fundamentals of simulation methods

winter term 2019/2020

Lecturers: Ralf Klessen

Tutors: Loke Lönnblad Ohlin, Toni Peter, Marcelo Barraza

mvcomp1

https://uebungen.physik.uni-heidelberg.de/vorlesung/20192/1062

- Homework assignments provided.
- Results and grades listed.

https://elearning2.uni-heidelberg.de/enrol/index.php?id=22625

- Enrollment key: To enrollment key for this lecture is "**von Neumann**”.
- Homework and further information (script) provided.

mvcomp1

Goals: After completion of this module, the students are endowed with the capacity to identify and classify numerical problems. They have reached active understanding of applicable numerical methods and algorithms. They are able to solve basic physical problems with adequate numerical techniques and to recognize the range of validity of numerical solutions.

Contents (from module description): Basic concepts of numerical simulations, continuous and discrete simulations

- Discretization of ordinary differential equations, integration schemes of different order
- N-body problems, molecular dynamics, collisionless systems
- Discretization of partial differential equations
- Finite element and finite volume methods
- Lattice methods
- Adaptive mesh refinement and multi-grid methods
- Matrix solvers and FFT methods
- Monte Carlo methods, Markov chains, applications in statistical physics

Module parts and teaching methods:
- Lecture on “Fundamentals of Simulation Methods” (4 hours/week)
- Exercise with homework (2 hours/week)

Workload and credit points: The workload for this module is 240 hours, corresponding to 8 credit points.

mvcomp1

Practice groups

- **Group G1** (Loke Ohlin Lönnblad)
  Phil 12 -- CIP Pool, Thu 11 - 13

- **Group G2** (Toni Peter)
  Phil 12 -- CIP Pool, Thu 14 - 16

- **Group G3** (Marcelo Barraza)
  Phil 12 -- CIP Pool, Fri 11 - 13

Homework

- Provided on Tuesday
- Hand in by Wednesday noon the following week
- Discussed and given back in tutorials after one more week.

mvcomp1

Schedule

WEEK	HW out	HW in	TUTORIALS	COMMENTS
14.10.	HW1	--	--	
21.10.	HW2	HW1	general discussion	
28.10.	HW3	HW2	HW1 discussion	
04.11.	HW4	HW3	HW2 discussion	
11.11.	HW5	HW4	HW3 discussion	
18.11.	HW6	HW5	HW4 discussion	
25.11.	HW7	HW6	HW5 discussion	
02.12.	HW8	HW7	HW6 discussion	
09.12.	HW9	HW8	HW7 discussion	
16.12.	HW10	HW9	HW8 discussion	
23.12.	....			Christmas
30.12.	....			Christmas
06.01.	HW11	HW10	HW9 discussion	
13.01.	HW12	HW11	HW10 discussion	
20.01.	--	HW12	HW11 discussion	
27.01.	--	--	HW12 discussion	
03.02.	--	--	--	Exam Week

ASCII (American Standard Code for Information Interchange)

Bits												
b_7					0	0	0	0	1	1	1	1
b_6					0	0	1	1	0	0	1	1
b_5					0	1	0	1	0	1	0	1
b_4 ↓	b_3 ↓	b_2 ↓	b_1 ↓	Row ↓ / Column →	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FC	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

1000100 = &#65 = A

INTEGER

example 4 bit integer:

- this is the standard representation!
- unique definition of zero
- attention: overflow!!

Decimal Representation	Unsigned Representation	Signed-Magnitude Representation	Ones Complement Representation	Twos-Complement Representation	Biased Representation
+8	1000	—	—	—	1111
+7	0111	0111	0111	0111	1110
+6	0110	0110	0110	0110	1101
+5	0101	0101	0101	0101	1100
+4	0100	0100	0100	0100	1011
+3	0011	0011	0011	0011	1010
+2	0010	0010	0010	0010	1001
+1	0001	0001	0001	0001	1000
+0	0000	0000	0000	0000	0111
-0	—	1000	1111	—	—
-1	—	1001	1110	1111	0110
-2	—	1010	1101	1110	0101
-3	—	1011	1100	1101	0100
-4	—	1100	1011	1100	0011
-5	—	1101	1010	1011	0010
-6	—	1110	1001	1010	0001
-7	—	1111	1000	1001	0000
-8	—	—	—	1000	—

