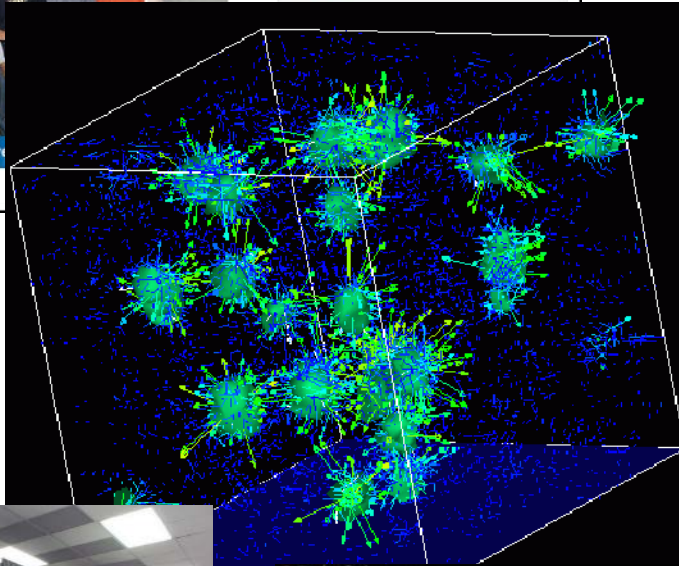




# Survey of Scientific Computing (SciComp 301)

Dave Biersach  
Brookhaven National  
Laboratory  
[dbiersach@bnl.gov](mailto:dbiersach@bnl.gov)



```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9
10 namespace SimpleEvents
11 {
12     public partial class Form1 : Form
13     {
14         Person person = new Person();
15
16         public Form1()
17         {
18             InitializeComponent();
19             person.FirstName = "Christian";
20             person.LastName = "Pano";
21         }
22
23         private void button1_Click(object sender, EventArgs e)
24         {
25             person.MainColor = textBox1.Text;
26         }
27     }
28 }
```

**Exam 2**  
Total of 100 points

5 pts

## 1. Solve a given 4x4 system

In the **q01** folder, edit the C++ console application to calculate the four unknowns for this system of equations:

$$x_1 + 2x_2 + x_3 - x_4 = 5$$

$$3x_1 + 2x_2 + 4x_3 + 4x_4 = 16$$

$$4x_1 + 4x_2 + 3x_3 + 4x_4 = 22$$

$$2x_1 + x_3 + 5x_4 = 15$$

15 pts

## 2. Create and solve a random 10x10 system

In the **q02** folder, edit the C++ console application to **generate** and solve a system of 10 random linear equations having 10 unknowns using Cramer's Rule

The coefficients and values should be taken from a uniform **real** distribution between  $[-10,10)$  with initial seed of 2016

Display all numbers with **four** digits of precision to the right of the decimal point

Copy the two helper functions shown on the right to initialize the **coeffMatrix** and the **valueVector** variables in **main()**

```
matrix1D CreateRandomVector(size_t rows)
{
    matrix1D A(rows);
    for (size_t row{}; row < rows; row++)
        A.at(row) = distribution(generator);
    return A;
}

matrix2D CreateRandomMatrix(size_t rows, size_t cols)
{
    matrix2D A;
    A.resize(rows, matrix1D(cols));
    for (size_t row{}; row < rows; row++)
        for (size_t col{}; col < cols; col++)
            A.at(row).at(col) = distribution(generator);
    return A;
}
```

20 pts

### 3. Calculate Riemann's $\pi(x)$

In the **q03** folder, edit the C++ console application to calculate and display the **Prime Counting Function**  $\pi(x)$  for each successive power of 10 from 1 to 6 inclusive

$x$	$\pi(x)$
10	4
$10^2$	25
$10^3$	168
$10^4$	1,229
$10^5$	9,592
$10^6$	78,498

```
File Edit View Terminal
10^1 = 4
10^2 = 25
10^3 = 168
10^4 = 1,229
10^5 = 9,592
10^6 = 78,498
Run time: 9.27152s
```

