Sprawozdanie z aproksymacji metodą Monte Carlo

Program został napisany w języku java. Mamy możliwość wyboru 4 funkcji podanych w zadaniu i ich zakresów [a,b]. Przy czym, aby wpisać dokładnie liczbę π należy przesłać do programu Stringa „pi”.

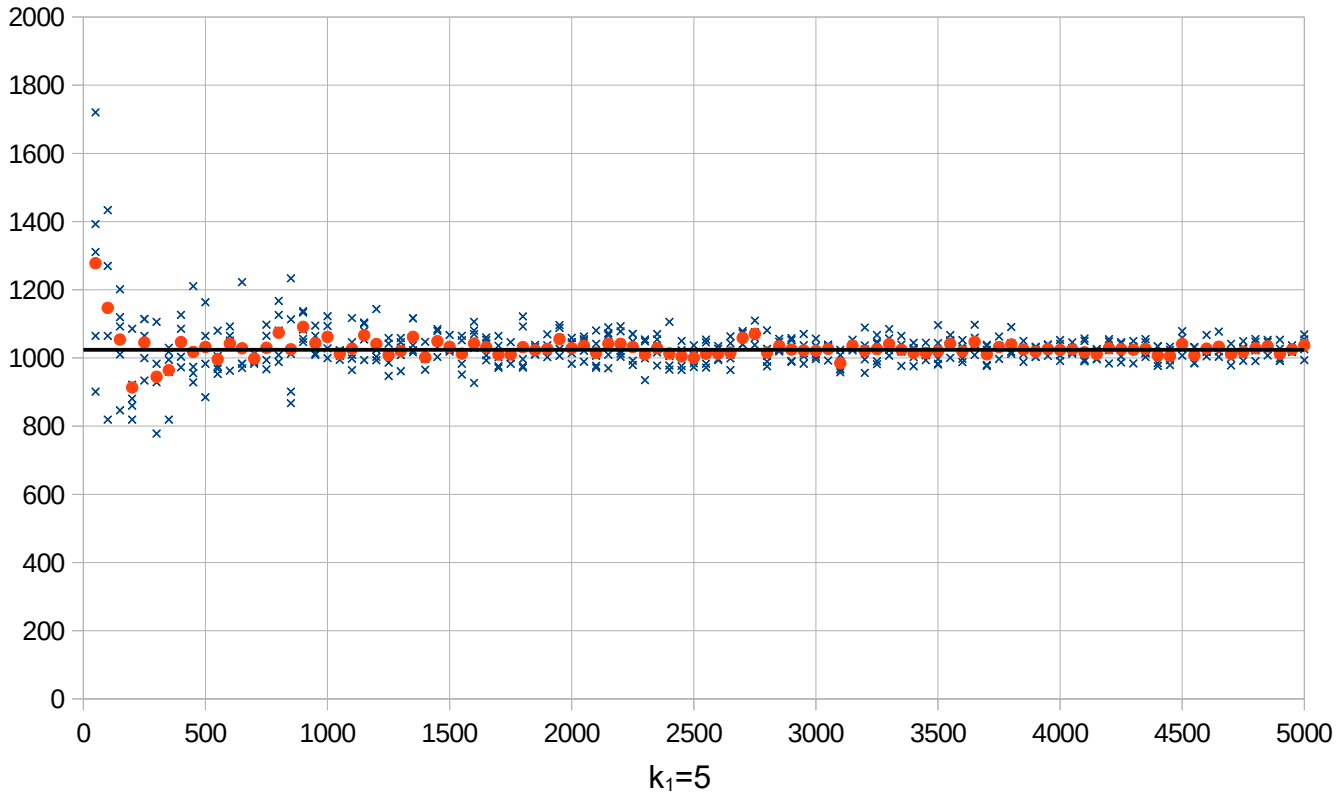
Przed wykonaniem zadania, zweryfikowałem używając dokumentacji, że funkcja nextDouble() klasy Random faktycznie posiada rozkład jednostajny (https://www.geneseo.edu/~baldwin/reference/random.html), dzięki czemu możemy przyjąć, że punkty mają losowe współrzędne.

