CSc 28
Discrete Structures

Chapter 11
Euler's Number e

Herbert G. Mayer, CSU CSC
Status 1/1/2021

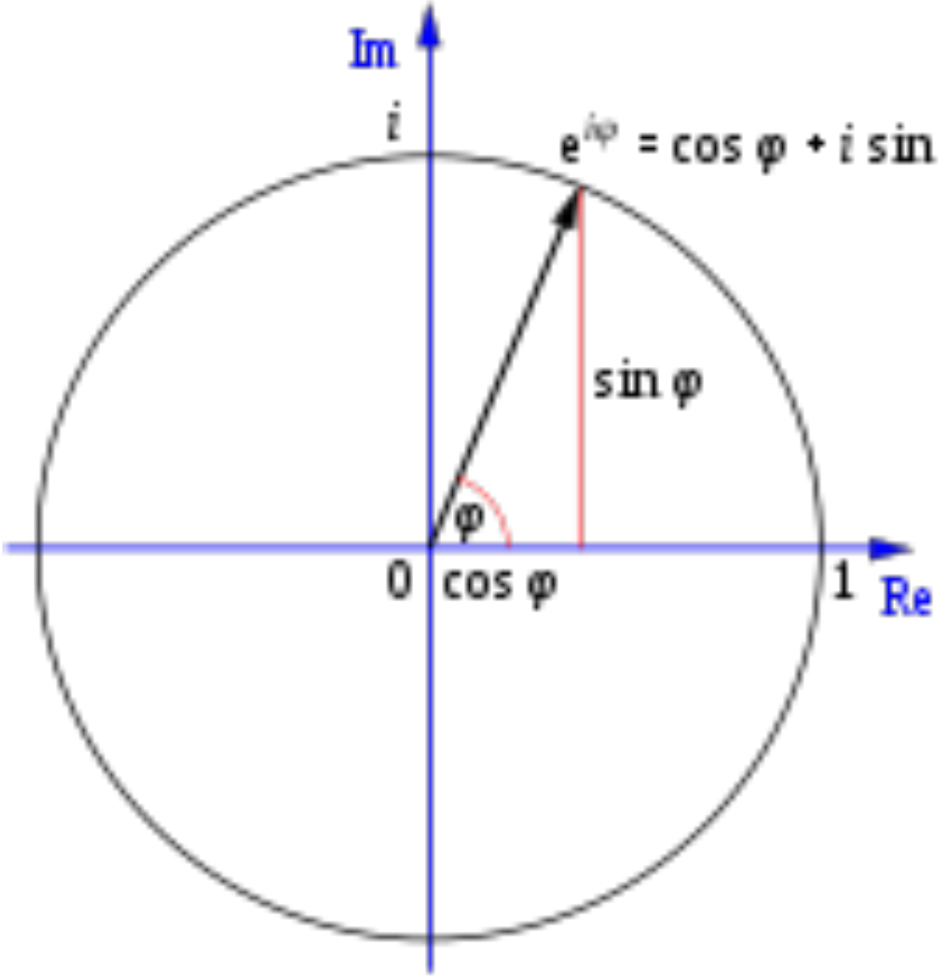
Syllabus

- Constant e
- Leonhard Euler
- Plot of e
- Compute e in C++
- Compute e^x in C++
- Differentiation of f(e^x)
- Summary
- References

Constant e

- The irrational number *e* is a mathematical constant approximately equal to 2.71828 . . . and is the base of the Natural Logarithm
- Irrational means: infinite number of decimal digits
- The number’s symbolic name is: e
- Sometimes e is called the *natural* number, or Euler's number; is an important mathematical constant
- When used as the base for a logarithm, the corresponding logarithm is called the natural logarithm, written as ln(x)

Constant e



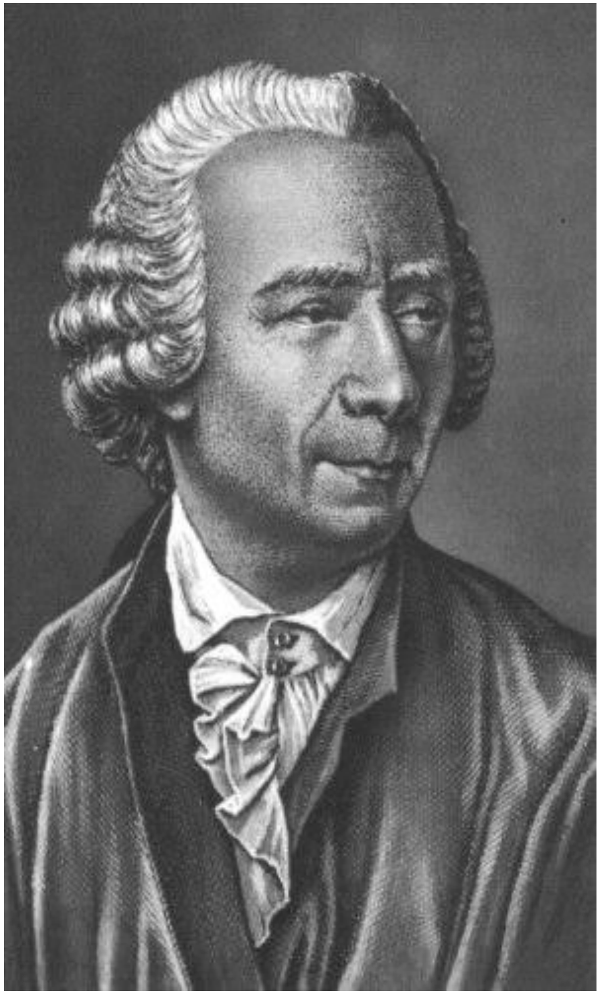
Euler's Formula:

$e^{i\phi} = \cos \phi + i \sin \phi$

in the complex plane

Leonhard Euler

- Detail on Euler's Number:
- Often, math functions, also natural constants can be accurately computed by infinite series
- Many a formula discovered –or publicized– by L. Euler [1], grand mathematician
- Worked with Bernoulli in Switzerland, then in St. Petersburg, and later Berlin



Early Portrait

Daniel Bernoulli

1700 - 1782

Leonhard Euler

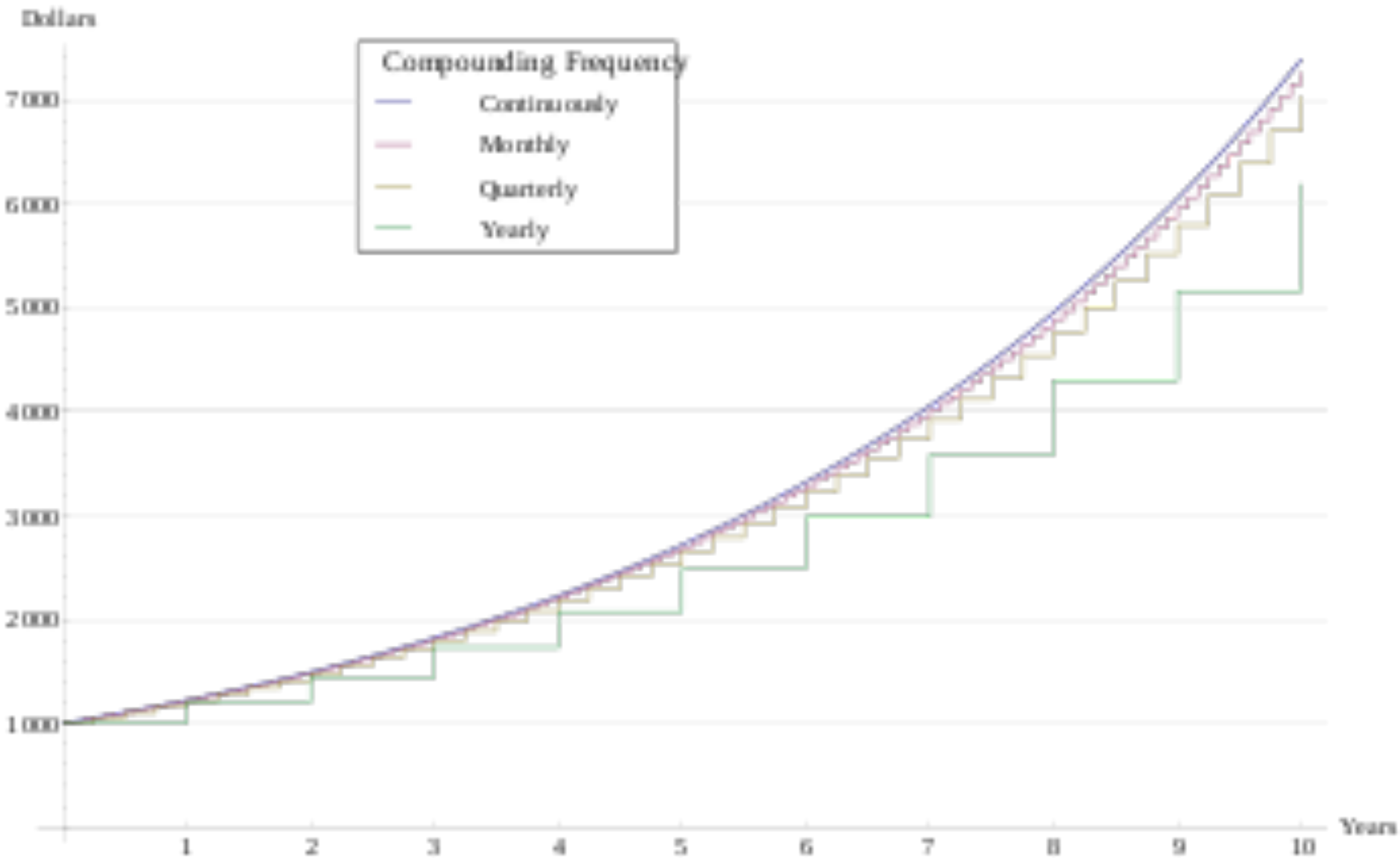
- Leonhard Euler was all of these: Great mathematician, Swiss physicist, astronomer, geographer, Russian researcher, logician and engineer
- Made important and influential discoveries in various branches of mathematics, such as infinitesimal computations
- Born: 1707 Basel, Switzerland
- Died: 1783 St. Petersburg