Special bonus points if your code runs faster than mine!

```
main.cpp
1  #include "stdafx.h"
2
3  using namespace std;
4  using namespace chrono;
5
6  vector<int> primes{ 2,3 };
7
8  int PrimeCountingFunction(int x)
9  {
10     // Add your code here
11 }
12
13 int main()
14 {
15     cout.imbue(locale(""));
16     cout << right;
17
18     boost::timer timer;
19
20     for (int i{ 1 }; i <= 6; i++)
21     {
22         cout << "10^" << to_string(i) << " = " << setw(7);
23         cout << PrimeCountingFunction((int)pow(10, i)) << endl;
24     }
25
26     cout << endl << "Run time: "
27         << timer.elapsed() << "s" << endl;
28
29     return 0;
30 }
31
```

You must write this function

10 pts

## 4. Calculate Gamma from Eta

$$\eta(s) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n^s}$$

$$\eta(s) = \frac{1}{\Gamma(s)} \int_0^{\infty} \frac{x^{s-1}}{e^x + 1} dx$$

```
main.cpp
1  #include "stdafx.h"
2
3  using namespace std;
4
5  double Eta(double s)
6  {
7      double eta = 1;
8      for (int n{ 2 }; n <= 1e5; n++)
9      {
10         double term = 1. / pow(n, s);
11         eta += (n % 2) == 0 ? -term : term;
12     }
13     return eta;
14 }
```

```
16 inline double f(double x, double s)
17 {
18     // Add your code here
19 }
20
21 double Simpsons(double s)
22 {
23     double a{ 0 };
24     double b{ 1e3 };
25     int intervals = 1e5;
26
27     double dx{ (b - a) / intervals };
28     double sum{ f(a,s) + f(b,s) };
29     a += dx;
30     for (int i{ 1 }; i < intervals; ++i, a += dx)
31         sum += f(a, s)*(2 * (i % 2 + 1));
32     return (dx / 3)*sum;
33 }
```

In the **q04** folder, edit the C++ console application to display **5!** using only  $\eta(s)$  and the value of the integral

15 pts

## 5. Expand a Standard CF

In the **q05** folder, edit the C++ console application to display the first **seven** terms of the standard continued fraction encoding for each value of  $x_n$ :

$$x = \frac{1 + \sqrt{4n^2 - 4n + 5}}{2}$$

$$n \in \mathbb{Z}^+, n < 10$$

```
main.cpp
1  #include "stdafx.h"
2
3  using namespace std;
4
5  vector<int> EncodeCF(double x)
6  {
7      vector<int> terms;
8      while(terms.size() < 20)
9      {
10         terms.push_back(floor(x));
11         x = x - floor(x);
12         if (x < 1e-9) break;
13         x = 1/x;
14     }
15     return terms;
16 }
17
18 void DisplayCF(vector<int> terms)
19 {
20     cout << "{";
21     auto itr = terms.begin();
22     while (true)
23     {
24         cout << *itr;
25         if (++itr == terms.end()) break;
26         cout << ", ";
27     }
28     cout << "}\n";
29     return;
30 }
31
32 int main()
33 {
34     for (int n{ 1 }; n < 1; n++)
35     {
36         // Insert your code here
37     }
38     return 0;
39 }
40
```

10 pts

## 6. Integrate the Standard Normal

In the **q06** folder, edit the C++ console application to use Simpson's Rule to estimate the area under the **standard normal** curve from **-1** to **1**