INTEGER

Größe (Bit)	Typische Namen	Vorzeichen	Grenzen des Wertebereichs (Zweierkomplement)		Dezimalstellen (ohne Vorzeichen)
			min	max	
8	char, Byte/byte	signed	−128	127	3
		unsigned	0	255	3
16	Word, Short/short, Integer	signed	−32.768	32.767	5
		unsigned	0	65.535	5
32	DWord/Double Word, int, long (Windows auf 16/32/64-Bit Systemen;[5] Unix/Linux/C99 auf 16/32-Bit Systemen[5])	signed	−2.147.483.648	2.147.483.647	10
		unsigned	0	4.294.967.295	10
64	Int64, QWord/Quadword, long long, Long/long (Unix/Linux/C99 auf 64-Bit Systemen[5][6][7])	signed	−9.223.372.036.854.775.808	9.223.372.036.854.775.807	19
		unsigned	0	18.446.744.073.709.551.615	20
128	Int128, Octaword, Double Quadword	signed	$\approx -1{,}70141 \cdot 10^{38}$	$\approx 1{,}70141 \cdot 10^{38}$	39
		unsigned	0	$\approx 3{,}40282 \cdot 10^{38}$	39
n	BigInteger	signed	-2^{n-1}	$2^{n-1} - 1$	$\lceil \log_{10} 2^{n-1} \rceil$
		unsigned	0	$2^{n} - 1$	$\lceil \log_{10} 2^{n} \rceil$

FLOATING POINT NUMBERS

number representation:
s = significant (integer representation of mantissa)
b = base (usually 2)
e = exponent

$$\boxed{s \times b^e \qquad\qquad \frac{s}{b^{p-1}} \times b^e}$$

FLOATING POINT (Decimals) Organization depends on word length. For **32 bit**:

$$\boxed{\begin{aligned} f &= (d_0, d_1, d_2, \ldots, d_{23}) \cdot 2^e \\ &\equiv (d_0 \cdot 2^0 + d_1 \cdot 2^{-1} + d_2 \cdot 2^{-2} + \ldots + d_{23} \cdot 2^{-23}) \cdot 2^e \ . \end{aligned}}$$

with e an 8 bit integer, $e = e_0 - 127$, and $e_0 = 0$ and $e_0 = 255$ represent $f = 0$ and $f = \infty$; so for finite non-zero f $-126 \leq e \leq 127$. Convention: $d_0 \equiv 1$ for normalization. thus d_0 free as sign bit.

– largest representable f

$$f_{\max} = (1, 1, 1, 1, \ldots, 1) \cdot 2^{127} \simeq 3.40 \cdot 10^{38} \ .$$

– smallest representable f

$$f_{\min} = (1, 0, 0, \ldots, 0) \cdot 2^{-126} \simeq 1.18 \cdot 10^{-38}$$

– machine precision ε_m: smallest Increment of mantissa $\varepsilon_m = 2^{-23} \simeq 1.19 \cdot 10^{-7}$, consequence:

$$1 + \varepsilon = 1$$

for $\varepsilon < \varepsilon_m$!

FLOATING POINT NUMBERS

number representation:

s = significant (integer representation of mantissa)
b = base (usually 2)
e = exponent

$$s \times b^e \qquad \frac{s}{b^{p-1}} \times b^e$$

floating point components:

	Sign	Exponent	Fraction
Single Precision	1 [31]	8 [30–23]	23 [22–00]
Double Precision	1 [63]	11 [62–52]	52 [51–00]

Single: SEEEEEEE EMMMMMMM MMMMMMMM MMMMMMMM
Double: SEEEEEEE EEEEMMMM MMMMMMMM MMMMMMMM MMMMMMMM MMMMMMMM MMMMMMMM MMMMMMMM

ambiguity in number representation: →

example 5 with 4 significant digits:
$.5000 \times 10^{2}$
0.050×10^{3}
$5000. \times 10^{-2}$

normalized representation:

5.000×10^{0}

FLOATING POINT NUMBERS

number representation:
s = significant (integer representation of mantissa)
b = base (usually 2)
e = exponent

$$s \times b^e \qquad\qquad \frac{s}{b^{p-1}} \times b^e$$

floating point components:

