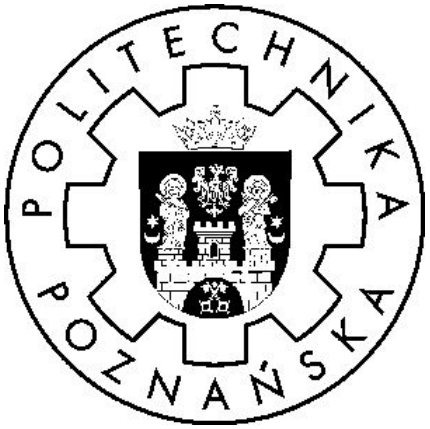


Wyliczanie π metodą Monte Carlo

Przetwarzanie Rozproszone, lab. nr 4

Tekst dokumentu został oparty o dokument autorstwa zespołu Politechniki Poznańskiej (A. D. Danilecki, M. Szychowiak i M. Sajkowski) przygotowującego materiały dydaktyczne na potrzeby projektu *elearningowego*. Oryginał dostępny pod adresem http://wazniak.mimuw.edu.pl/index.php?title=Pr_m06_lab

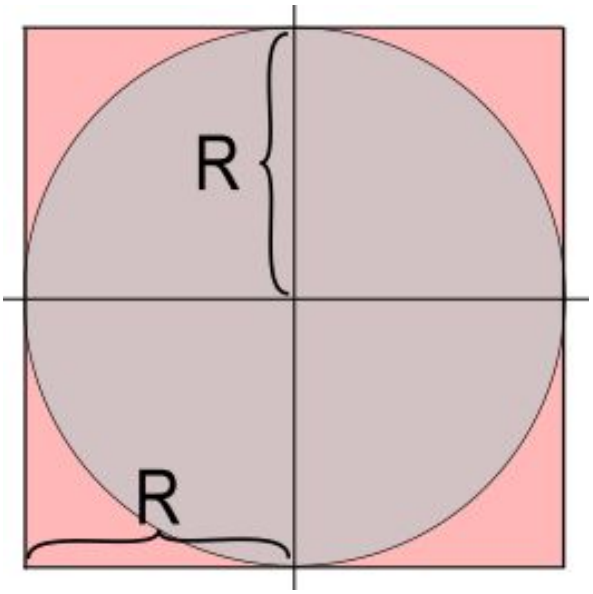
Zakres ćwiczenia

Głównym celem ćwiczenia jest podniesienie umiejętności w zakresie stosowania dotąd nabytej wiedzy. Tak więc nie zostaną wprowadzone żadne nowe funkcje biblioteki MPI. Zostanie za to przedstawiona metoda wyliczania liczby π, która stosunkowo dobrze nadaje się do zrównoleglenia.

Opis metody Monte-Carlo obliczania liczby π

Liczbę π można obliczać na wiele różnych sposobów. Sposób, który podamy obecnie, nie jest bynajmniej najlepszym z nich a głównym motywem wybrania właśnie jego jest tzw. cel dydaktyczny. Metoda Monte-Carlo, z którą zaraz się zapoznasz, jest dość łatwa do zrozumienia oraz dobrze daje się ją zrównoleglić, co czyni ją idealną do wszelkiego rodzaju kursów programowania w środowisku rozproszonym i tradycyjnie pojawia się w wielu ćwiczeniach uczących programowania w Ada95, PVM czy też MPI.

Obejrzyj teraz rysunek poniżej. Przedstawia on koło o promieniu R wpisane w kwadrat o boku 2R.



Gdybym zadał ci teraz pytanie, ile wynosi pole przedstawionego wyżej kwadratu, na pewno odpowiedziałbyś bez wahania $4r^2$. Pole koła zaś wynosi πr^2. Oznacza to, że stosunek pola koła do pola kwadratu wynosi:

$$\frac{P_{koła}}{P_{kwadratu}} = \frac{\pi r^2}{4r^2}$$

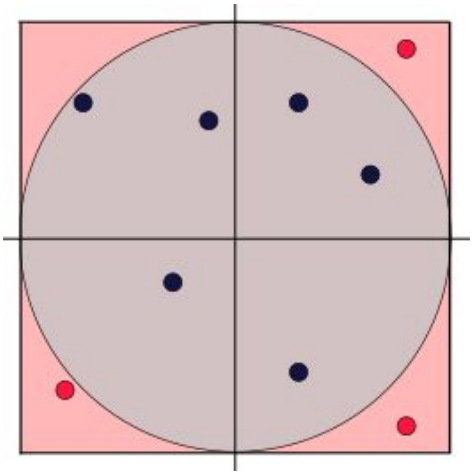
Z tego z kolei wynika, że mając obliczone wcześniej w jakiś sposób pole kwadratu i pole koła wpisanego w ten kwadrat, możemy łatwo obliczyć π:

$$4\frac{P_{koła}}{P_{kwadratu}} = \pi$$

Ta konstatacja być może wywołała na twojej twarzy uśmiech. W końcu, żeby obliczyć pole koła, musimy znać π, mógłbyś/mogłabyś powiedzieć. Owszem, to prawda. Ale istota metody Monte-Carlo polega na tym, że możemy zastosować powyższą równość bez obliczania pola koła.