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Is this area the same as the probability of a normally distributed random variable falling within the **first** standard deviation away from the mean?

```
main.cpp
1  #include "stdafx.h"
2
3  using namespace std;
4
5  inline double f(double x)
6  {
7      return 0;
8  }
9
10 double Simpsons(double a, double b)
11 {
12     int intervals = 1e5;
13     double dx{ (b - a) / intervals };
14     double sum{ f(a) + f(b) };
15     a += dx;
16     for (int i{ 1 }; i < intervals; ++i, a += dx)
17         sum += f(a)*(2 * (i % 2 + 1));
18     return (dx / 3)*sum;
19 }
20
21 int main()
22 {
23     cout << Simpsons(0, 0) << endl;
24
25     return 0;
26 }
27
```

5 pts

## 7. Decrypt Ciphertext

In the **q07** folder, edit the C++ console application to decrypt the file **ciphertext2.txt**

To get started, execute the **CERN ROOT** code from **Session 14 Lab 2**



10 pts

## 8. Finding an Open Reading Frame (ORF)

[https://en.wikipedia.org/wiki/Open\\_reading\\_frame](https://en.wikipedia.org/wiki/Open_reading_frame)

<http://vlab.amrita.edu/?sub=3&brch=273&sim=1432&cnt=1>

**DNA** (Deoxyribonucleic acid) is the genetic material that contains all the genetic information in a living organisms. The information is stored as genetic codes using adenine (A), guanine (G), cytosine(C) and thymine (T). During the transcription process, DNA is copied to mRNA. Each of these base pairs will bond with a sugar and phosphate molecule to form a nucleotide. **Three nucleotides** that code for a particular amino acid during translation is called as a **codon**. An Open Reading Frame (ORF) goes from the **start codon** to a **stop codon** and encodes a specific protein. By analyzing the ORF we can predict the possible amino acids that might be produced during translation.

Given a sequence, create 6 different reading frames labeled +1, +2, +3, -1, -2 and -3:

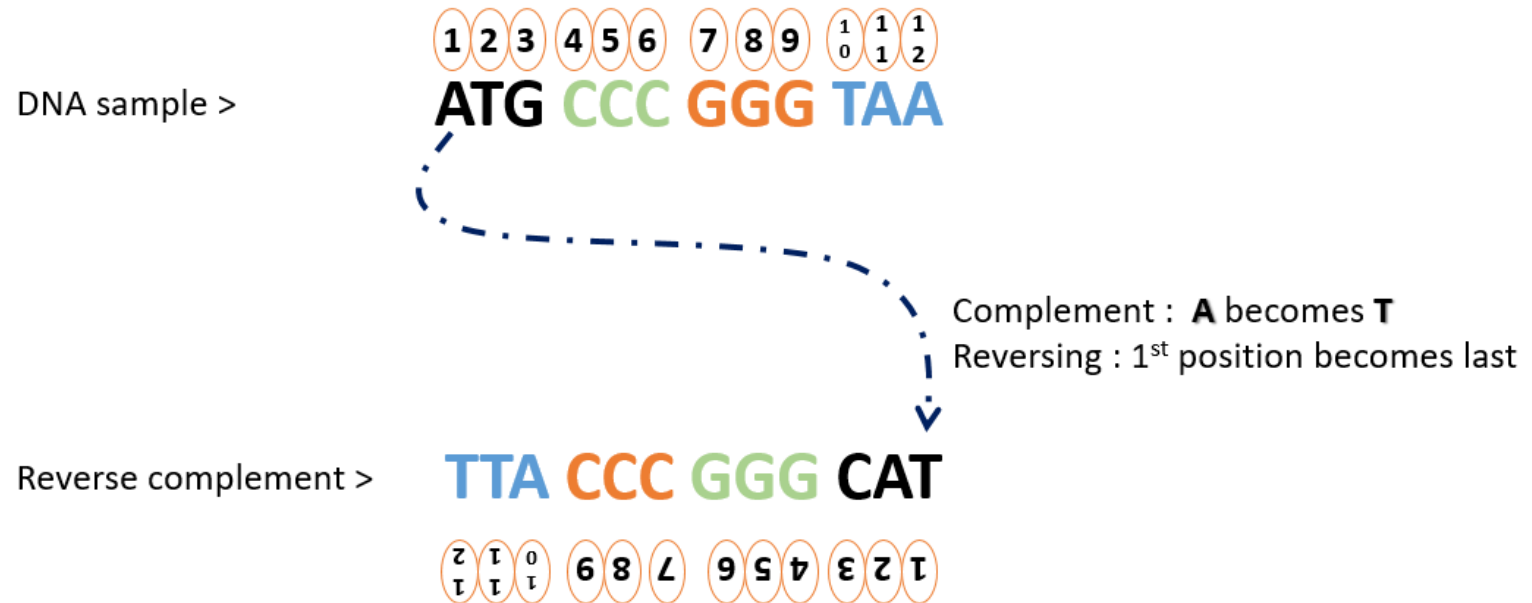
- The first reading frame is obtained by considering the sequence in words of 3
- The second reading frame is formed after skipping the first nucleotide and then grouping the remaining sequence into words of 3 nucleotides
- The third reading frame is formed after skipping the first 2 nucleotides and then grouping the remaining sequence into words of 3 nucleotides
- The other negative 3 reading frames can be found only after finding the **reverse complement of the sequence**
- The same process as for the +1, +2 and +3 strands is repeated for the -1, -2 and -3 strands, but now using the *reverse complement* sequence