	Sign	Exponent	Fraction
Single Precision	1 [31]	8 [30–23]	23 [22–00]
Double Precision	1 [63]	11 [62–52]	52 [51–00]

floating point range:

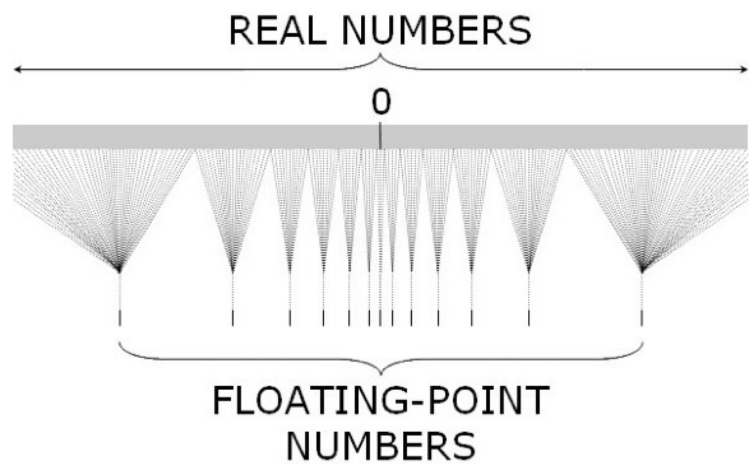
	Denormalized	*Normalized*	*Approximate Decimal*
Single Precision	$\pm\, 2^{-149}$ *to* $(1-2^{-23})\times 2^{-126}$	$\pm\, 2^{-126}$ *to* $(2-2^{-23})\times 2^{-127}$	$\pm \approx 10^{-44.85}$ *to* $\approx 10^{38.53}$
Double Precision	$\pm\, 2^{-1074}$ *to* $(1-2^{-52})\times 2^{-1022}$	$\pm\, 2^{-1022}$ *to* $(2-2^{-52})\times 2^{-1023}$	$\pm \approx 10^{-323.3}$ *to* $\approx 10^{308.3}$

FLOATING POINT NUMBERS

number representation:
s = significant (integer representation of mantissa)
b = base (usually 2)
e = exponent

$$\boxed{s \times b^e \qquad\qquad \frac{s}{b^{p-1}} \times b^e}$$

IMPORTANT not all numbers can be represented!



Example: 0.1 = 1/10 in decimal, but there is no “finite” representation in binary:

$e = -4$; s = 1100110011001100110011001100110011...,

where, as previously, s is the significand and e is the exponent.

When rounded to 24 bits this becomes

$e = -4$; s = 110011001100110011001101,

which is actually 0.100000001490116119384765625 in decimal.

As a further example, the real number π, represented in binary as an infinite series of bits is

11.0010010000111111011010101000100010000101101000110000100011010011...

but is

11.0010010000111111011011

when approximated by rounding to a precision of 24 bits.

FLOATING POINT NUMBERS

number representation:
s = significant (integer representation of mantissa)
b = base (usually 2)
e = exponent

$$s \times b^e \qquad \frac{s}{b^{p-1}} \times b^e$$

Example: π

$$\pi = \left(1 + \sum_{n=1}^{p-1} \mathrm{bit}_n \times 2^{-n}\right) \times 2^e$$
$$= \left(1 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-4} + 1 \times 2^{-7} + \cdots + 1 \times 2^{-23}\right) \times 2^1$$
$$= 1.5707964 \times 2$$

In binary single-precision floating-point, this is represented as s = 1.10010010000111111011011 with e = 1. This has a decimal value of

3.1415927410125732421875,

whereas a more accurate approximation of the true value of π is

3.14159265358979323846264338327950...

Adding floating point numbers

A simple method to add floating-point numbers is to first represent them with the same exponent. In the example below, the second number is shifted right by three digits, and we then proceed with the usual addition method:

```
123456.7 = 1.234567 × 10^5
101.7654 = 1.017654 × 10^2 = 0.001017654 × 10^5
```

```
Hence:
123456.7 + 101.7654 = (1.234567 × 10^5) + (1.017654 × 10^2)
                    = (1.234567 × 10^5) + (0.001017654 × 10^5)
                    = (1.234567 + 0.001017654) × 10^5
                    =  1.235584654 × 10^5
```