1753 Portrait of Euler

Constant e

- "e” is a –or *the*– key natural constant!
- Similar to, different from, it’s cousin π = 3.14159…, AKA pi
- Π is a numerical constant derived when a circle’s circumference is divided by its diameter
- e is found in mathematical formulas describing a nonlinear increase or decrease of natural growth, e.g. when computing continual compound interest
- e = 2.718281828459045235360287471352662497 ...
- Pops up in statistical "bell curve”, also in the shape of the hanging cables of suspension bridges, etc.

e in Compound Interest



Earning 20% interest on initial $1,000 at various frequencies

Constant e

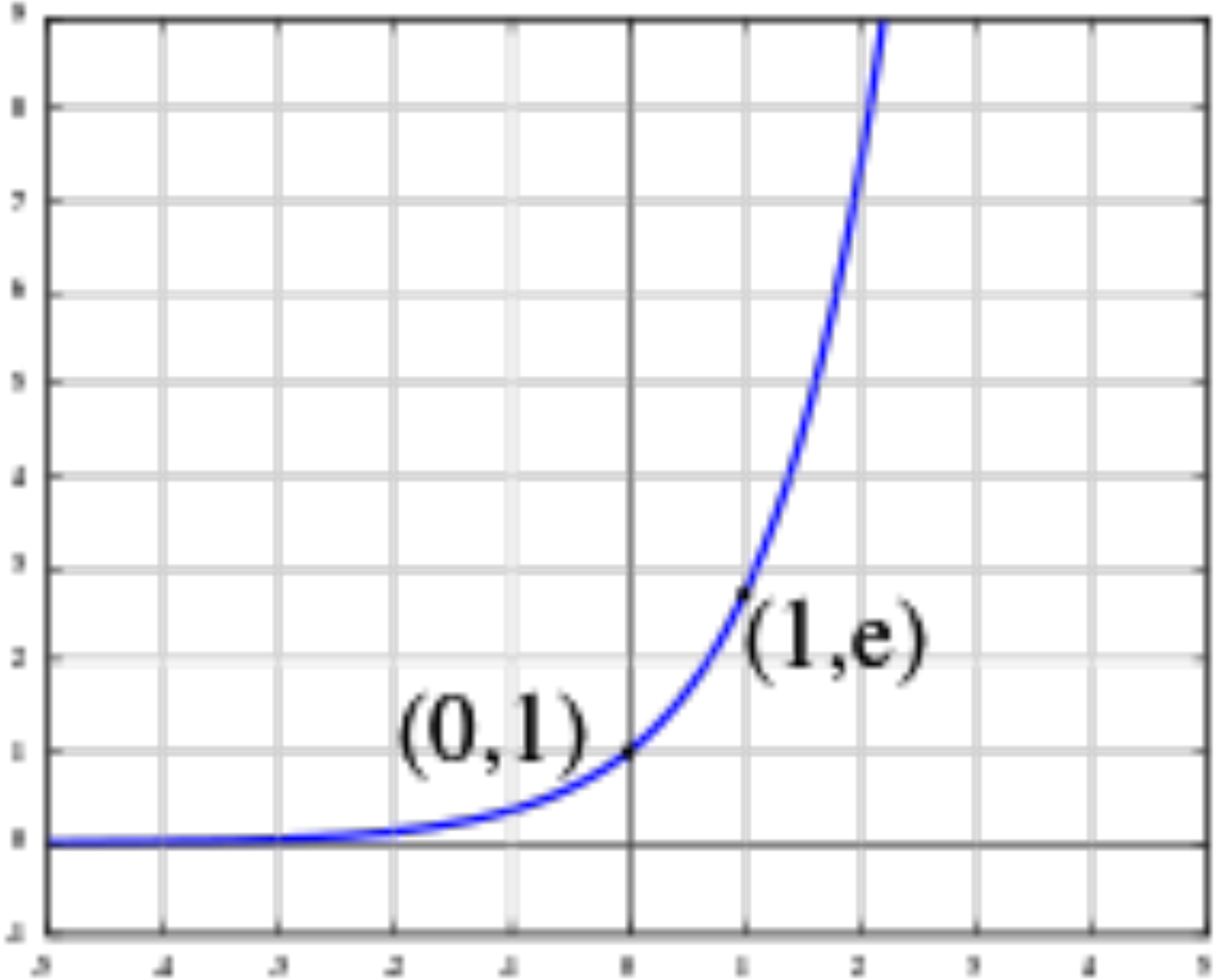
- e surfaces in problems of probability, and in some counting problems
- Even in study of the distribution of prime numbers across the integer spectrum of numbers
- e is the base of natural logarithms; matter of fact that is how natural log is defined ☺
- And is: $e = \Sigma\ 1 / n!$ for $n = 0$ to ∞

