Łukasz Plust 186437

Laboratorium Nr.2 Metody Numeryczne

Zadanie 1

I.

Zdefiniowanie zmiennych potrzebnych w kodzie.

```
%1
a = 1;%bok kwadratu
r_max = a/2;%promien
n_max = 200;%ilosc pecherzykow

%2
X = 0;%wspolrzedna X srodka kola
Y = 0;%wspolrzedna Y srodka kola
R = 0;%promien

n = 0;%2d chyba -> pocztkowa wartosc narysowanych okregow(ilosc)

x = [];%zapamietana wartosc X okregu
y = [];%zapamietana wartosc Y okregu
r = [];%zapamietana wartosc R okregu

count = 0;%liczba prob losowania kolka
all_area = [];%powierzchnia calkowita
counter = [];%zapamietana liczba losowan
```

II.

```
while (n <= n_max)% wartosc jest mniejsza od maksymalnej
    fit = false;%ustawiam flage na false->zakladam,ze nie pasuje
    while(fit == false)
        X = rand(1) * a;% rand(1)->wybiera liczbe miedzy 0 a 1
        Y = rand(1) * a;
        R = rand(1) * r_max; % rand(5) -> wyrzuci macierz 5x5 z warosciami od 0 do 1?
        X_LEFT = X - R;
        X_RIGHT = X + R;
        Y_UP = Y + R;
        Y_DOWN = Y - R;
        if (X_RIGHT <= a && X_LEFT > 0 && Y_UP <= a && Y_DOWN >0)
            fit = true;%warunek mieszczenia sie w kwadracie
        end
        count = count +1;%liczba prob losowania kolka
    end
```

Pierwsza część kodu sprawdza warunek gdy ilość okręgów jest mniejsza, bądź równa wartości maksymalnej okręgów.

W dalszej części programu losuję wartości parametrów okręgu z zakresu <0,1> i skaluję do wartości maksymalnej parametru a oraz r_max.

Następnie deklaruje wartości, które pozwolą mi sprawdzić czy wylosowane wartości spełniają warunek mieszczenia się okręgu w kwadracie. Jeśli warunek się spełni flaga fit zostanie ustawiona na true. Po każdym przejściu pętli zwiększany jest count - licznik prób losowania okręgu.

III.

```
    intersects = false;
    for i = 1:n
        DistanceBetweenCenters = sqrt((x(i)-X)^2 + (y(i) - Y)^2);
        SumOfRadius = R + r(i);
        if(SumOfRadius >= DistanceBetweenCenters)
        %jesli odleglosc wieksza niz promien to nie przecinaja sie
            intersects = true;
        break;
        end
    end
```

Druga część kodu sprawdza czy okręgi się przecinają. W celu weryfikacji czy okręgi się przycinają zastosowałem wzór na odległość między punktami i przypisałam do zmiennej DistanceBetweenCenters. Zsumowałem też wartości promieni i przypisałem do zmiennej SumOfRadius. Zamysł jest taki, że gdy odległość miedzy środkami jest większa od sumy promieni to okręgi się nie przecinają. Natomiast dla sumy promieni większej niż odległości miedzy środkami okręgi będą się przecinały.

IV.

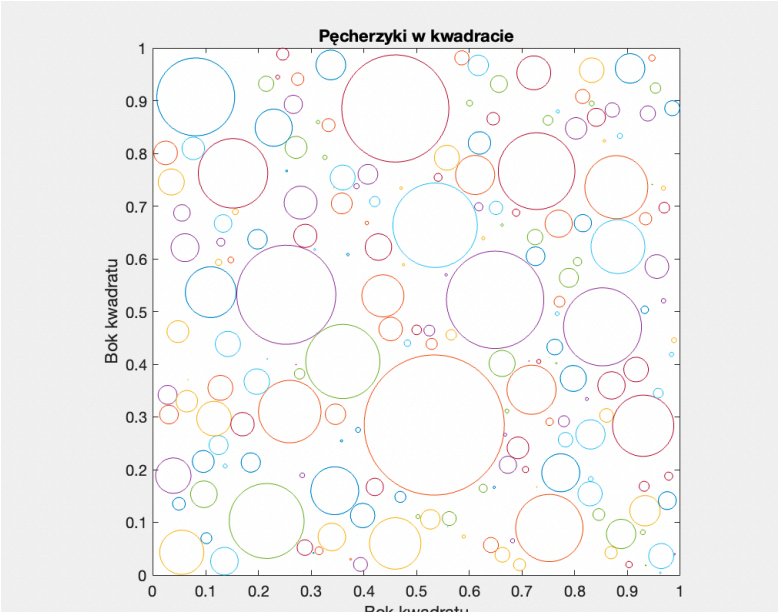
```
    if(intersects==false)
        n = n + 1;%zliczanie liczby narysowanych okregow
        x(n) = X;
        y(n) = Y;
        r(n) = R;
        area = pi*R*R;%potrzebne do zapamietania powierzchni
        if(n==1)
            all_area(n) = area;
        else
            all_area(n) = all_area(n-1) + area;
        end
        counter(n) = count;%zapamietywanie liczby losowan
        count = 0;

        fprintf(1, ' %s%5d%s%.3g\r ', 'n =',  n, ' S = ', area);
        pause(0.01);
        axis equal
        axis([0 a 0 a]); % x nalezy od 0 do a, y nalezy od 0 do a
        plot_circ(X,Y,R);
        hold on%elementy dodane dotychczas zostana zachowane po kolejnym wywolaniu plot
    end
```

