시뮬레이션 기초 및 실습

HW 1

제출일 2019년 4월 3일
과목명 시뮬레이션 기초 및 실습
담당교수 김지범 교수님
학과 산업경영공학과
학번 201401210
이름 강형원

1. 반지름이 1인 원 $x^2+y^2=1$ 위에 n개의 점이 있다고 하자. 이 n개의 점 사이의 거리는 모두 같다. 이 n개의 점을 이용하여 이 원에 내접하는 다각형을 그렸더니 n=6인 경우 아래 그림 (a)와 같은 다각형이 나왔다. n개의 균등하게 떨어진 점을 이용하면 이 다각형의 넓이는, $A_n = \frac{n}{2} sin(\frac{2\pi}{n})$이다. 이 n개의 균등하게 떨어진 점에서 접선을 그어서 이 원에 외접하는 다각형을 그리면 원에 외접하는 다각형을 만들 수 있다. n=6인 경우 아래 그림 (b)와 같은 다각형이 나왔다. n개의 균등하게 떨어진 점을 이용하면 이 다각형의 넓이는, $B_n = n*\tan(\frac{\pi}{n})$이다. 원의 넓이는 π이므로 아래 그림을 보면 $A_n < \pi < B_n$이 성립함을 알 수 있다. n값을 점점 크게 하면서 $\rho_n = \frac{A_n + B_n}{2}$, 값으로 π 값을 근사화 하고자 한다. 절대 오차는 $|\rho_n - \pi|$라고 정의할 수 있다.

1) 사용자로부터 원하는 절대 오차 임계값 (tol)을 입력 받아서 $|\rho_n - \pi|$<tol를 만족시킬 때 까지 n값을 1씩 증가시키면서 실험을 수행하고자 한다. tol 값이 0.001일 때 $|\rho_n - \pi|$<tol를 만족시키는 가장 작은 정수 n값이 얼마가 나오는지 출력해 보고 이때의 절대 오차 값도 출력해보자. 'while' 반복문을 사용하고 반복문을 수행하기 전 최초의 n값은 3부터 시작하자.

- Algorithm

```
n = 3;  % 점의 수
an = n/2*sin(2*pi/n);   % 내접하는 다각형의 넓이
bn = n*tan(pi/n);   % 외접하는 다각형의 넓이
pn = (an+bn)/2; % pi값 근사화
tol = 0.001;    % 오차 임계값

while   ρ −π|<tol 만족시킬 때 까지
    n의 값을 1씩 증가시키면서   n, Bn을 계산하여 ρn의 값을 구한다.
end
```

- code

```
n = 3;  % 점의 수
an = n/2*sin(2*pi/n);   % 내접하는 다각형의 넓이
bn = n*tan(pi/n);   % 외접하는 다각형의 넓이
pn = (an+bn)/2; % pi값 근사화
tol = 0.001;    % 오차 임계값

while abs(pn-pi) >= tol % abs(pn-pi)<tol 될 때 까지
    n = n+1;
    an = n/2*sin(2*pi/n);
    bn = n*tan(pi/n);
    pn = (an+bn)/2;
end
```

- 결과

```
n의 값:
    72

절대 오차 값:
   9.9534e-04
```

2) $|A_{n+1}-A_n|<$tol 또는 $|B_{n+1}-B_n|<$tol를 만족시킬 때 까지 n값을 1씩 증가시키면서 실험을 수행하고자한다. tol값이 0.001일 때 $|A_{n+1}-A_n|<$tol 또는 $|B_{n+1}-B_n|<$tol를 만족시키는 가장 작은 정수n값이 얼마가 나오는지 출력해 보고 이때의 절대 오차 값도 출력해보자. 'while' 반복문을 사용하고 반복문을 수행하기 전 최초의 n값은 3부터 시작하자.

- Algorithm

```
n = 3;  % 점의 수
an = n/2*sin(2*pi/n);   % 내접하는 다각형의 넓이
bn = n*tan(pi/n);   % 외접하는 다각형의 넓이
pn = (an+bn)/2; % pi값 근사화
tol = 0.001;    % 오차 임계값

while      +1 − An|<tol 또는 |Bn+1 − Bn|<tol만족시킬 때 까지
    n의 값을 1씩 증가시키면서 An, Bn을 계산한다.
end
```

