**Системи за Паралелна Обработка**

**Изготвил:** Михаел Василев Димитров, фн. 80806, 3 курс, КН

**Ръководител:** ас. Христо Христов

Проверил: …….……....

Pi Generator

Пресмятане на Pi използвайки сходящия ред - Ramanujan 1

# Съдържание

[Съдържание](#h.d6pwl0o67145)

[Условие](#h.5a5lgtb8reih)

[Задача](#h.iqj63bhpi3hv)

[Цел](#h.f885llwjwvvj)

[Реализация](#h.tuyzhh4izduq)

[Паралелизъм](#h.xe5kqzigoc1)

[Алгоритъм](#h.bla5old0xzn1)

[Mетричните показатели](#h.uqdcmfswz0dn)

[Тестови замервания](#h.g34yvgro91d5)

## Условие

Използвайки сходящи редове, можем да сметнем стойността на с произволно висока точност. Един от бързо сходящите към редове е този, открит от индийския математик **Srinivasa Ramanujan** през 1910-1914 година. За стойността на **Pi** имаме:

## 

### Задача

Да се напише програма за генериране на апроксимаций на числото **Pi** използвайки цитирания ред, която използва паралелни процеси и осигурява пресмятане на **pi** със зададена от потребителя точност.

### Цел

Целта на програмата е да измерим ускорението на паралелната програма. Като за целта ще сравняваме серийната (непаралелна) версия на програмата със паралелната.

## Реализация

Програмата приема като парамети :

**-p** - точност на приближението, като брой членове на реда

**-t** - брой процеси, които да се използват. При **‘-t 1’** програмата е серийна

**-h** - информация за останалите параметри и техните стойности по подразбиране

### Паралелизъм

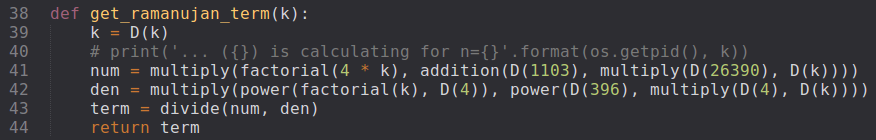
Паралелизъм в програмата се осъществява, чрез **multiprocessing** подхода, защото задачата използва интензивно процесора и реализация с нишки няма да е ефективна.

### Алгоритъм

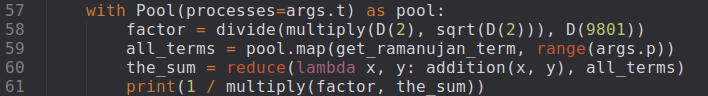
Нека с **Tk** означим **k**-тия член на сходящия ред от условието**.**

Тогава едно приближение на **pi** с точност **n** ще получим като умнижим сбора от всички **Tк, k = 0,...,n** с константата от формулата. повдигнем полученият резултат на **-1** степен

Реализиране принципа за декомпозиция “разделяй и владей” като всеки породен процес пресмята **Tk** за дадено **k** по показния по-долу начин:



1. (59): След като се прочетат подадените аргументи програмата генерира списък от стойности за **k** от 0 до подадената точност
2. (57): Главния процес поражда процеси с помоща на обекта **Pool**.
3. (59): метода **pool.map()** раздробява списъка на части, които предава на породените процеси като отделни задачи.
4. (59): Всеки проден процес изпълнява функцията за пресмятане на **Tk** в/у някоя стойност от списъка(споделената памет).
5. (60): След като завършат всички задачи, в процеса извикал **Pool** се завършва пресмятането на приближението и се извежда резултат



всички приближения на Pi се смятат с 1000 знака след десетичната запетая

### Mетричните показатели

Програмата е разработена на **Python3.4** и е тествана на процесор:

**Intel(R) Xeon(R) CPU X5650 @ 2.67GHz x 16**

Целта е да се оцени ускорението **S(забързване, speed-up)** и ефективността **E(efficiency)** на описания алгоритъм, където ако **Т(p)** e времето необходимо за завършване с **p** наброй процеса, то:

* **S(p) = T(1)/T(p)**
* **E(p) = S(p)/P**

### Тестови замервания

замерванията са извършени със следните входни параметри:

python3.4 calculate.py -p 1000 -t 1 -q  
python3.4 calculate.py -p 1000 -t 2 -q  
python3.4 calculate.py -p 1000 -t 4 -q  
python3.4 calculate.py -p 1000 -t 6 -q  
python3.4 calculate.py -p 1000 -t 8 -q  
python3.4 calculate.py -p 1000 -t 12 -q  
python3.4 calculate.py -p 1000 -t 16 -q  
python3.4 calculate.py -p 1000 -t 18 -q  
python3.4 calculate.py -p 1000 -t 20 -q  
python3.4 calculate.py -p 1000 -t 28 -q  
python3.4 calculate.py -p 1000 -t 32 -q