Trzecia część kodu dla przypadku, gdy okręgi się nie przecinają zwiększam ilość narysowanych okręgów. Następnie zapisuje w tablicach wartości x, y oraz r. W celu zapamiętania powierzchni korzystam ze wzoru na pole koła i przypisuję do zmiennej area. W przypadku, gdy nie ma żadnego koła, suma pól jest równa polu pierwszego koła, dla pozostałych dodaje do poprzedniej wartości all_area nową wartość koła(area). Zapamiętuję liczbę losowań okręgu do counter. Kolejne komendy to podpowiedzi z instrukcji.

V.

Narysowane pęcherzyki w kwadracie.



Zadanie 2.

I.

Deklaracja zmiennych potrzebnych w zadaniu.

```
%ZAD A
%wygenerowanie tablicy edges
Edges = sparse([1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 5, 5, 6, 6, 7;
                4, 6, 4, 5, 3, 6, 5, 7, 6, 5, 6, 4, 4, 7, 6]);

%ZAD B
N = 7; %liczba stron w sieci
d = 0.85;%parametr z polecenia
```

II.

```
%macierz sąsiedztwa->przedstawia za pomoca macierzy sasiedztwa polaczenie sieci
B = sparse(Edges(2,:), Edges(1,:),1,N,N); %1->wartosc elementow niezerowych
%macierz jednostkowa
I = speye(N);
%L(i) wyznaczam dzięki wczesniej obliczonej macierzy B
%L(i) -> liczba linków wychodzacych ze strony i-tej
L = zeros(1,N);

for i = 1:N
    L(i) = sum(B(:,i));
end
L = sparse(L)';%tu trzeba zrobic transpozycje
%dzieki sparse podawane sa tylko dane odnosnie wartosci, pomija sie 0

%wektor b
b = zeros(N,1);
b(:,1) = (1-d)/N;%wypelniam cala kolumne(dlatego biore wszystkie wiersze czyli -> :,

%macierz diagonalna->wartosci tylko na przekatnej,reszta rowna 0
A=spdiags(1./L,0,N,N);

M = sparse(I - d*B*A); %macierz M
```

Macierz sąsiedztwa B tworzę używając Edges(2,:) -> oznacza, że biorę 2 wiersz i wszystkie kolumny, Edges(1,:) -> oznacza, że biorę 1 wiersz i wszystkie kolumny

Macierz jednostkową tworzę dzięki funkcji speye.

Liczbę linków wychodzących ze strony i-tej obliczam sumując L(j) = B(i,j) od i=1 do N

Następnie na zmiennej L wykonuję transpozycję potrzebną do obliczenia A przy spdiags.

Zmienną b (wszystkie wiersze pierwszej kolumny) wypełniam obliczoną wartością.

M obliczam, według wzoru mając wszystkie zmienne.

III.

W dalszej części programu wykonuję polecenie z C i D oraz obliczam r przez M\b.
Następnie generuję wykres poleceniem bar(r) i zapisuję plik do bar.png

```
%ZAD C
whos A B I M b
%ZAD D
spy(B);
print -dpng spy_b

%ZAD E
%r->wartosci PR wszystkich N stron w sieci
r = M\b;

bar(r);
saveas(gcf, "bar.png");
```

Zadanie C

sparse_test

Name	Size	Bytes	Class	Attributes
A	7x7	176	double	sparse
B	7x7	304	double	sparse
I	7x7	176	double	sparse
M	7x7	416	double	sparse
b	7x1	56	double	

Zadanie D