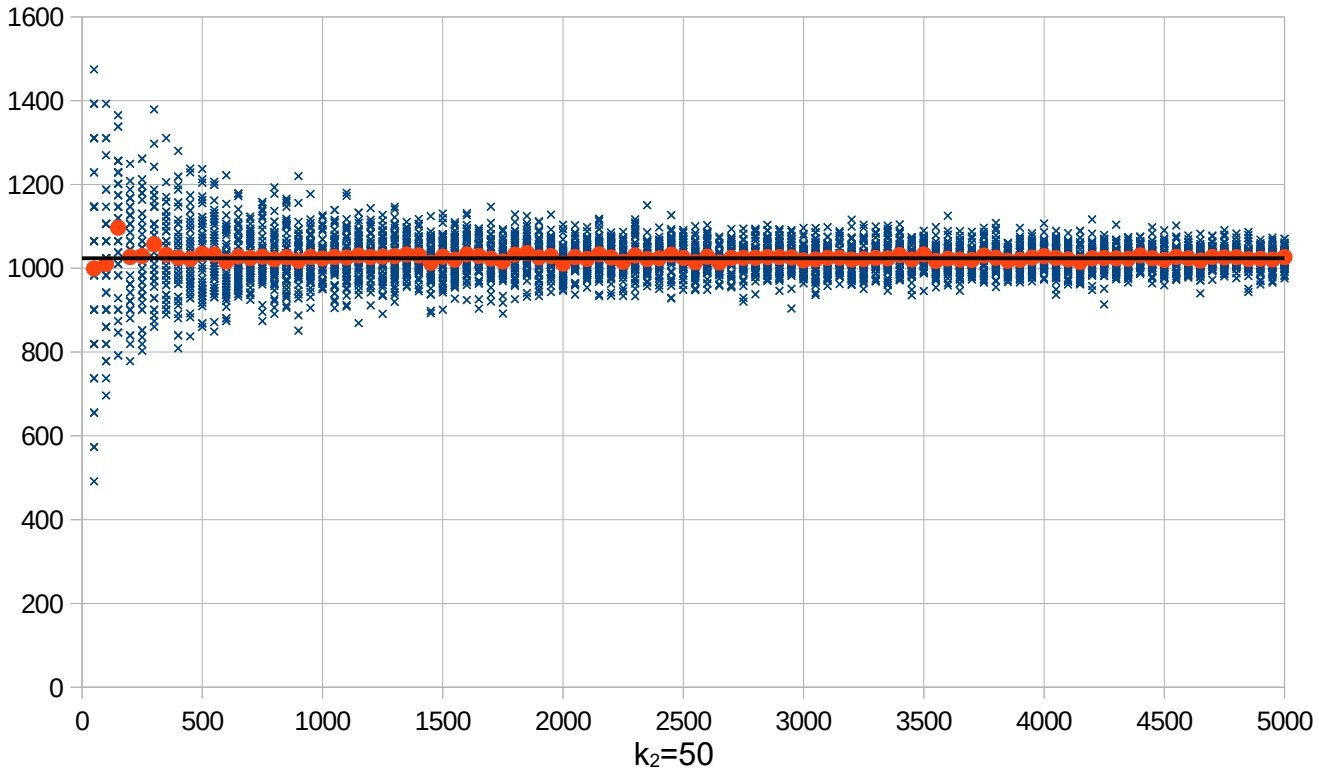
W celu stworzenia wykresów zdecydowałem się skorzystać z darmowego programu LibreOffice Calc – w tym celu zaaplikowałem w programie FileWriter’a, który odpowiada za tworzenie plików typu .csv (Comma seperated values).

Zadanie z podpunktu b – aproksymacja wartości liczby π – znajduję się pod funkcją nr4. Należy wtedy podać zakres [0,1] – w ten sposób obliczamy przybliżenie ¼ pola powierzchni koła o promieniu 1. Wynik następnie jest mnożony razy 4 dając nam pole całego koła – π (spodziewane).