In detail:

```
  e=5;  s=1.234567     (123456.7)
+ e=2;  s=1.017654     (101.7654)
```

```
  e=5;  s=1.234567
+ e=5;  s=0.001017654  (after shifting)
--------------------
  e=5;  s=1.235584654  (true sum: 123558.4654)
```

number representation:
s = significant (integer representation of mantissa)
b = base (usually 2)
e = exponent

$$s \times b^e \qquad\qquad \frac{s}{b^{p-1}} \times b^e$$

Computing with floating point numbers

Floating point numbers are *commutative*:
a + b = b + a and **a * b = b * a**

BUT, they are not necessarily *associative*
a + (b + c) = (a + b) + c

```
a = 1234.567, b = 45.67834, c = 0.0004
```

```
(a + b) + c:
    1234.567   (a)
  +   45.67834 (b)
  ____________
    1280.24534   rounds to   1280.245
```

```
   1280.245  (a + b)
  +   0.0004 (c)
  ____________
   1280.2454   rounds to  1280.245 <--- (a + b) + c
```

```
a + (b + c):
  45.67834 (b)
+  0.0004  (c)
____________
  45.67874
```

```
    45.67874 (b + c)
+ 1234.567   (a)
____________
  1280.24574   rounds to  1280.246<--- a + (b + c)
```

Computing with floating point numbers

AND, they are not necessarily *distributive*

(a + b) * c = a*c + b*c

```
1234.567 × 3.333333 = 4115.223
1.234567 × 3.333333 = 4.115223
                      4115.223 + 4.115223 = 4119.338
but
1234.567 + 1.234567 = 1235.802
                      1235.802 × 3.333333 = 4119.340
```

Computing with floating point numbers

Example: solution of quadratic formula

Solution of $ax^2 + bx + c = 0$:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

If $ac \ll b^2$, so $b^2 - 4ac = b^2$ due to finite machine prec., computer gives $x_1 = 0$ instead of

$$x_1 = \frac{-b + b\sqrt{1 - 4ac/b^2}}{2a} \simeq \frac{-b + b(1 - 2ac/b^2 + \ldots)}{2a} \simeq -\frac{c}{b}$$

which can be $\mathcal{O}(1)$!

Take round-off errors into account in your calculations!

Example: **Archimedes approach to calculate π**

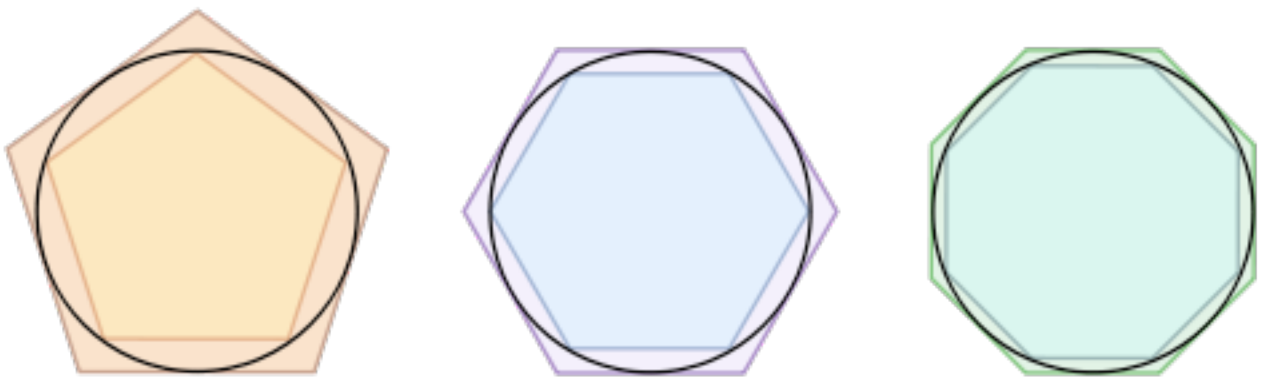
Example: calculation of π

$$t_0 = \frac{1}{\sqrt{3}}$$

first form : $$t_{i+1} = \frac{\sqrt{t_i^2+1}-1}{t_i}$$ second form : $$t_{i+1} = \frac{t_i}{\sqrt{t_i^2+1}+1}$$

$$\pi \sim 6 \times 2^i \times t_i, \qquad \text{converging as i} \to \infty$$

Here is a computation using IEEE "double" (a significand with 53 bits of precision) arithmetic:



Take round-off errors into account in your calculations!