- code

```
n = 3;  % 점의 수
an = n/2*sin(2*pi/n);   % 내접하는 다각형의 넓이
bn = n*tan(pi/n);   % 외접하는 다각형의 넓이
pn = (an+bn)/2; % pi값 근사화
tol = 0.001;    % 오차 임계값

while abs(an-pi) >= tol && abs(bn-pi) >= tol % abs(an-pi) < tol 또는
abs(bn-pi) < tol이 될 때까지
    n = n+1;
    an = n/2*sin(2*pi/n);
    bn = n*tan(pi/n);
end
```

- 결과

```
n의 값:
   102

절대 오차 값:
    0.1060
```

2. 수업시간에 배운 random walk 시뮬레이션처럼 행수와 열수가 모두 (2n+1)인 정사각형 모양의 타일을 사용하고 로봇이 타일 끝에 도달하면 실험은 종료된다. 로봇의 최초 위치도 이 정사각형 모양의 타일의 한 가운데서 시작한다. 아래 그림 (c)는 n=5인 경우이다. 이 실험에서도 n=5로 정하자. 수업시간에 배운 random walk 시뮬레이션과의 차이는 로봇이 움직일 때 동서남북만이 아닌 동서남북에다가 북동, 북서, 남동, 남서 방향이 추가된 총 8가지 방향으로 이동가능하다. 단, 8가지 방향으로 이동할 확률은 모두 같다고 가정하자. 이 실험 (trial)을 총 1000번 연속으로 반복 실험했다고 가정하자.

1) Lecture 4의 Page 23과 같이 n=5, 10, 20, 40까지 증가시키면서 로봇이 타일 끝에 도달할 때까지의 평균 이동 횟수 (1000번 반복 실험의 평균)을 출력해보고 이를 따로 정리해서 표 형태로 보여주자. 평균적으로 8방향으로 움직이는 것이 4방향으로 이동하는 것보다 평균 이동 횟수가 짧은 것으로 보이는가? n값이 2배 커질 때 마다 평균 이동 횟수가 대략 얼마 정도 증가하는가? 토의해보자.

- Algorithm

```
trial = 1000; % trial 횟수

% 로봇이 타일 끝에 다다를 때까지 움직이는 횟수를 구하는 실험을 1000번 반복
for n=[5 10 20 40]  타일 크기 n=5, 10, 20, 40
    count_wh = 0; 1000번 반복 실험하면서 전체 이동 횟수
    for 1000번 반복
        xc=0;yc=0;
        count = 0; 로봇이 타일 끝에 다다르기 까지 이동 횟수
        while 로봇이 타일 끝에 다다를 때 까지
                로봇 방향 정하고 방향에 따라서 xc와 yc 업데이트
                count(이동 횟수) 증가
        end
        count_wh(1000번 반복 실험하면서 전체 이동 횟수) 업데이트
    end
    끝에 다다를 때까지 평균 이동횟수 구하기
end
```

로봇이 움직이고 횟수를 구하는 큰 알고리즘은 4방향과 8방향 모두 동일
0에서 1사이 난수를 뽑아 범위를 4로 나누고 동, 서, 남, 북으로 지정해주면 4방향으로 표현 할 수 있으며, 범위를 8로 나누고 동, 서, 남, 북, 북동, 북서, 남동, 남서로 지정해주면 8방향을 표현 할 수 있다.

- code

```
trial = 1000; % trial 횟수

% 4방향으로 이동 할 때
for n=[5 10 20 40]  % 타일 크기 [5 10 20 40]
    count_wh = 0;
    for k=1:trial
```

```
        xc=0;yc=0;
        count = 0;
        while abs(xc) < n && abs(yc) < n % 타일 끝에 다다를 때 까지
            r = rand;
            if r<=0.25
                yc=yc+1;    % 북쪽으로 한 칸 이동
            elseif 0.25<r && r<=0.5
                xc=xc+1;    % 동쪽으로 한 칸 이동
            elseif 0.5<r && r<=0.75
                yc=yc-1;    % 남쪽으로 한 칸 이동
            else
                xc=xc-1;    % 서쪽으로 한 칸 이동
            end
            count = count + 1;  % 이동 횟수
        end
        count_wh = count_wh + count;     % 1000번 시도 했을 경우 총 이동 횟수
    end
    mean = count_wh/1000;   % 끝에 다다를 때 까지 평균 이동 횟수
end

% 8방향으로 이동 할 때
for n=[5 10 20 40]  % 타일 크기 [5 10 20 40]
    count_wh = 0;
    for k=1:trial
        xc=0;yc=0;
        count = 0;
        while abs(xc) < n && abs(yc) < n % 타일 끝에 다다를 때 까지
            r = rand;
            if r<=0.125
                yc=yc+1;    % 북쪽으로 한 칸 이동
            elseif 0.125<r && r<=0.25
                xc=xc+1;    % 동쪽으로 한 칸 이동
            elseif 0.25<r && r<=0.375
                yc=yc-1;    % 남쪽으로 한 칸 이동
            elseif 0.375<r && r<=0.5
                xc=xc-1;    % 서쪽으로 한 칸 이동
            elseif 0.5<r && r<=0.625
                xc=xc+1;    % 북동쪽으로 대각선 이동
                yc=yc+1;
            elseif 0.625<r && r<=0.75
                xc=xc+1;    % 남동쪽으로 대각선 이동
                yc=yc-1;
            elseif 0.75<r && r<=0.875
```

```
            xc=xc-1;    % 북서쪽으로 대각선 이동
            yc=yc+1;
        else
            xc=xc-1;    % 남서쪽으로 대각선 이동
            yc=yc-1;
        end
        count = count + 1;  % 이동 횟수
    end
    count_wh = count_wh + count;    % 1000번 시도 했을 경우 총 이동 횟수
end
mean = count_wh/1000;   % 끝에 다다를 때 까지 평균 이동 횟수
end
```