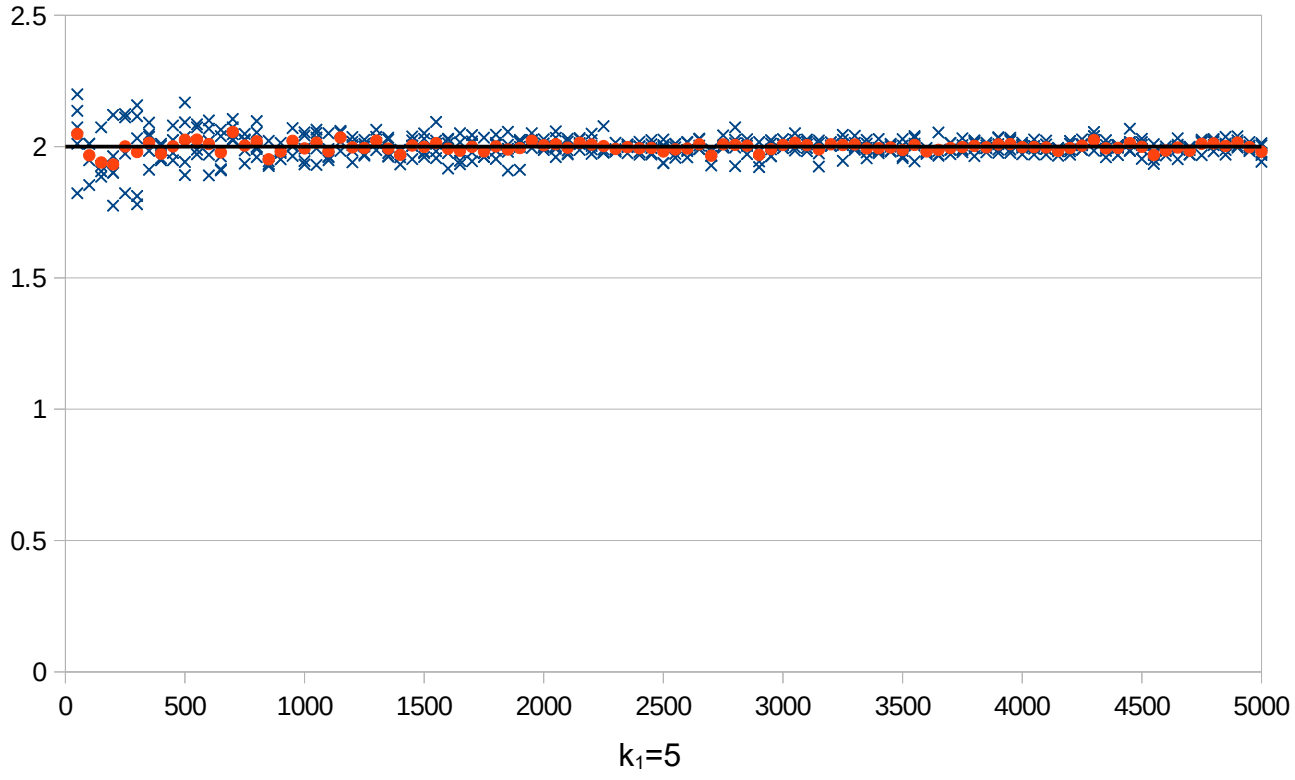
Poniżej przedstawiam wyniki moich eksperymentów:

