y \leq 3$ ならば机と干渉

$(x - 7)^2 + (y - 3)^2 \leq 1^2$ ならばテーブルと干渉

それ以外ならば自由

干渉チェック

線分の干渉

点 P と点 Q とを結ぶ線分が自由



線分 PQ 上の任意の点で自由

短い線分の干渉チェック

点 P と点 Q が近い（距離が短い）



線分の有限個の内分点すべてで自由ならば
線分で自由と判定しても（実用上）良い

干渉チェック

線分 PQ 仮定：点 P と点 Q は自由

点 P の位置ベクトル x_P 点 Q の位置ベクトル x_Q

ϵ : 小さい正の定数 干渉チェックにおける点の間隔

短い線分の干渉チェック

初期化

$$d = \epsilon \frac{x_Q - x_P}{\|x_Q - x_P\|}, \quad n = \frac{\|x_Q - x_P\|}{\epsilon}$$

for $k = 1, 2, \dots, n$ {

もし $x_P + kd$ が干渉しているならば干渉を返す

}

自由を返す

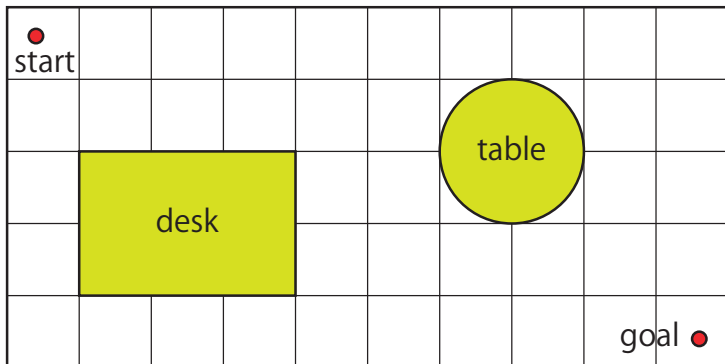
MATLAB: PRM_free_point_check.m

```
function status = PRM_free_point_check ( p )
    x = p(1); y = p(2);
    if ( 1 <= x && x <= 4 && 1 <= y && y <=3 )
        status = 0;
        return;
    end
    if ((x-7)^2 + (y-3)^2 <= 1^2)
        status = 0;
        return;
    end
    status = 1;
end
```

MATLAB: PRM_free_edge_check.m

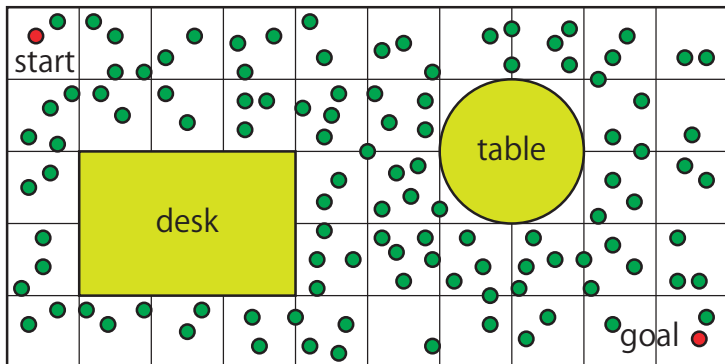
```
function status = PRM_free_edge_check ( ps, pe )
    eps = 1e-3;
    dse = pe - ps; ndse = norm(dse);
    d = eps*(dse/ndse);
    n = ndse/eps;
    for k=1:n
        p = ps + k*d;
        if PRM_free_point_check(p) == 0
            status = 0;
            return;
        end
    end
    status = 1;
end
```

PRM (Probabilistic Roadmaps)



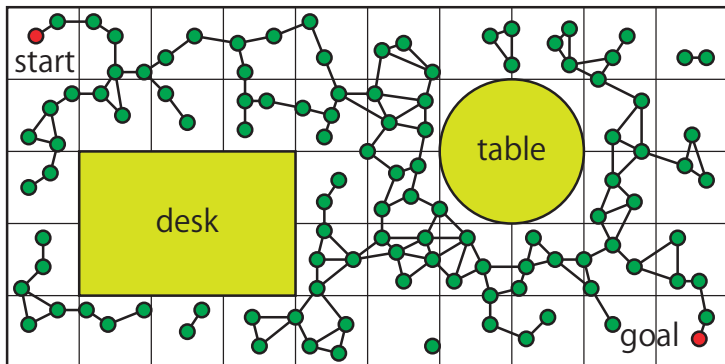
スタートとゴールの設定

PRM (Probabilistic Roadmaps)