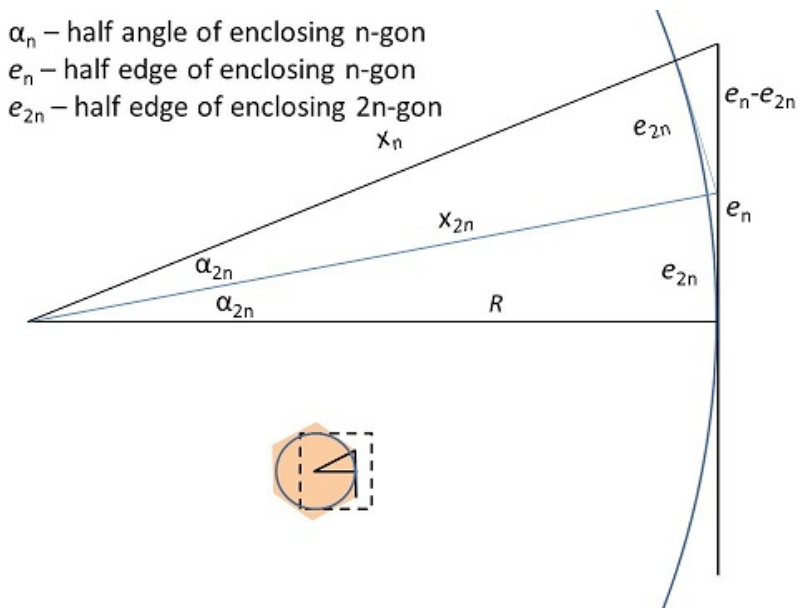
Example: **Archimedes approach to calculate π**

$$t_0 = \frac{1}{\sqrt{3}}$$

$$\text{first form :} \qquad t_{i+1} = \frac{\sqrt{t_i^2+1}-1}{t_i} \qquad \text{second form :} \qquad t_{i+1} = \frac{t_i}{\sqrt{t_i^2+1}+1}$$

$$\pi \sim 6 \times 2^i \times t_i, \qquad \text{converging as i} \to \infty$$

Here is a computation using IEEE "double" (a significand with 53 bits of precision) arithmetic:



$$\begin{aligned} x_n/R &= (e_n - e_{2n})/e_{2n} \\ (x_n+R)/R &= e_n/e_{2n} \\ (x_n+R)/e_n &= R/e_{2n} \\ x_n^2 &= R^2 + e_n^2 \end{aligned}$$

$$a_n \stackrel{\text{def}}{=} R/e_n$$

$$a_{2n} = a_n + \sqrt{a_n^2+1}$$

recursive formula for a_n

set $t_n = 1/a_n$ to obtain $$\boxed{t_{i+1} = \frac{t_i}{\sqrt{t_i^2+1}+1}}$$

http://www.mathworks.de/matlabcentral/fileexchange/29504-the-computation-of-pi-by-archimedes/content/html/ComputationOfPiByArchimedes.html

1

$$
\begin{aligned}
x_n/R &= (e_n - e_{2n})/e_{2n} & +1 \\
(x_n + R)/R &= e_n/e_{2n} & \cdot R/e_n \\
(x_n + R)/e_n &= R/e_{2n}
\end{aligned}
$$