- 결과

	n=5	n=10	n=20	n=40
4방향	29.2070	119.8930	479.0790	1823
8방향	19.3400	78.1060	323.8960	1230.2

결과를 비교해보았을 때 8방향으로 움직이는 것이 4방향으로 움직이는 것보다 타일 끝에 더 적은 이동 횟수로 도달하는 것을 확인 할 수 있으며, n값이 커짐에 따라 도달하는 데 걸리는 이동 횟수가 커지는 것을 확인 할 수 있은데 n의 값이 2배 커질 때 마다 이동 횟수는 4배 정도 커지는 것을 볼 수 있다.

2) 로봇이 타일에 끝에 도달하는 경우는 2가지가 있다. 완전 코너 부분, 혹은 타일의 끝이지만 완전 코너 부분이 아닌 경우. 완전 코너 부분에 돌아갈 횟수와 타일의 끝이지만 완전 코너 부분이 아닌 부분에 들어가는 평균 횟수를 비교하고자 한다. 각 부분에 들어간 평균 횟수(1000번 반복 실험의 평균)를 비교하여 출력해보자. 둘 중 어느 부분에 도달할 횟수가 더 많다고 예상하는가? 실험으로 보여주고 토의해 보자. n값이 커짐에 따라 결과는 어떻게 달라지는가? 토의해보자.

- **Algorithm**

```
trial = 1000; % trial 횟수

for n=[5 10 20 40]  % 타일 크기 [5 10 20 40]
    count_wh = 0;   % 전체 이동 횟수
    count1 = 0; % 완전 코너가 아닌 부분에 들어가는 경우 이동 수
    count2 = 0; % 완전 코너에 들어가는 경우 이동 수
    n1 = 0; % 완전 코너가 아닌 부분에 들어가는 횟수
    n2 = 0; % 완전 코너에 들어가는 횟수
    for 1000번 반복
        xc=0;yc=0; % xc, yc 초기화
        count = 0; % 이동횟수 초기화
        while 로봇이 타일 끝에 다다를 때 까지
                로봇 방향 정하고 방향에 따라서 xc와 yc 업데이트
                count(이동 횟수) 증가
        end
```

```
        count_wh(1000번 반복 실험하면서 전체 이동 횟수) 업데이트
        if 타일 끝에 도달하는 경우 중
            if 완전 코너에 도달하는 경우
                완전 코너에 도달하는 경우 도달 횟수와 이동 횟수 구하기
            else    완전 코너 부분이 아닌 부분에 들어가는 경우
                완전 코너에 부분이 아닌 부분에 도달하는 경우 도달 횟수와 이동 횟수 구하기
            end
        end
    end
   전체, 완전 코너 부분이 아닌 부분에 들어가는 경우, 완전 코너 부분에 들어가는 경우 평균
   이동 횟수 구하기
end
```