To identify each open reading frame (ORF):

- Find the start codon and stop codons in each reading frame
- Display the sequence stretch beginning with a start codon and ending in a stop codon

10 pts

## 8. Finding an Open Reading Frame (ORF)



## 8. Finding an Open Reading Frame (ORF)

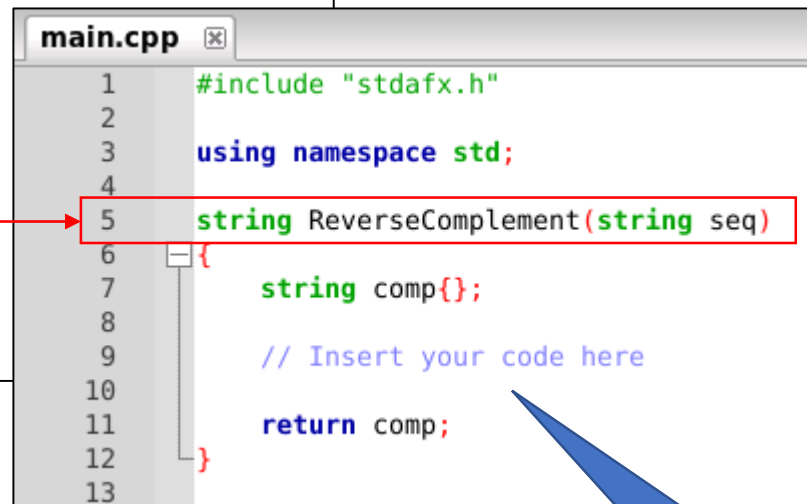
<http://vlab.amrita.edu/?sub=3&brch=273&sim=1432&cnt=1>

```
int main()
{
    string seq{ "CGCTACGTCTTACGCTGGAGCTCTCATGGATCGGTTGGTAGGGCTCGATCACATCGCTAGCCAT" };
    cout << seq << endl << endl;

    string seqRevComp = ReverseComplement(seq);

    for (auto offset :
        {
            1,2,3,-1,-2,-3
        })
    {
        if (offset > 0)
            FindORF(seq, offset);
        else
            FindORF(seqRevComp, offset);
    }

    return 0;
}
```



```
main.cpp
1  #include "stdafx.h"
2
3  using namespace std;
4
5  string ReverseComplement(string seq)
6  {
7      string comp{};
8
9      // Insert your code here
10
11     return comp;
12 }
13
```

In the **q08** folder, edit the existing C++ console application to generate the **reverse complement** of a given DNA sequence