Summands for e

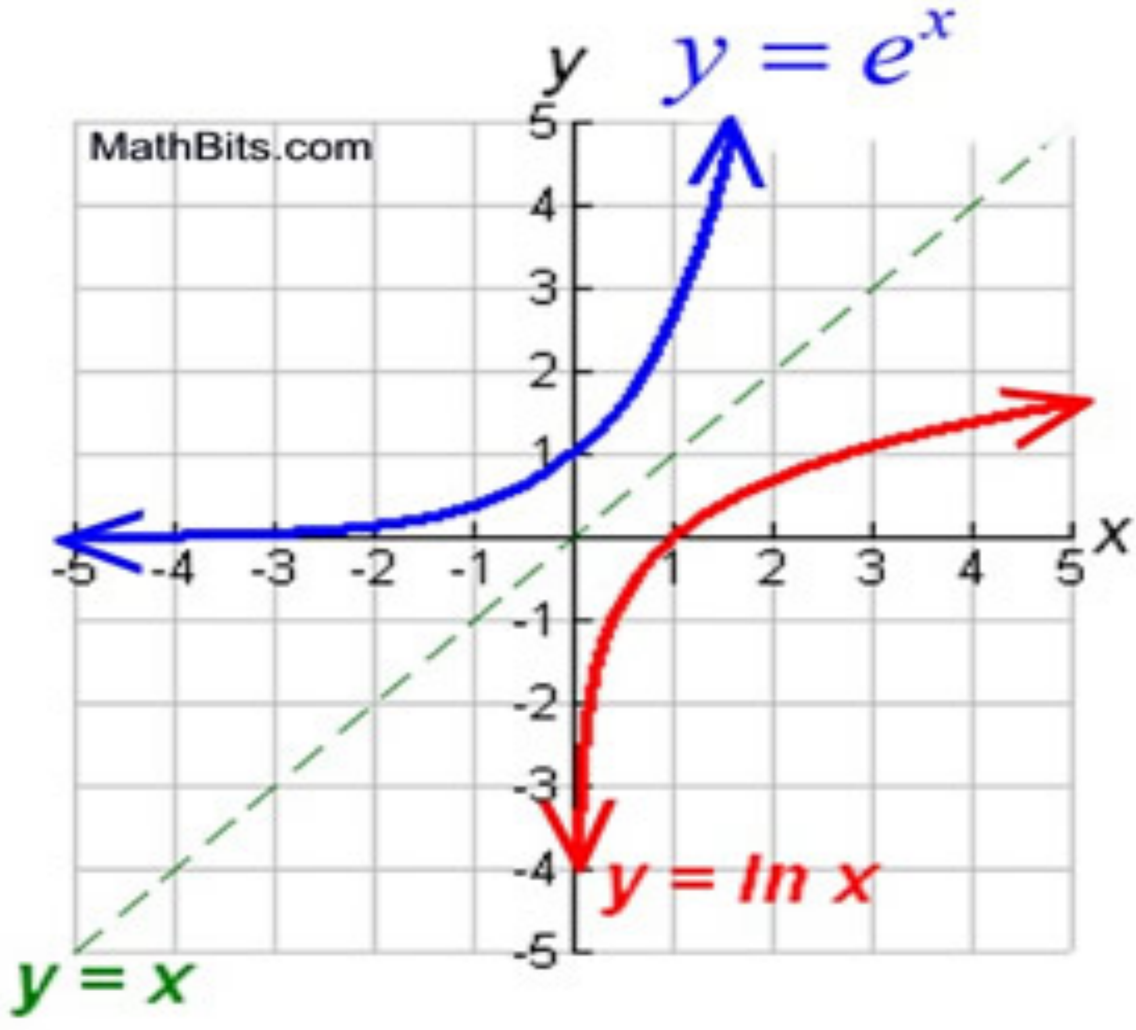
1/0!	=	1/1	=	1.000000000000000000000000
1/1!	=	1/1	=	1.000000000000000000000000
1/2!	=	1/2	=	0.500000000000000000000000
1/3!	=	1/6	=	0.166666666666666666666667
1/4!	=	1/24	=	0.041666666666666666666667
1/5!	=	1/120	=	0.008333333333333333333333
1/6!	=	1/720	=	0.001388888888888888888889
1/7!	=	1/5040	=	0.000198412698412698412698
1/8!	=	1/40320	=	0.000024801587301587301587
1/9!	=	1/362880	=	0.000002755731922398589065
1/10!	=	1/3628800	=	0.000000275573192239858907
1/11!	=	1/39916800	=	0.000000025052108385441719
1/12!	=	1/479001600	=	0.000000002087675698786810
1/13!	=	1/6227020800	=	0.000000000160590438368216
1/14!	=	1/87178291200	=	0.000000000011470745597730
1/15!	=	1/1307674368000	=	0.000000000000764716373182

. . .