- code

```
trial = 1000; % trial 횟수

for n=[5 10 20 40]  % 타일 크기 [5 10 20 40]
    count_wh = 0;   % 전체 이동 횟수
    count1 = 0; % 완전 코너가 아닌 부분에 들어가는 경우 이동 수
    count2 = 0; % 완전 코너에 들어가는 경우 이동 수
    n1 = 0; % 완전 코너가 아닌 부분에 들어가는 횟수
    n2 = 0; % 완전 코너에 들어가는 횟수
    for k=1:trial
        xc=0;yc=0;
        count = 0;
        while abs(xc) < n && abs(yc) < n % 타일 끝에 다다를 때 까지
            r = rand;
            if r<=0.125
                yc=yc+1;    % 북쪽으로 한 칸 이동
            elseif 0.125<r && r<=0.25
                xc=xc+1;    % 동쪽으로 한 칸 이동
            elseif 0.25<r && r<=0.375
                yc=yc-1;    % 남쪽으로 한 칸 이동
            elseif 0.375<r && r<=0.5
                xc=xc-1;    % 서쪽으로 한 칸 이동
            elseif 0.5<r && r<=0.625
                xc=xc+1;    % 북동쪽으로 대각선 이동
                yc=yc+1;
            elseif 0.625<r && r<=0.75
                xc=xc+1;    % 남동쪽으로 대각선 이동
                yc=yc-1;
            elseif 0.75<r && r<=0.875
                xc=xc-1;    % 북서쪽으로 대각선 이동
```

```
            yc=yc+1;
        else
            xc=xc-1;    % 남서쪽으로 대각선 이동
            yc=yc-1;
        end
        count = count + 1;  % 이동 횟수
    end
    count_wh = count_wh + count;     % 1000번 시도 했을 경우 총 이동 횟수

    if abs(xc) >= n || abs(yc) >= n     % 타일 끝에 도달하는 경우 중
        if abs(xc) >= n && abs(yc) >= n     % 완전 코너에 도달 하는 경우
            count2 = count2 + count;
            n2 = n2 + 1;
        else    % 완전 코너 부분이 아닌 부분에 들어가는 경우
            count1 = count1 + count;
            n1 = n1 + 1;
        end
    end
end
mean = count_wh/1000;   % 전체 평균 이동 횟수
mean1 = count1/n1;  % 완전 코너 부분이 아닌 부분에 들어가는 평균 이동 횟수
mean2 = count2/n2;  % 완전 코너 부분에 들어가는 평균 이동 횟수
end
```

- **결과**

	n=5	n=10	n=20	n=40
전체 평균 이동 횟수	20.1830	78.6380	307.1370	1246.7
완전 코너 부분이 아닌 부분에 들어가는 횟수	986	996	1000	1000
완전 코너 부분이 아닌 부분에 들어가는 평균 이동 횟수	20.1156	78.6396	307.1370	1246.7
완전 코너 부분에 들어가는 횟수	14	4	0	0
완전 코너 부분에 들어가는 평균 이동 횟수	24.9286	78.2500	NaN	Nan

완전 코너에 도달하는 경우보다 완전 코너가 아닌 부분에 도달하는 경우가 더 많았으며, n값이 커짐에

따라 완전 코너에 도달하는 경우가 점점 줄었으며, n의 값이 20, 40인 경우 완전 코너에 들어가는 횟수가 없어서 완전 코너 부분에 들어갈 때 평균 이동 횟수를 구할 수 없었다.

3. 다음 수식은 n값이 크면 Euler 상수로 수렴하는 것으로 알려져 있다. n값은 양의 정수이다.

$$E_n = \left(\sum_{i=1}^{n} \frac{1}{i}\right) - \log(n)$$

while 반복문을 이용하여 $|E_{n+1} - E_n| < 0.001$이 되는 n값을 찾아보자. 이 수식은 n값이 커짐에 따라서 어떤 값으로 수렴하는가?

- Algorithm

```
n = 0;

while    +1 − En| < 0.001이 될 때까지 반복
    n을 1씩 증가하면서 En+1과 En을 계산하고
    |En+1 − En|을 계산하여 업데이트
end
```

- code

```
n = 0;

while 1
    n = n+1;
    en1 = 0;
    en = 0;
    for i=1:n+1
        en = en1;
        en1 = en1 + 1/i;
    end
    en1 = en1 - log(n+1); % Euler 상수 (E(n+1))
    en = en - log(n);   % Euler 상수 (En)
    if abs(en1-en) < 0.001
        break   % abs(en1-en) < 0.001인 경우 반복문 중단
    end
end
```

- 결과

```
n:
    22

en:
    0.5998
```

$|E_{n+1} - E_n| < 0.001$**이 되는 n의 값은 22이고 n의 값이 커짐에 따라 0.59에 수렴한다.**