Simulation von π

Dickbauer Y., Moser P., Perner M.

PS Computergestützte Modellierung, WS 2016/17

December 15, 2016

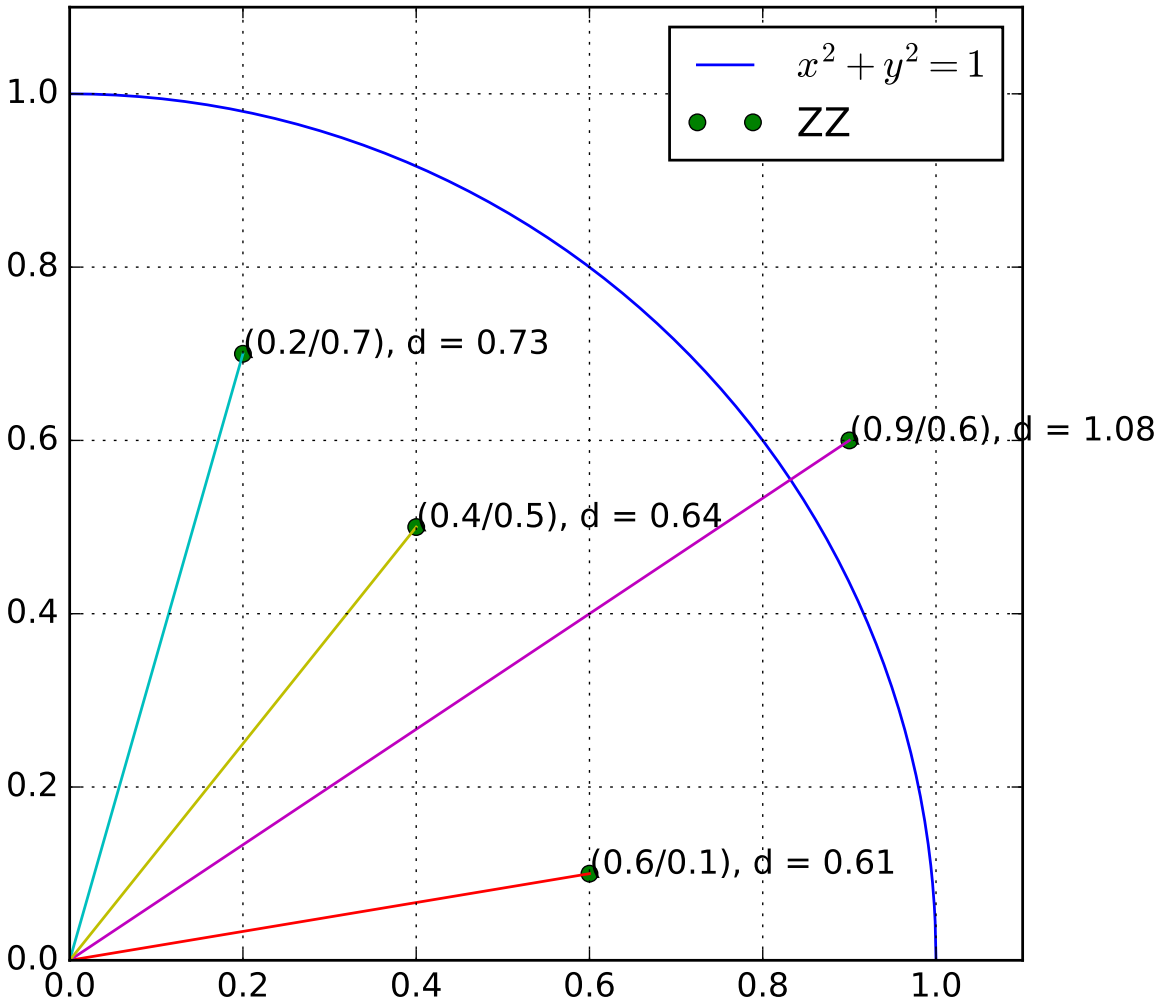
Outline

Aufgabenstellung

Berechnen Sie näherungsweise mittels Simulation die Zahl π (3.14159)
(Hinweis: Einheits(viertel)kreis)