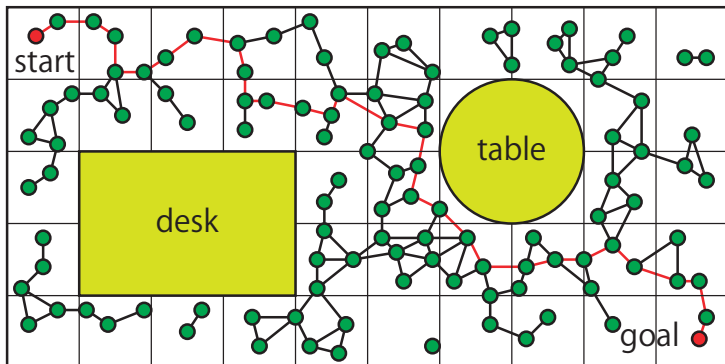
自由な点をランダムに生成

PRM (Probabilistic Roadmaps)



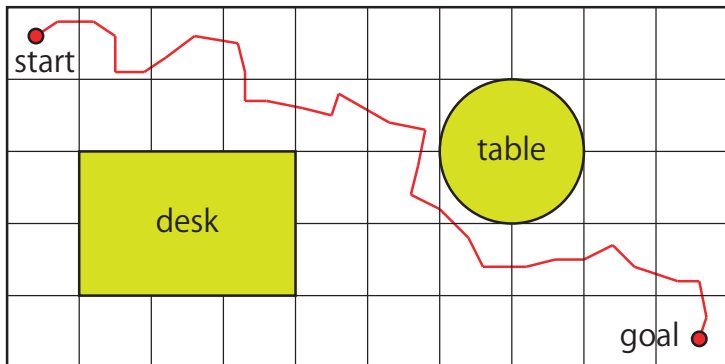
自由な短い線分をエッジで接続

PRM (Probabilistic Roadmaps)



スタートからゴールへ至る経路

PRM (Probabilistic Roadmaps)



結果

PRM (Probabilistic Roadmaps)

部屋内に自由な点を生成するアルゴリズム

集合 $V = \text{空集合}$

```
while (集合  $V$  の大きさが指定した個数未満) {  
    部屋内にランダムに点を生成  
    点が無ければ, 集合  $V$  に追加する  
}
```

サンプルプログラム

- PRM 自由な点を見つける
- PRM 自由な点か否かを判断

PRM (Probabilistic Roadmaps)

部屋内に自由な線分を生成するアルゴリズム

集合 $E = \text{空集合}$

```
for 集合  $V$  に含まれるすべての点  $P$  に対して {  
  for 集合  $V$  に含まれるすべての点  $Q$  に対して {  
    if 距離  $PQ$  が指定した値以下 {  
      線分  $PQ$  が自由ならば, 集合  $E$  に線分  $PQ$  を追加  
    }  
  }  
}
```

サンプルプログラム

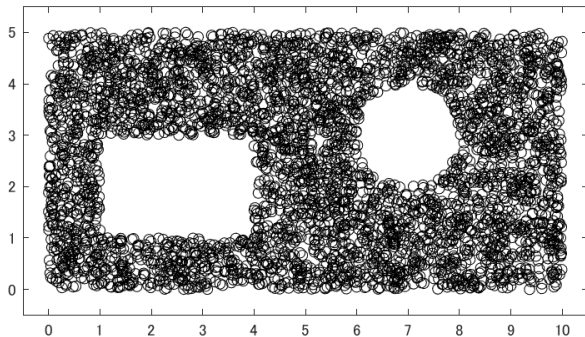
- PRM 自由な線分を見つける
- PRM 自由な線分か否かを判断

PRM (Probabilistic Roadmaps)