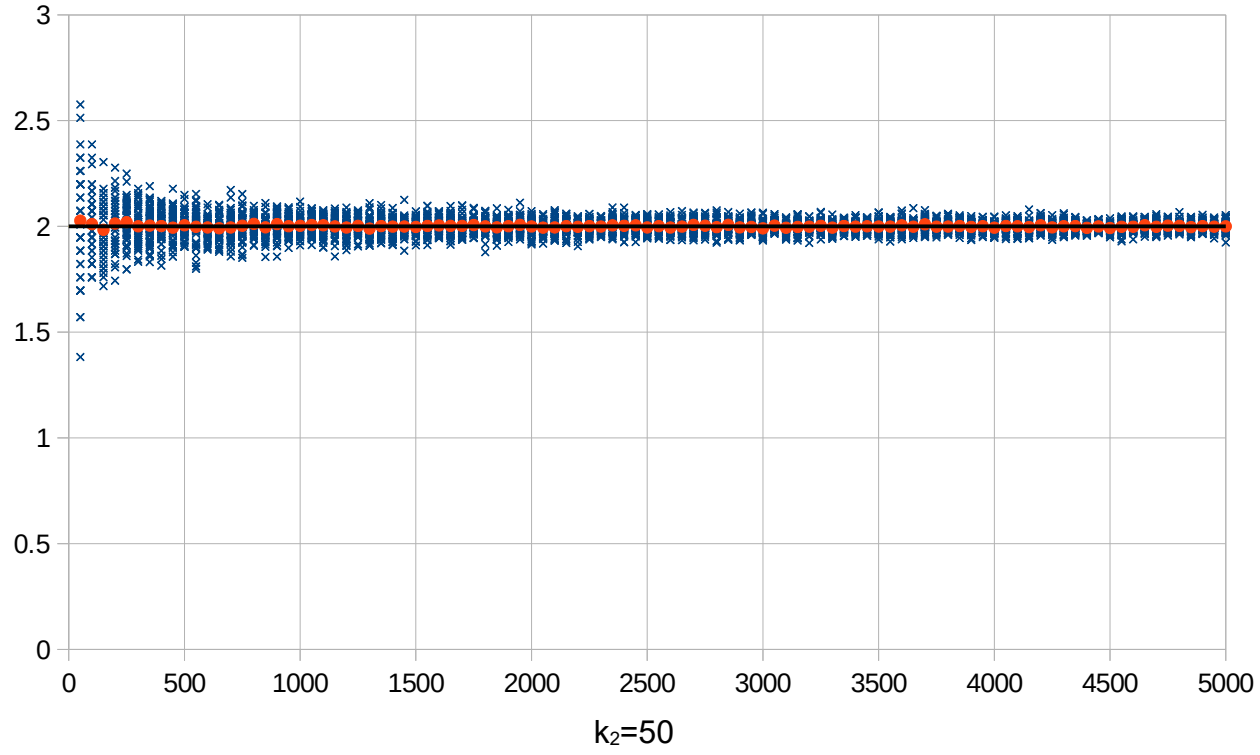
1) Funkcja 1 – x^3 dla [0,8]