Plot of e Function



Plots of e, and ln



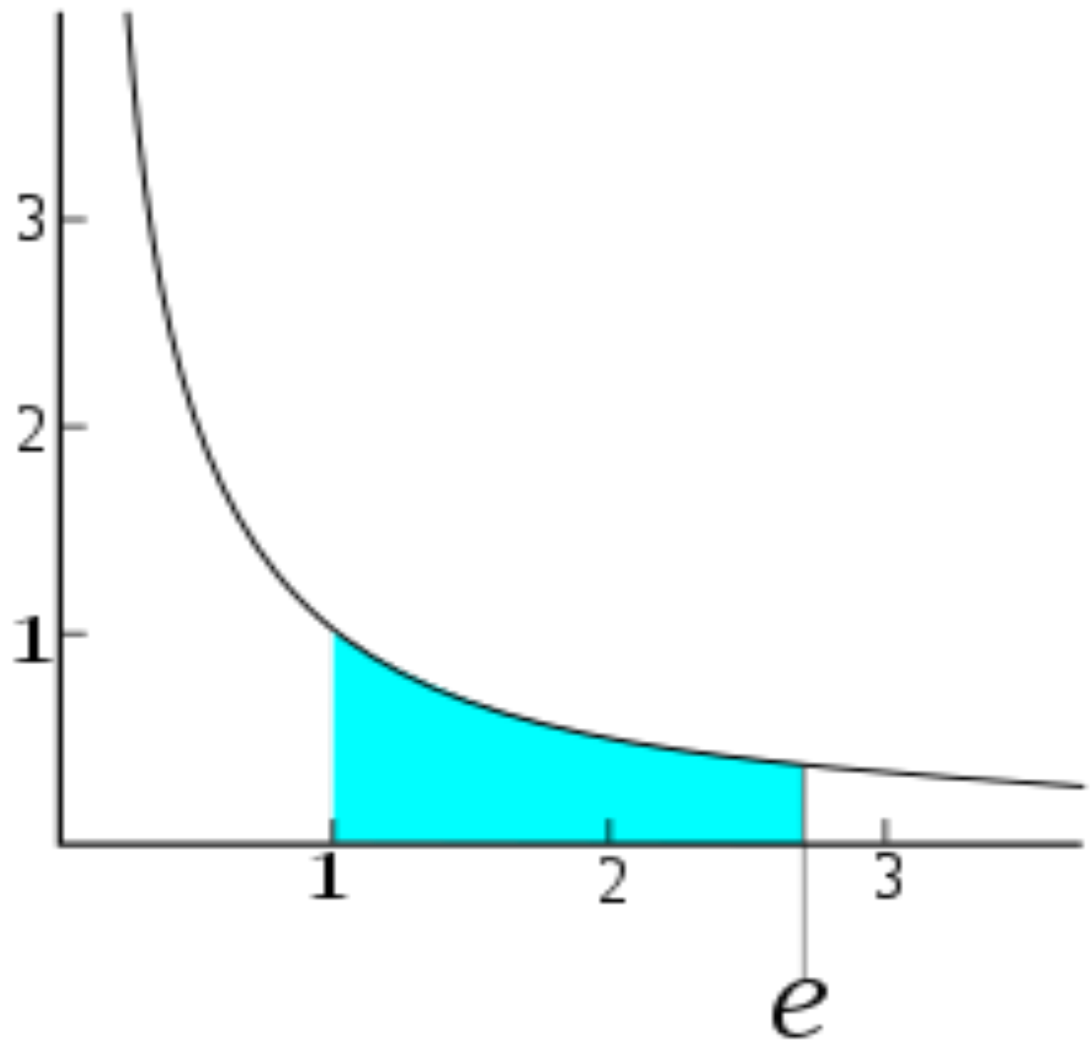
Formula for e

- Natural constant e is "computable" by infinite series
- The formula for infinite series of e:

$$e = e^1 = \sum_{n=0}^{n=\infty} 1^n / n!$$

- AKA $e = 1^0/0! + 1^1/1! + 1^2/2! + 1^3/3! + 1^4/4! + 1^5/5! \ldots$
- Or simpler:
- $e = 1 / 0! + 1 / 1! + 1 / 2! + 1 / 3! + 1 / 4! + 1 / 5! \ldots$
- We shall generate e and later e^x iteratively in C++
- e computation is just a special case of e^x, with $x = 1$

2-D Surface Area e for y = 1 / x



Compute e in C++

```
#include . . .
unsigned fact( unsigned limit )   // iteration's limit
{ // fact
    int result = 1;
    for( int i = 1; i <= limit; i++ ) {
        result *= i;                // no overflow check!
    }//end for
    return result;
} //end fact

// start with 1.0, then:
// iterate Euler's formula from 1 to "index"
double e( unsigned index )
{ // e
    double result = 1.0;            // = 1/0!
    for( int i = 1; i <= index; i++ ) {
        result += 1.0 / fact( i );
    } //end for
    return result;
} //end e
```