Wyobraź sobie teraz, że rzucamy ziarenkami piasku w narysowany kwadrat z zakreślonym w środku kołem. Zgodzisz się chyba z stwierdzeniem, że jeśli będziemy rzucali dostatecznie długo, w końcu ziarnka piasku pokryją cały kwadrat, a stosunek liczby ziarenek piasku w środku narysowanego koła w stosunku do wszystkich ziarenek piasku w całym kwadracie będzie równy mniej więcej stosunkowi pola koła do pola kwadratu. Bardziej formalnie możemy ten wniosek ubrać w słowa w następujący sposób:

Jeżeli będziemy losować punkty o współrzędnych od -$2r$ do $2r$, to stosunek liczby punktów zawierających się w kole o środku w punkcie <0,0> i promieniu r do wszystkich



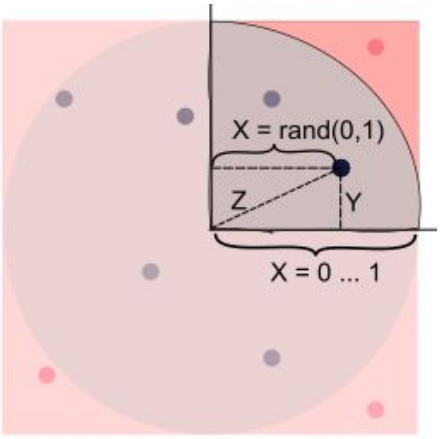
wylosowanych punktów, będzie dążył w nieskończoności (z pewnym prawdopodobieństwem) do stosunku tego pola koła do koła kwadratu o boku $2r$.

Ideę tę prezentuje rycina po lewej. Widzisz tam wylosowanych dziewięć punktów, z czego sześć znalazło się w środku koła. Podstawiając do naszego wzoru, 4*6/9=2.66666, co oczywiście jest bardzo słabym przybliżeniem liczby π.

Zauważ, że stosunek z naszego wzoru będzie identyczny również do ćwiartki koła. Jeżeli pole koła podzielimy na cztery i tak samo podzielimy pole kwadratu, to ich stosunek będzie wciąż taki sam. Oznacza to, że wystarczy, jeżeli będziemy losowali punkty o współrzędnych od 0 do r.

Cała metoda sprowadza się więc do tego, by losować punkty, sprawdzać, czy mieszczą się w kole, i następnie podstawiać liczby wylosowanych punktów do wzoru powyżej. Losując *odpowiednio dużo* punktów, powinniśmy otrzymać z pewnym prawdopodobieństwem rozsądne przybliżenie liczby π.

Dla ułatwienia, przyjmiemy *r*=1, a więc będziemy losować współrzędne X oraz Y jako liczby rzeczywiste z zakresu od 0 do 1 (patrz obrazek poniżej).



Oczywiście, im więcej punktów wylosujemy, tym przybliżenie π bliższe faktycznej wartości 3,14159265358979323846264338327 95.... zakładając rzecz jasna, że będziemy losować współrzędne z równym prawdopodobieństwem.

Zadanie do samodzielnego wykonania

Twoim zadaniem obecnie jest napisanie programu który będzie obliczał liczbę \pi wyżej opisaną metodą Monte-Carlo. Aby ułatwić zadanie, pod adresem http://www.cs.put.poznan.pl/adanilecki/pr/mpi/pi.tgz odnajdziesz szkielet programu. Sam natomiast powinieneś zdecydować w jaki sposób podzielić obliczenia i jak zaimplementować algorytm. Szkielet programu można także ściągnąć przy pomocy napisanego przez nas skryptu setup-mpi, tak, jak zostało to przedstawione na zajęciach.

Do sprawdzenia, czy wylosowany punkt leży w obrębie koła należy po prostu sprawdzać, czy odległość wylosowanego punktu od środka układ współrzędnych jest mniejsza do r, a odległość tę można wyliczyć korzystając z wzoru Pitagorasa. Oczywiście, nie musisz (a nawet nie powinieneś/nie powinnaś) wyliczać pierwiastka. Aby się dowiedzieć, czy punkt o współrzędnych X oraz Y znajduje się w środku koła, wystarczy skorzystać ze wzoru poniżej:

$$X^2 + Y^2 < 1$$

Dlaczego 1? Bo R = 1.

W praktyce, nie powinieneś/nie powinnaś mieć problemu z uzyskaniem dokładności do dwóch miejsc po przecinku. Większość grupy zwykle daje radę uzyskać przybliżenie z dokładnością do pięciu miejsc po przecinku. Jedynie najlepszym udaje się dojść do 3,14159265.

Pamiętaj o kilku wskazówkach:

1. Nie nazywaj programu wykonywalnego *pi* (w systemie istnieje już polecenie o takiej nazwie).
2. Pamiętaj, by inicjować zarodek liczb losowych.
3. Pamiętaj, że jeżeli dzielisz dwie liczby całkowite, w C wynik jest liczbą całkowitą (czyli w języku C, 1 podzielone przez 3 daje ZERO).

UWAGA! Zasadą, której należy się trzymać zawsze w pisaniu programów rozproszonych, jest minimalizacja komunikacji. Im mniej procesy się komunikują i im więcej wykonują obliczeń lokalnie, tym większe przyśpieszenie. W przypadku wielu programów pisanych na ćwiczeniach stosunek obliczeń lokalnych do komunikacji jest tak niski, że praktycznie opóźnienia komunikacyjne i tak niweczą wszelkie pozytywne skutki rozproszenia aplikacji – niemniej jednak należy o tej zasadzie pamiętać.

Należy tak skonstruować program, by minimalizować liczbę błędów numerycznych. Błędy numeryczne mogą być tutaj wprowadzane przy operacjach dzielenia i (ewentualnie) obliczania pierwiastków (podczas sprawdzania, czy punkt mieści się w kole). Należy więc wybrać takie rozwiązanie, w którym tych operacji będzie jak najmniej. W przypadku naszego

problemu, rozwiązanie dobre to takie, w którym nie będzie w ogóle operacji na pierwiastkach.