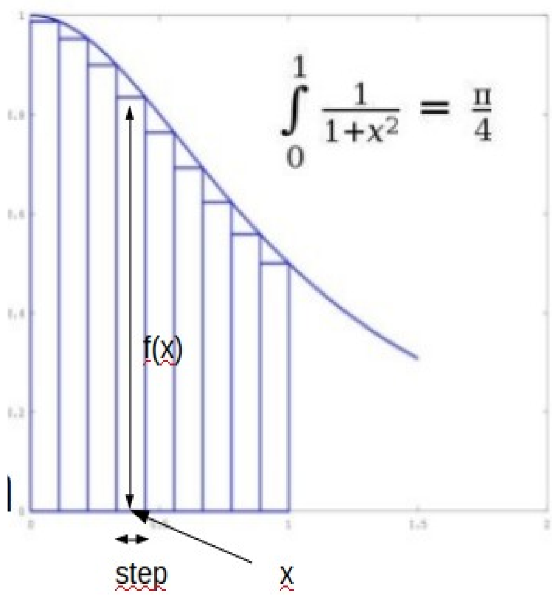
Παράλληλος και Κατανεμημένος Υπολογισμός

Εργαστήριο 5

Υπολογισμός του π με αριθμητική ολοκλήρωση

Αφού μελετήσετε τον Code/VecSum εφαρμόστε τις αντίστοιχες μεθόδους σε ένα πρόβλημα αριθμητικής ολοκλήρωσης, στον υπολογισμό του π. (Ακολουθιακός κώδικας στο φάκελλο NumInt).

Η ιδέα του υπολογισμού του π με αριθμητική ολοκλήρωση φαίνεται παρακάτω. Χωρίζουμε το πεδίο τιμών σε numSteps μικρά τμήματα, με μήκος step. Στο μέσο κάθε τμήματος υπολογίζουμε τα x, f(x) και αθροίζουμε τις τιμές των εμβαδών step*f(x) για να υπολογίσουμε το π/4.




```
step = 1.0 / numSteps;
for (i=0; i < numSteps; ++i) {
            x = ((double)i+0.5)*step;
            sum += 4.0/(1.0+x*x);
}
pi = sum * step;
```

f(x)

Μελετήστε τον ακολουθιακό κώδικα στο φάκελλο NumInt. Εκτελέστε πειράματα με τον ακολουθιακό κώδικα και χρονομέτρηση για να καταλάβετε τη συμπεριφορά του κώδικα, δηλαδή να διαπιστώσετε ότι ο χρόνος εκτέλεσης είναι ευθέως ανάλογος της τιμής numSteps.

Στη συνέχεια με βάση το κώδικα στο Code/VecSum υλοποιήστε ορισμένα από τα σχήματα απεικόνισης-αναγωγής και συγκρίνετε τους χρόνους εκτέλεσης.

Δοκιμάστε τουλάχιστο τα παρακάτω:
1. μια μέθοδο χωρίς αμοιβαίο αποκλεισμό – αλλά με παραγωγή ορθού αποτελέσματος
2. μια μέθοδο με αμοιβαίο αποκλεισμό, χρήση locks
3. μια μέθοδο με αμοιβαίο αποκλεισμό, χρήση synchronized – όχι synchronized(getClass)! *
4. το σχήμα με λανθασμένη χρήση του αμοιβαίου αποκλεισμού για να διαπιστώσετε τις διαφορές χρόνου με την ορθή χρήση του αμοιβαίου αποκλεισμού.

** Όσοι/όσες προγραμματίζουν σε άλλες γλώσσες πλην της Java: να χρησιμοποιήσουν δομές ή κλάσεις, όχι καθολικές μεταβλητές.*

*** Για όσους/όσες ενδιαφέρονται: ο σύνδεσμος* *https://en.wikipedia.org/wiki/Approximations_of_%CF%80*

παραθέτει δεκάδες αλγορίθμους υπολογισμού του π. Δείτε για παράδειγμα τη μέθοδο Monte Carlo (επιφάνεια τεταρτημορίου κύκλου, δείτε το σχετικό animation στον παραπάνω σύνδεσμο) και σκεφτείτε τη παραλληλοποίησή του. Θα βρείτε πολλές υλοποιήσεις στο Διαδίκτυο.