- Eingabe: Anzahl an Punkten
- Output: Pi

Grafische Darstellung anhand des Einheitskreises



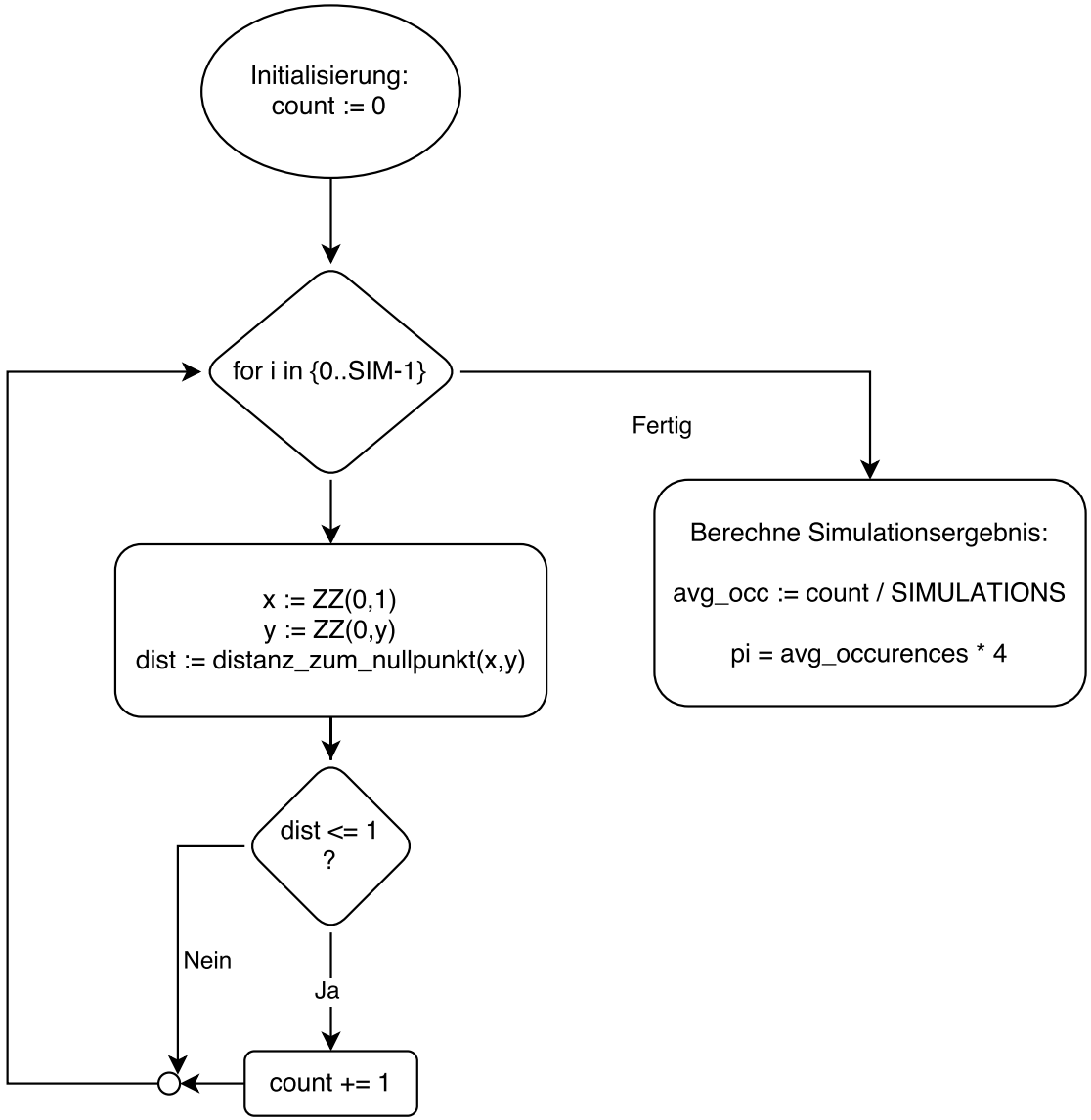
Fläche des Viertelkreises

$$\frac{1}{4} * A_{Kreis}$$
$$=$$
$$\frac{1}{4} * 1^2 * \pi$$

WSK, dass ZZ innerhalb

$$p = \frac{\pi}{4}$$

Flow Chart


Initialisierung:
count := 0
for i in {0..SIM-1}
Fertig
Berechne Simulationsergebnis:
avg_occ := count / SIMULATIONS
pi = avg_occurences * 4
x := ZZ(0,1)
y := ZZ(0,y)
dist := distanz_zum_nullpunkt(x,y)
dist <= 1
?
Nein
Ja
count += 1

Main Funktion - Programmeinstieg

```
1   def main():
2       # user input:
3       simulations = user_input((
4           ('Number of simulations', int, 10000), ),DEBUG)[0]
5
6       count = 0 # how often did we shoot in the unit circle
7
8       for i in range(simulations):
9           # create two randoms representing cords in a [1,1] rectangle
10          x = random_number_from_interval(0,1)
11          y = random_number_from_interval(0,1)
12          # check wheter these cords are lying within or beyond the unit circle
13          if euclidean_distance((0, 0), (x, y)) <= 1:
14              count += 1
15
16      # avg_occurrences p should be pi/4 / 1 --> 4 * avg = pi
17      avg_occurrences = count/simulations
18      pi = avg_occurrences * 4
19      print('Pi is simulated:', pi)
20      print('Difference:     {:+.5f} %'.format((pi/REFERENCE_PI - 1) * 100))
```

Funktion euclidean_distance(p1, p2)

- Diese Funktion verlangt zwei Punkte (x1, y1) (x2, y2) als Eingabeparameter
- Gibt die eukliedsche Distanz zurück

```
1   def euclidean_distance(point_1, point_2):
2       """
3           Calculates the euclidean distance between two points
4
5           point_1: a tuple of (x,y) values
6           point_2: a tuple of (x,y) values
7       """
8       delta_x = point_2[0] - point_1[0]
9       delta_y = point_2[1] - point_1[1]
10      return (delta_x ** 2 + delta_y ** 2) ** 0.5
```