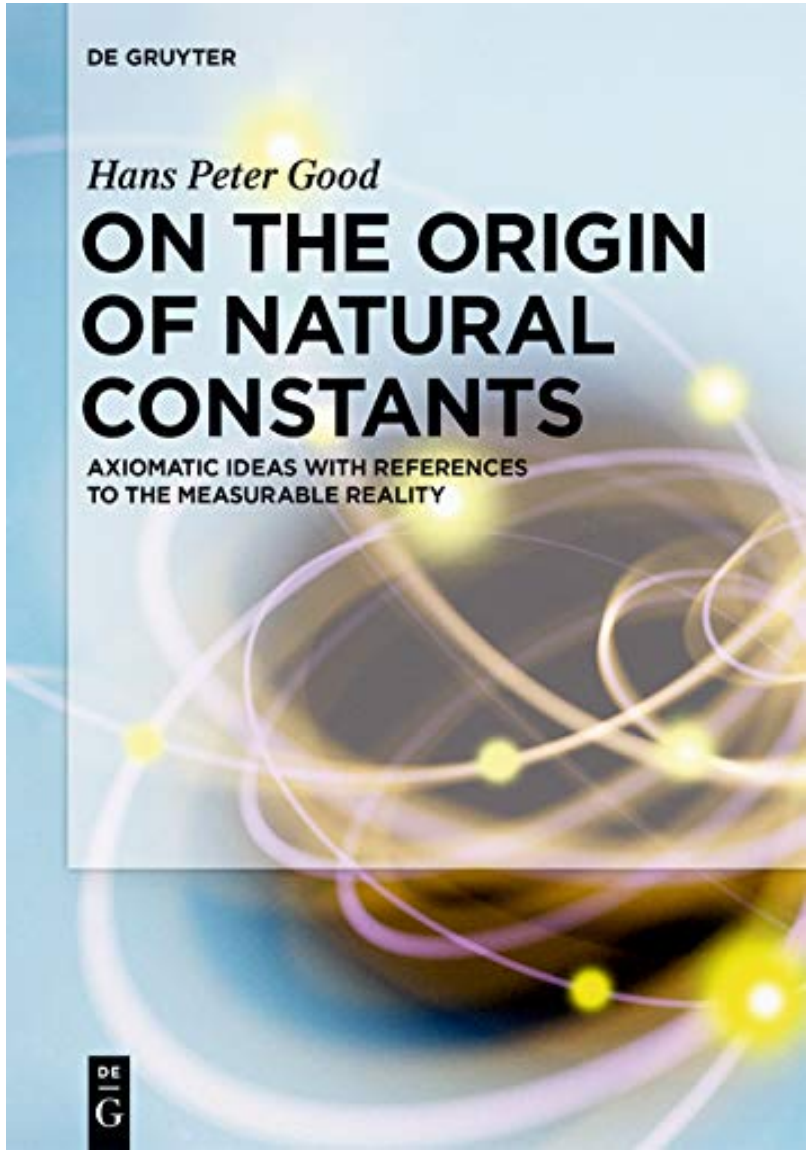
Compute e in C++

```
. . .

int main( void )
{ // main
   // formula for e uses iteration algorithm
   // pass "steps" to specify: "number of iterations"
   // starting at 1 show gradual progress of precision of e
   // arbitrary number 13, magic, yet known to be safe
   for( int steps = 1; steps < 13; steps++ ) {
      printf( " step(%2d ) e = %2.12f\n", steps, e( steps ) );
   } //end for
   // in the end: print library's reference point for 'e'

   printf( "Actual value of e = 2.718281828459...\n" );
   return 0;
} //end main
```

Generate e Output

```
>a.out
      step( 1 ) e = 2.000000000000
      step( 2 ) e = 2.500000000000
      step( 3 ) e = 2.666666666667
      step( 4 ) e = 2.708333333333
      step( 5 ) e = 2.716666666667
      step( 6 ) e = 2.718055555556
      step( 7 ) e = 2.718253968254
      step( 8 ) e = 2.718278769841
      step( 9 ) e = 2.718281525573
      step(10 ) e = 2.718281801146
      step(11 ) e = 2.718281826198
      step(12 ) e = 2.718281828286
Actual value of e = 2.718281828459...
>
```

Other Natural Constants



Formula for e^x

- Infinite series for e^x is known to be:

$$e^x = \sum_{n=0}^{n=\infty} x^n / n!$$

AKA $e^x = x^0 / 0! + x^1 / 1! + x^2 / 2! + x^3 / 3! + x^4 / 4! + x^5 / 5! \ldots$