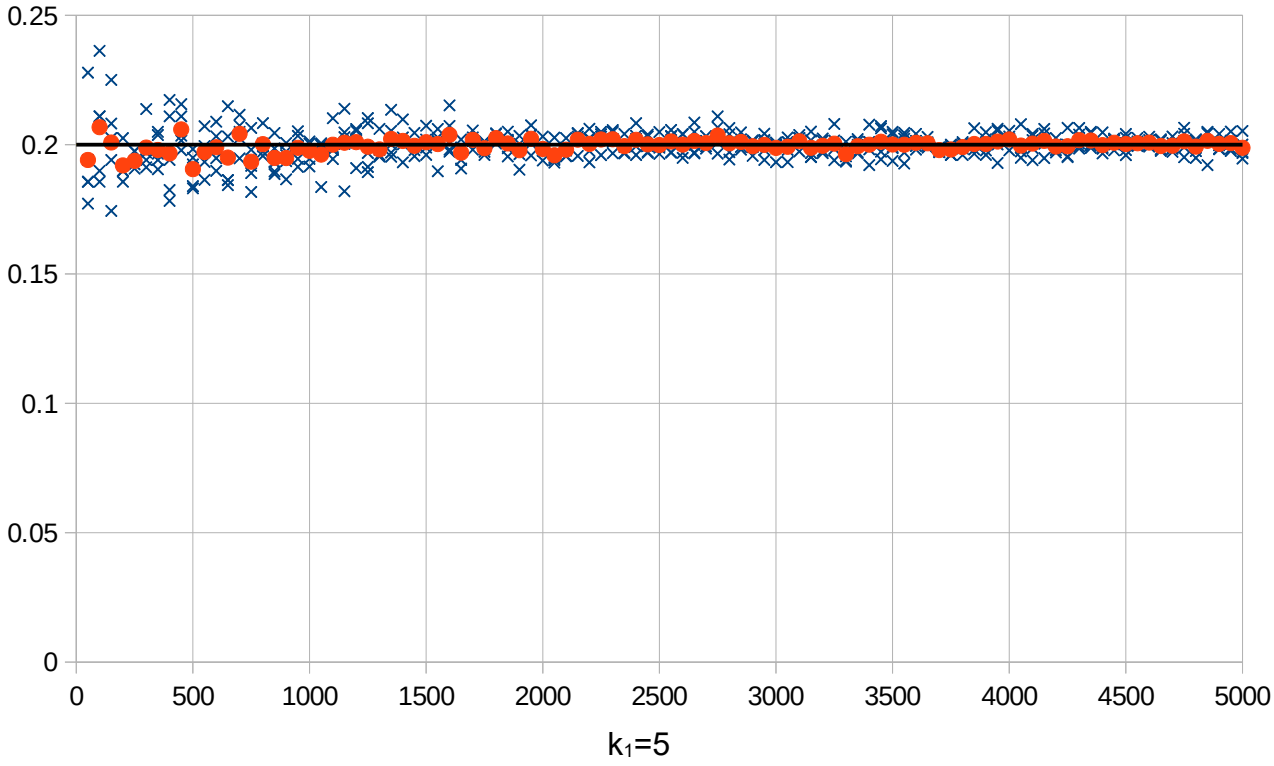


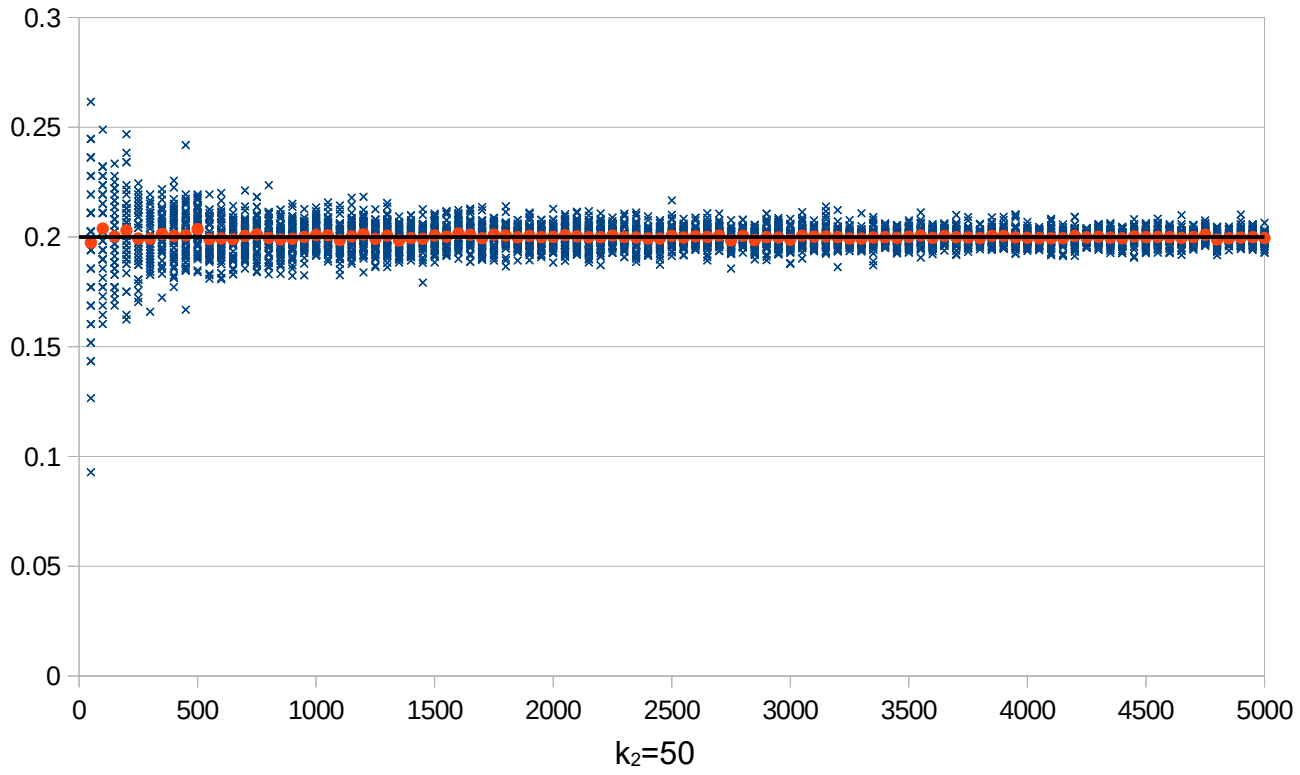
2) Funkcja 2 – sinx dla [0,π]