You must write this function

## 8. Finding an Open Reading Frame (ORF)

```
void FindORF(const string& seq, int offset)
{
    vector<string> frame = CreateReadingFrame(seq, offset);

    size_t posStart{};
    if (!FindCodon(frame, { "ATG" }, 0, posStart))
        return;

    size_t posStop{};
    if (!FindCodon(frame, { "TAA", "TAG", "TGA" }, posStart + 1, posStop))
        return;

    if (posStart + 1 == posStop)
        return;

    cout << "Frame ";
    if (offset > 0)
        cout << "+";
    cout << offset << ": ";
    for (size_t i{ posStart }; i <= posStop; i++)
        cout << frame.at(i) << " ";
    cout << endl;
}
```

```
vector<string> CreateReadingFrame(const string& seq, int offset)
{
    size_t i = abs(offset) - 1;
    vector<string> frame{};
    while (i < seq.length() - 2)
    {
        frame.push_back(seq.substr(i, 3));
        i += 3;
    }
    return frame;
}
```

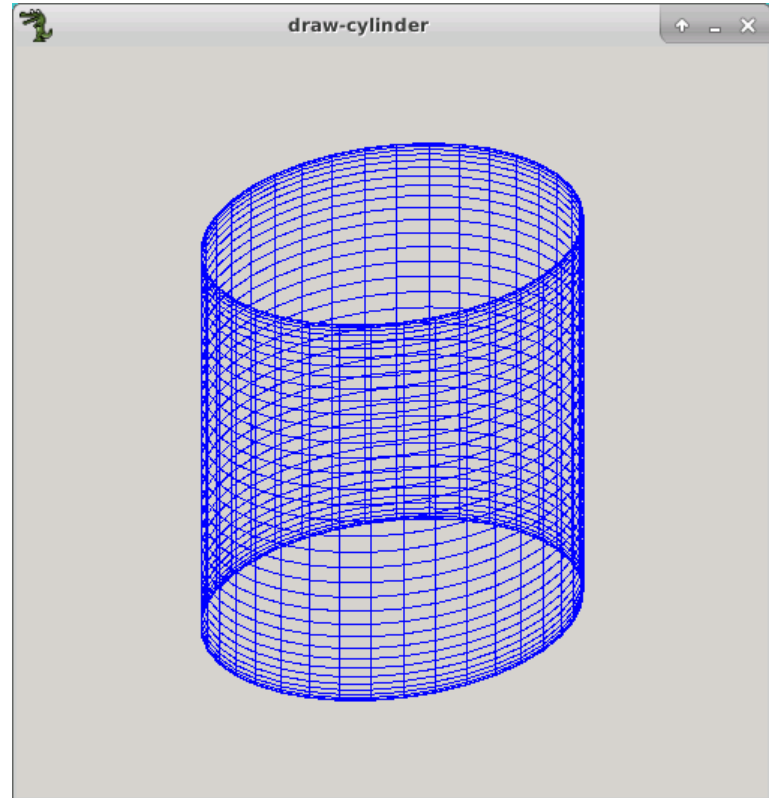
```
bool FindCodon(const vector<string>& frame,
               const vector<string>& codons, size_t start, size_t& pos)
{
    pos = start;
    bool found = false;
    while (!found && pos < frame.size())
    {
        for (const string& codon : codons)
            if (frame.at(pos) == codon)
                found = true;
        if (!found)
            pos++;
    }
    return found;
}
```

5 pts

## 9. Draw a 3D Cylinder

In the **q09** folder, edit the C++ Allegro application to draw an open-ended 3D cylinder with **radius 100** and **length 200**

Make it a wireframe drawing.  
No back face culling or facet shading is required.



5 pts

## 10. Sample a Sinusoid

In the **q10** folder, program **CERN ROOT** to draw a sine wave with exactly **7 crests** per every **13 units**, over the domain  $0 \leq x \leq 26$