- Now compute e^x in C++
- Similar to case before, where exponent x of e was consistently 1
- Was intended to look simplistic ☺ earlier: e^1
- Now exponent is x, we shall compute: e^x

e^x Function in C++

```
unsigned fact( unsigned limit )
{ // fact
    int result = 1;
    for( int i = 1; i <= limit; i++ ) {
        result *= i;              // no overflow check!
    }//end for
    return result;
} //end fact

double ex( int index, double expo )
{ // ex -> e power x
    double result = 1.0;          // start: index 1 and 1.0
    double numerator = expo;
    for( int i = 1; i <= index; i++ ) {
        result += numerator / fact( i );
        numerator *= expo;
    } //end for
    return result;
} //end ex
```

e^x Function in C++

```
void iterate_ex( double expo )
{ // iterate_ex
   for( int i = 1; i < 13; i++ ) { // 13 picked arbitrarily
     printf("step(%2d) e^%f = %2.12f\n",i, expo, ex(i,expo ) );
   } //end for
   printf("Actual value of e = 2.718281828459\n" );
   printf("And value of e**2 = 7.389056099\n" );
} //end iterate_ex

int main()
{ // main
  for( float expo = 1.0; expo < 2.1; expo += 0.250 ) {
       printf( "exponent = %f\n", expo );
       iterate_ex( expo );
       printf( "\n" );
   }//end for
   return 0;
} //end main
```

e^x Function in C++

```
exponent = 1.250000
      step( 1 ) e^1.250000 = 2.250000000000
      step( 2 ) e^1.250000 = 3.031250000000
      step( 3 ) e^1.250000 = 3.356770833333
      step( 4 ) e^1.250000 = 3.458496093750
      step( 5 ) e^1.250000 = 3.483927408854
      step( 6 ) e^1.250000 = 3.489225599501
      step( 7 ) e^1.250000 = 3.490171704973
      step( 8 ) e^1.250000 = 3.490319533954
      step( 9 ) e^1.250000 = 3.490340065756
      step(10 ) e^1.250000 = 3.490342632232
      step(11 ) e^1.250000 = 3.490342923877
      step(12 ) e^1.250000 = 3.490342954256
         Actual value of e = 2.718281828459
         And value of e**2 = 7.389056099
```

e^x Function in C++

```
exponent = 1.500000
      step( 1 ) e^1.500000 = 2.500000000000
      step( 2 ) e^1.500000 = 3.625000000000
      step( 3 ) e^1.500000 = 4.187500000000
      step( 4 ) e^1.500000 = 4.398437500000
      step( 5 ) e^1.500000 = 4.461718750000
      step( 6 ) e^1.500000 = 4.477539062500
      step( 7 ) e^1.500000 = 4.480929129464
      step( 8 ) e^1.500000 = 4.481564767020
      step( 9 ) e^1.500000 = 4.481670706613
      step(10 ) e^1.500000 = 4.481686597552
      step(11 ) e^1.500000 = 4.481688764498
      step(12 ) e^1.500000 = 4.481689035366
         Actual value of e = 2.718281828459
         And value of e**2 = 7.389056099
```

e^x Function in C++

```
exponent = 2.000000
      step( 1 ) e^2.000000 = 3.000000000000
      step( 2 ) e^2.000000 = 5.000000000000
      step( 3 ) e^2.000000 = 6.333333333333
      step( 4 ) e^2.000000 = 7.000000000000
      step( 5 ) e^2.000000 = 7.266666666667
      step( 6 ) e^2.000000 = 7.355555555556
      step( 7 ) e^2.000000 = 7.380952380952
      step( 8 ) e^2.000000 = 7.387301587302
      step( 9 ) e^2.000000 = 7.388712522046
      step(10 ) e^2.000000 = 7.388994708995
      step(11 ) e^2.000000 = 7.389046015713
      step(12 ) e^2.000000 = 7.389054566832
         Actual value of e = 2.718281828459
         And value of e**2 = 7.389056099
```