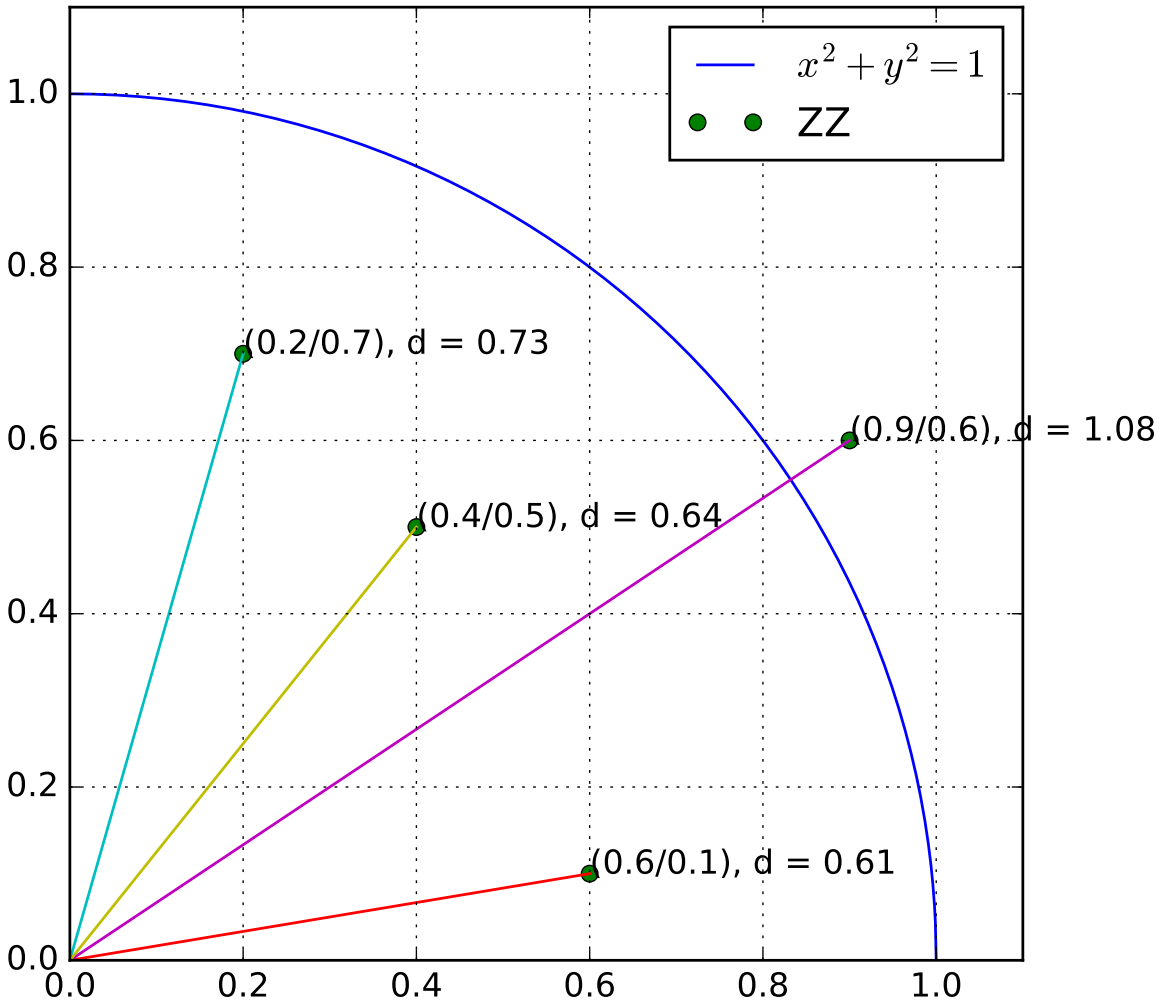
Beispiel anhand fixer Zufallszahlen

- Annahme der Zufallszahlen wie folgt:

iteration	0	1	2	3
x	0.6	0.2	0.9	0.4
y	0.1	0.7	0.6	0.5
dist	0.61	0.73	1.08	0.64
Im Kreis?	1	1	0	1

count $= 3$
iterations $= 4$
avg_occ $= \frac{3}{4}$
pi_sim $= 4\frac{3}{4} = 3.0$
abweichung: -4.5

Grafische Darstellung des Beispiels



Anhang: Modifikation des Source Codes um Demo Beispiel zu erhalten

```
1     # Fuege folgenden Code vor der main() Funktion ein:
2   ZZ = [0.6, 0.1, 0.2, 0.7, 0.9, 0.6, 0.4, 0.5] * 1000
3   i = -1
4   def random_number_from_interval(x,y):
5       global i
6       i += 1
7       return ZZ[i]
```