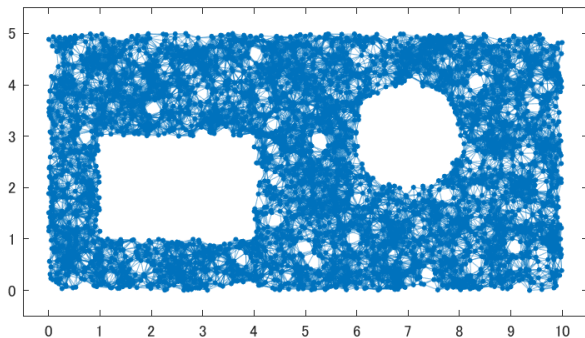
サンプルプログラム

- PRM グラフを作成する
- PRM 最短経路を見つける

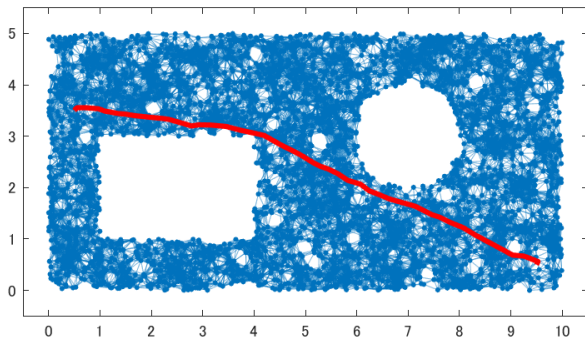
実行例（グラフを作成する）



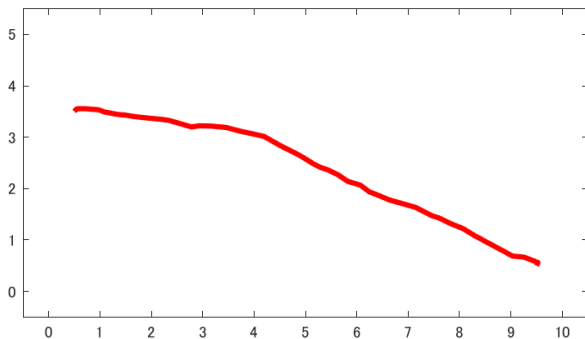
実行例（グラフを作成する）



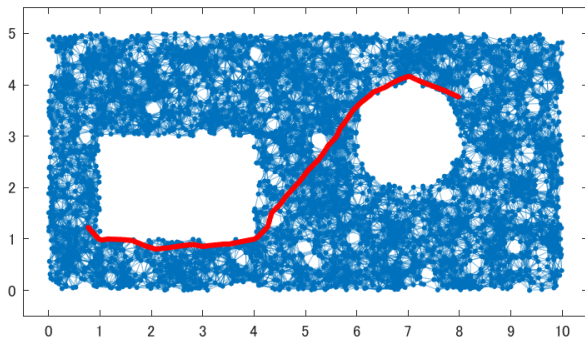
実行例（最短経路を見つける）



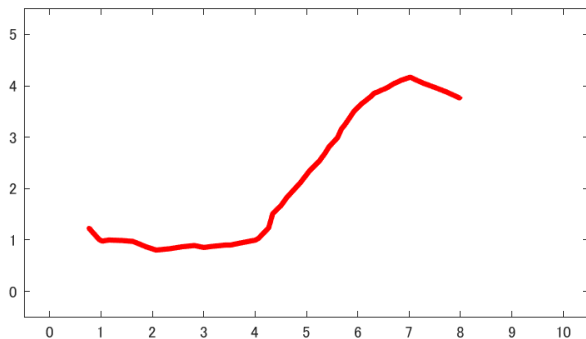
実行例（最短経路を見つける）



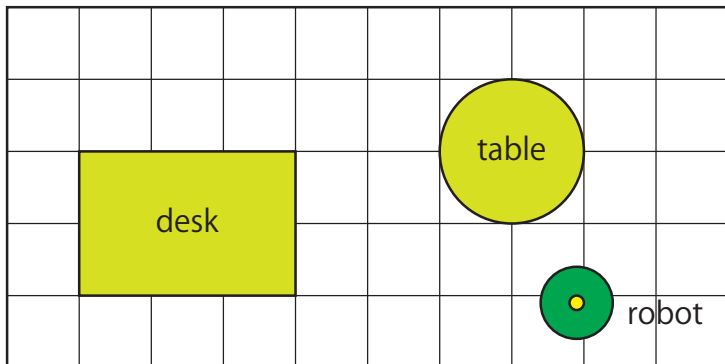
実行例（最短経路を見つける）



実行例（最短経路を見つける）



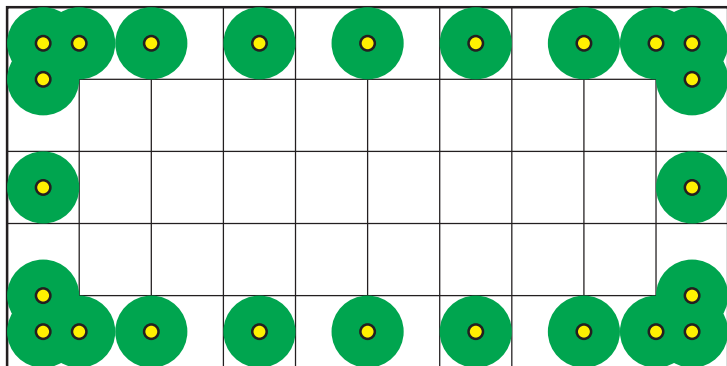
移動ロボットの経路計画



円形移動ロボット

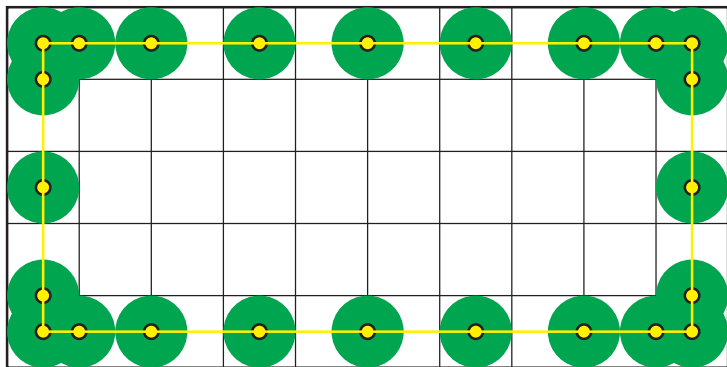
半径 0.5m 中心の座標 (x, y)

移動ロボットの経路計画