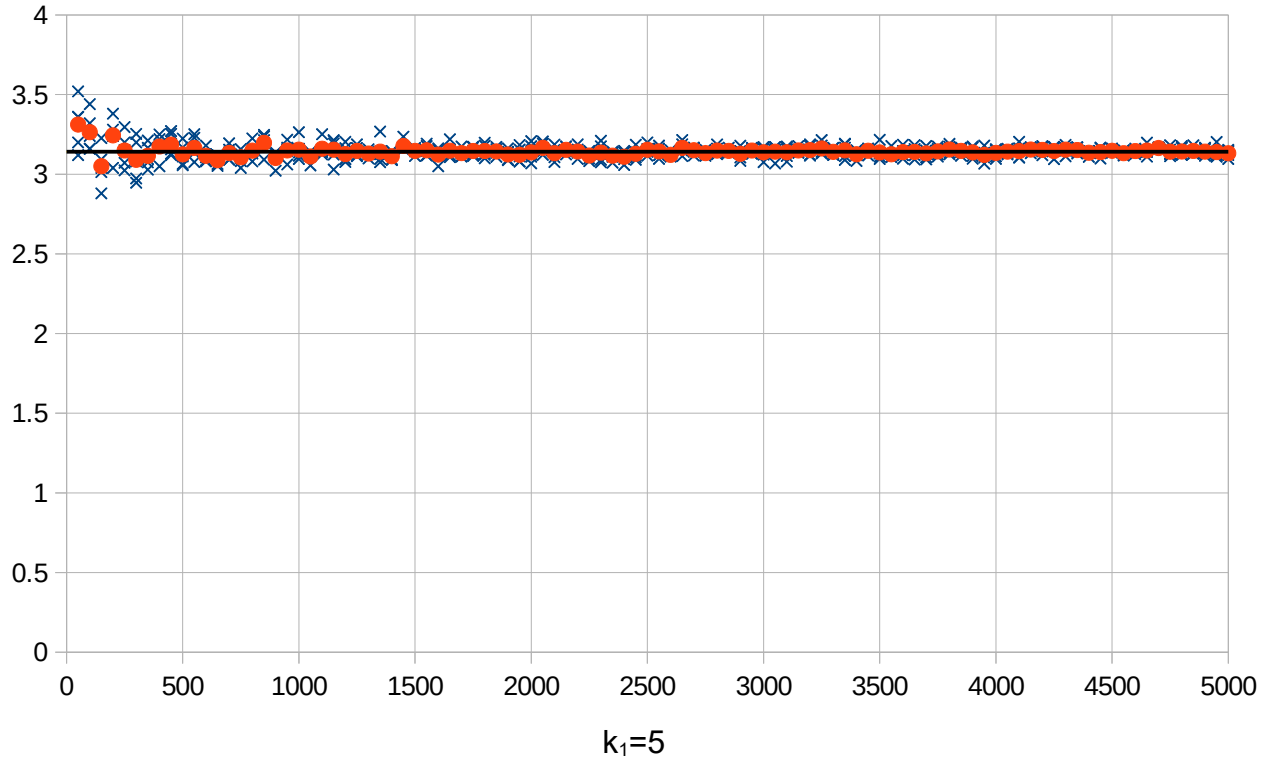


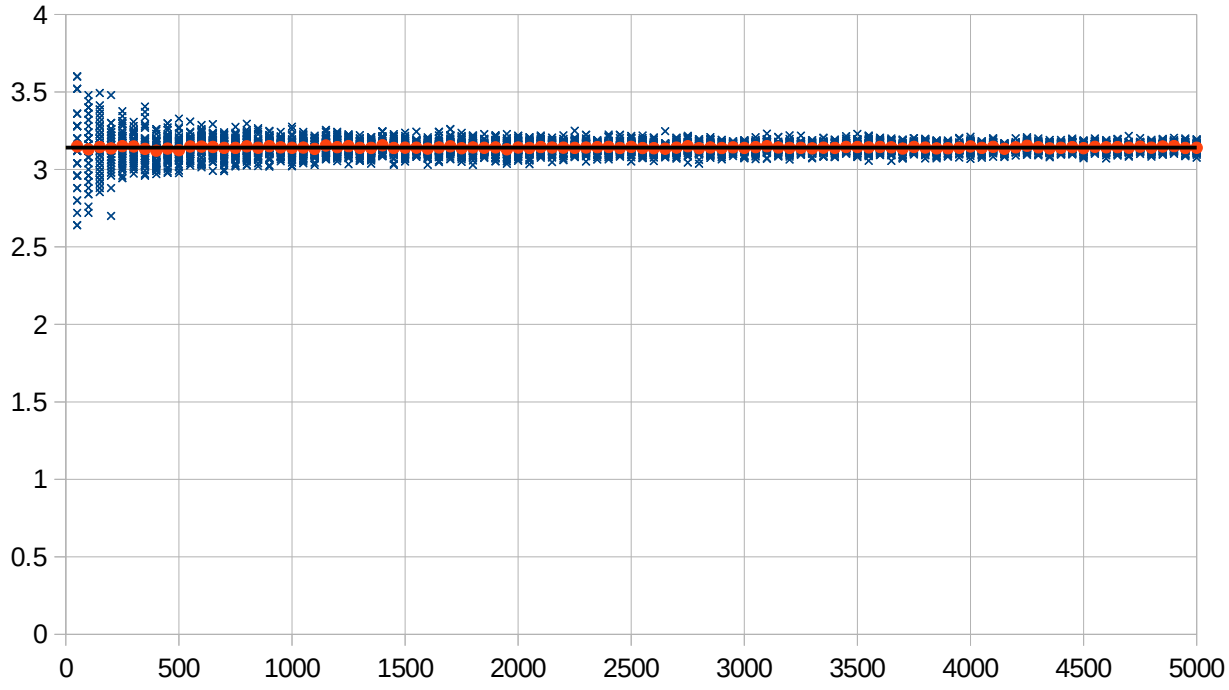
3) Funkcja $4x(1-x)^3$ dla [0,1]





4) Aproksymacja liczby π





k_2=50

Obserwacje

Rozdzielmy dwa przypadki – dla k = 5 i dla k = 50.

Dla pierwszego z nich widzimy, że poszczególne punkty dla n<=2000 (przykladowo wykres funkcji nr 3) znaczącą różnią się od oczekiwanej wartości – mają one zarówno niską precyzję jak i dokładność. W przypadku średnich wartość, również widoczna jest niska precyzja, ale sama dokładność jest już na znacznie wyższym poziomie – przeciwnie do niebieskich krzyżyków czerwone punkty zaczynają fluktuować wokół oczekiwanej wartości zdecydowanie szybciej (czyli w tym kontekście dla mniejszych n) i od umownego punktu n=2000 możemy mówić o zarówno wysokiej precyzji jak i dokładności eksperymentu (ten punkt krytyczny jest oczywiście zależny od eksperymentu – jak widać w przypadku nr 4 średnie wartości różnią się znacząco od faktycznej wartości liczby π tylko do n=1000).

Wnioskiem, który od razu się nasuwa jest fakt, iż dla tych 4 funkcji istnieje punkt n, od którego średnie wartości nie zyskują, ani na dokładności, ani na precyzji. Można więc rozważać owe n jako optymalną liczbę losowych punktów umieszczonych na płaszczyźnie dla k=5. Taka optymalizacja pozwala nam zaoszczędzić czas i moc obliczeniową komputera.

W przypadku k=50 oczywiście nie zmienia się niska precyzja poszczególnych pomiarów – sposób ich obliczania jest wspólny dla obu przypadków. Natomiast dla każdej funkcji widoczne jest to, że nawet dla niskich n, program dość szybko aproksymuje oczekiwaną wartość. Zwiększanie liczby losowych punktów ponad około 200 nie daje nam praktycznie żadnych korzyści.

Wnioski

Zwiększanie k jest skuteczniejsze niż analogiczne działanie na n. Przyjmijmy jako wartość podstawową, że k=5 i n=50. Tak jak jest to widoczne na wykresach zwiększenie k 10-krotnie daje nam dokładny wynik nawet dla niskich wartości n. Jednakże, żeby osiągnąć taki sam efekt poprzez mnożenie n potrzeba 30-40 krotnie większej liczby punktów.

Istnieje optymalna para (n,k), która szybko znajduję dokładną aproksymacje dla relatywnie niskiego n i k.

Co można poprawić?

Gdybym miał po raz kolejny wykonywać ten eksperyment na pewno zdecydowałbym się na język c++ połączony z pythonem – w ten sposób tworzenie wykresów byłoby o wiele mniej czasochłonne.

Kajetan Plewa INA SEM3