Differentiation of $f(e^x)$

- $f1(x) = e^x$
- $f1(x)' = e^x$ = e^x -- ^ different notation

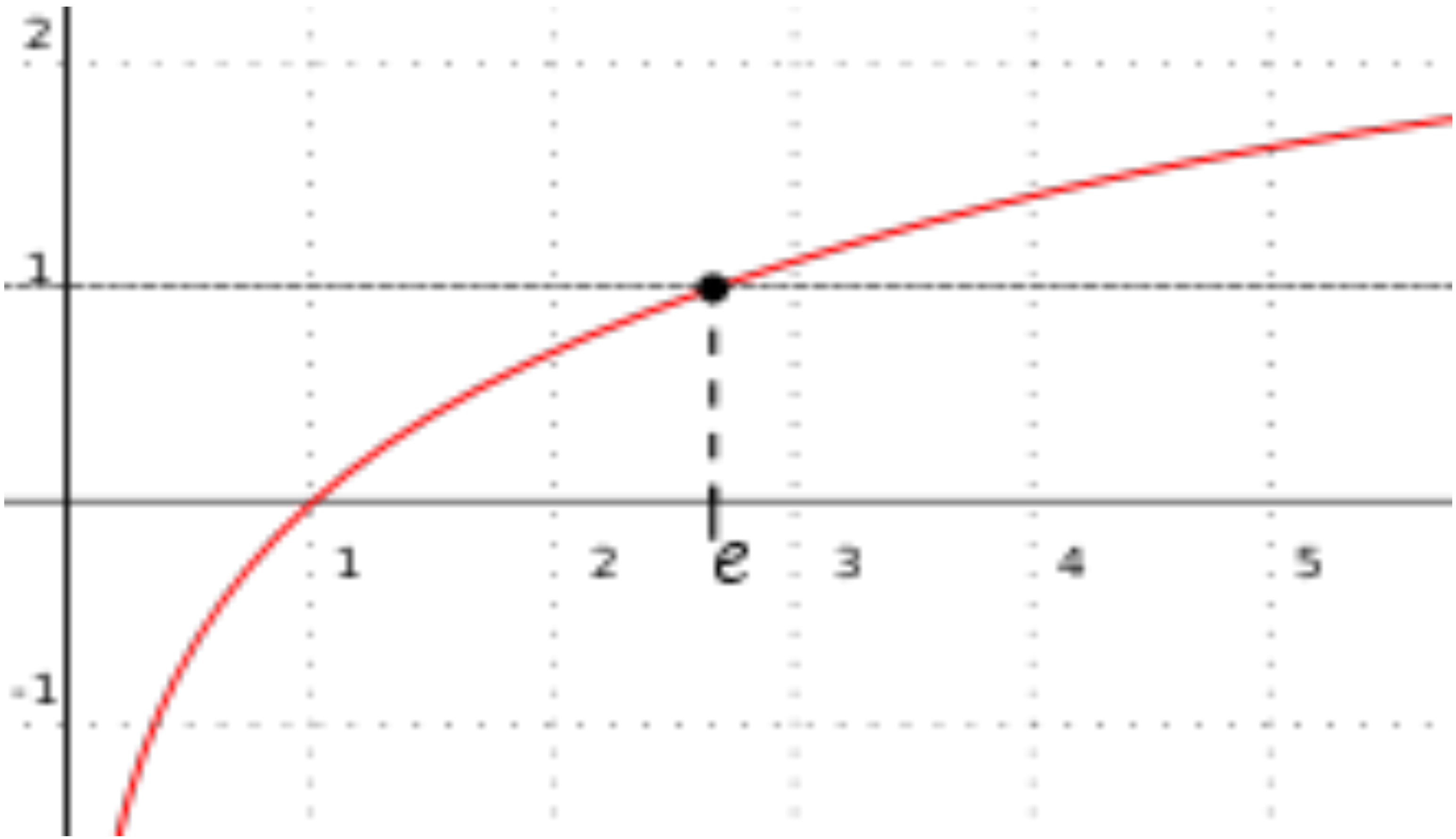
- $f2(x) = 5x\, e^{2x}$ -- use Product Rule
- $f2(x)' = 5e^{2x} + 5x\, 2e^{2x}$
- $f2(x)' = 5e^{2x} + 10x\, e^{2x} = 5e^{2x}(1 + 2x)$

- $f3(x) = e^{-x/2} \sin(ax)$ -- use Product Rule
- $f3(x)' = (-1/2)\, e^{-x/2} \sin(ax) + a\, e^{-x/2} \cos(ax)$

Integrals of f(e^x)

- $\int e^x \, dx \quad = \quad e^x$ plus some constant – 1
- $\int e^{cx} \, dx \quad = \quad e^{cx}/c$ – 2
- $\int x \, e^{cx} \, dx \quad = \quad x/c \; e^{cx} - 1/c^2 \; e^{cx}$ – 3
- $\int x \, e^{cx} \, dx \quad = \quad (x/c - 1/c^2) \, e^{cx}$ – 4
- $\int x^2 \, e^{cx} \, dx \quad = \quad (x^2/c - 2x/c^2 + 2/c^3) \, e^{cx}$ – 5
- $\int x^3 \, e^x \, dx \quad = \quad (x^3 - 3x^2 + 6x - 6) \, e^x$ – 6
- $\int x^n \, e^{cx} \, dx \quad = \quad x^n \, e^{cx} / c - n/c \;\; x^{n-1} \, e^{cx} \, dx$ – 7

Natural Log ln(x) Inverse of e^x



Value of natural log for argument *e*, i.e. ln(*e*), equals 1

Summary

- Programmers sometimes use approximations in mathematical computations
- Practiced here for the natural constant e
- Cost: iterative steps, and non-perfect result, mathematically speaking
- In practice: Accuracy can be parameterized by specifying the number of iterative steps
- To any desired precision, as needed
- Hence imprecision of effectively skipping an infinite number of summands in a series poses no true problem
- Not a real accuracy issue in Discrete Mathematics and thus in Discrete Structures

References

1. Leonhard Euler: https://en.wikipedia.org/wiki/Leonhard_Euler
2. Euler family tree: http://eulerarchive.maa.org/historica/family-tree.html
3. Constant e: https://en.wikipedia.org/wiki/E_(mathematical_constant)