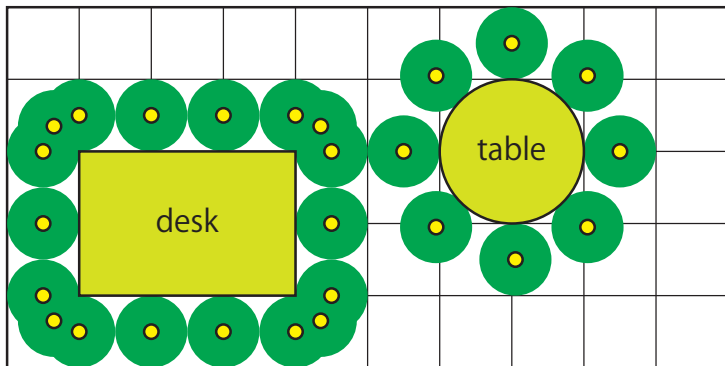
壁に沿ってロボットを動かす

移動ロボットの経路計画



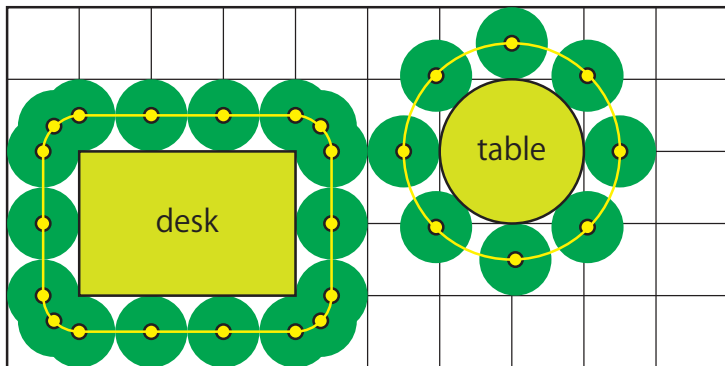
壁に沿ってロボットを動かす
ロボットの中心が描く軌跡

移動ロボットの経路計画



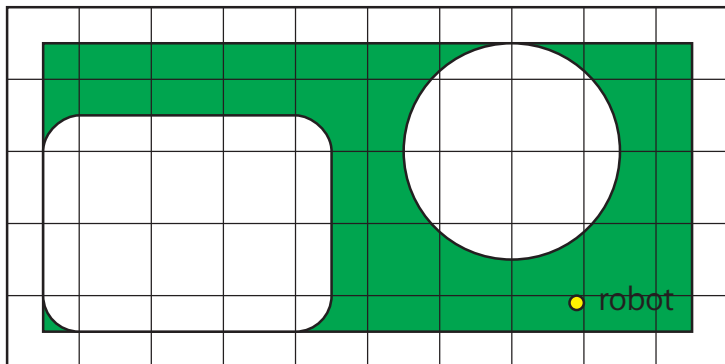
障害物（机，テーブル）に沿ってロボットを動かす

移動ロボットの経路計画



障害物（机，テーブル）に沿ってロボットを動かす
ロボットの中心が描く軌跡

移動ロボットの経路計画



移動ロボットの中心が存在しうる領域

→ 移動ロボットを点とみなして経路を計画する

配位空間 (configuration space, C-space)

まとめ

乱数

一様乱数 正規乱数
モンテカルロ法

移動ロボットの経路計画

PRM (Probabilistic Roadmaps)
配位空間 (configuration space, C-space)

レポート (MATLAB Grader)

MATLAB grader 「確率的手法」
締切：11月27日（月曜）午前1時