2

$$x_n^2 = R^2 + e_n^2$$

$$\frac{x_n^2}{e_n^2} = \frac{R^2}{e_n^2} + 1 = \alpha_n^2 + 1$$

3

$$a_n \stackrel{\text{def}}{=} R/e_n$$

$$a_{2n} = \frac{R}{e_{2n}} = \frac{x_n + R}{e_n} = \frac{x_n}{e_n} + \alpha_n$$

4

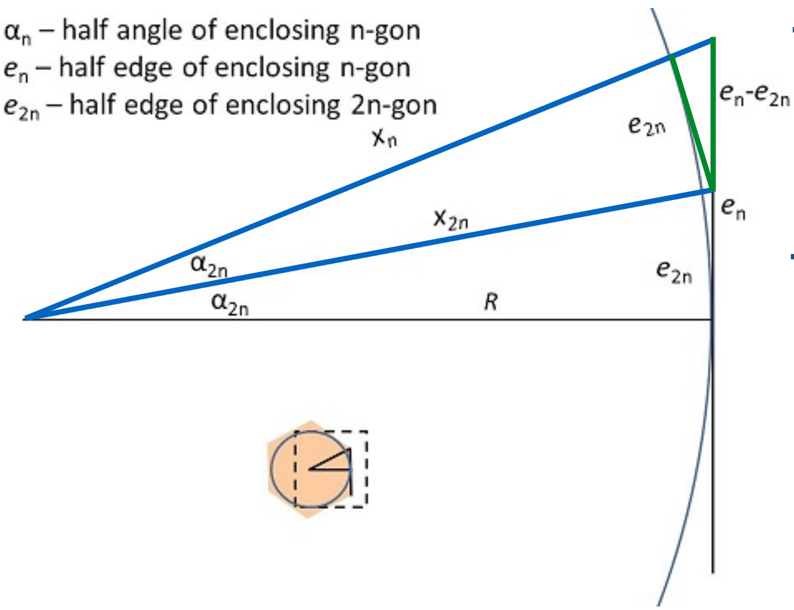
$$\boxed{a_{2n} = a_n + \sqrt{a_n^2 + 1}}$$

recursive formula for a_n

5 set $t_n = 1/a_n$ to obtain

$$\boxed{t_{i+1} = \frac{t_i}{\sqrt{t_i^2 + 1} + 1}}$$

α_n – half angle of enclosing n-gon
e_n – half edge of enclosing n-gon
e_{2n} – half edge of enclosing 2n-gon



http://www.mathworks.de/matlabcentral/fileexchange/29504-the-computation-of-pi-by-archimedes/content/html/ComputationOfPiByArchimedes.html

Take round-off errors into account in your calculations!

Example: **Archimedes approach to calculate π**

$$t_0 = \frac{1}{\sqrt{3}}$$

$$\text{first form :} \quad t_{i+1} = \frac{\sqrt{t_i^2+1}-1}{t_i} \qquad \text{second form :} \quad t_{i+1} = \frac{t_i}{\sqrt{t_i^2+1}+1}$$

$$\pi \sim 6 \times 2^i \times t_i, \qquad \text{converging as i} \to \infty$$

Here is a computation using IEEE "double" (a significand with 53 bits of precision) arithmetic:

i	$6 \times 2^i \times t_i$, first form	$6 \times 2^i \times t_i$, second form
0	3.4641016151377543863	3.4641016151377543863
1	3.2153903091734710173	3.2153903091734723496
2	3.1596599420974940120	3.1596599420975006733
3	3.1460862151314012979	3.1460862151314352708
4	3.1427145996453136334	3.1427145996453689225
5	3.1418730499801259536	3.1418730499798241950
6	3.1416627470548084133	3.1416627470568494473
7	3.1416101765997805905	3.1416101766046906629
8	3.1415970343230776862	3.1415970343215275928
9	3.1415937488171150615	3.1415937487713536668
10	3.1415929278733740748	3.1415929273850979885
11	3.1415927256228504127	3.1415927220386148377
12	3.1415926717412858693	3.1415926707019992125
13	3.1415926189011456060	3.1415926578678454728
14	3.1415926717412858693	3.1415926546593073709
15	3.1415919358822321783	3.1415926538571730119
16	3.1415926717412858693	3.1415926536566394222
17	3.1415810075796233302	3.1415926536065061913
18	3.1415926717412858693	3.1415926535939728836
19	3.1414061547378810956	3.1415926535908393901
20	3.1405434924008406305	3.1415926535900560168
21	3.1400068646912273617	3.1415926535898608396
22	3.1349453756585929919	3.1415926535898122118
23	3.1400068646912273617	3.1415926535897995552
24	3.2245152435345525443	3.1415926535897968907
25		3.1415926535897962246
26		3.1415926535897962246
27		3.1415926535897962246
28		3.1415926535897962246

The true value is 3.14159265358979323846264338327...

some links

some interesting links for the lecture:

floating point numbers:

(1) http://www.h-schmidt.net/FloatConverter/IEEE754.html

further information on Archimedes' approach to calculate pi:

(2) http://www.pbs.org/wgbh/nova/physics/approximating-pi.html

(3) http://betterexplained.com/articles/prehistoric-calculus-discovering-pi/

(4) https://illuminations.nctm.org/Activity.aspx?id=3548