ENGR-UH 1000

Computer Programming for Engineers

Assignment 2: Mechanical Engineering: Case Study – Flight Simulator Wind Speed

Mohamad Eid

Fall 2021

Pavly Halim | poh2005

Step 1: Problem Identification and Statement

Using the polynomials assume that the wind speed for a particular region can be modeled by using an average value and a gust value that is added to the average. Representing the simulation duration, and step size. Additionally, storm probability, minimum and maximum storm amplitudes values and minimum and maximum duration of storm. On the other side, As representing the microburst probability, minimum and maximum values of microburst amplitude and minimum and maximum values of microburst duration. Also, must incorporate the simulation of storms and microbursts during the storm.

Step 2: Gathering Information

Here is an example plot of the generated wind speed with no storms or microbursts, with the following settings: average wind speed: 10 miles per hour, gust value: 2 miles per hour, simulation duration: 1 hour, time increment: 10 seconds. The data is shown in Figure 1.

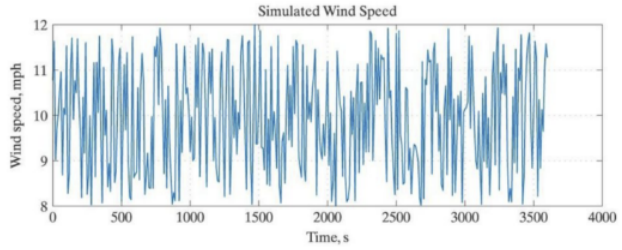


Figure 1: Simulated wind speeds.

An example of the wind speed generated data including three storms is shown in figure 2 (based on the data generated in Figure 1). The example is generated using a 0.5% possibility of encountering a storm, an average storm amplitude of 10 miles per hour, and a storm duration of 5 minutes.

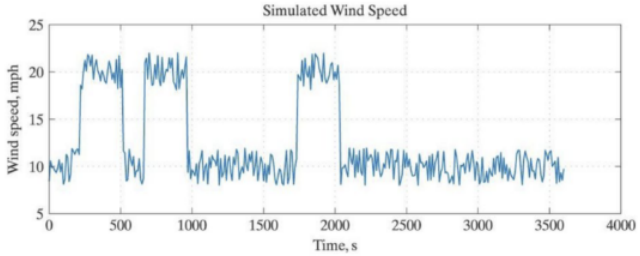


Figure 2: simulated wind speeds with three storms.

An example of one microburst simulation added to the simulation of two storms is shown Figure 3.

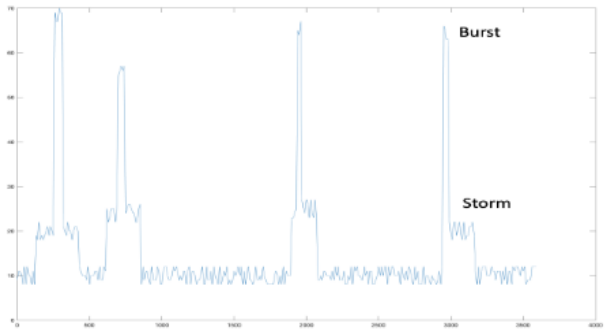


Figure 3: Simulated wind speeds with four storms/microbursts.

How to generate random numbers and how storms and microbursts are simulated.

Using generate random number function in which we can return a pseudo random integral number between the range we determined as in our case. We have to generate 5 random Values: The wind speed is a random number between the (average speed + gust value) and the (average speed - gust value). The second two values are the storm magnitude (a random number between the minimum and maximum storm amplitude values) to the simulated wind speed for a duration T (a random number between the minimum and maximum storm duration). The last two are the burst amplitude (a random number between the minimum and maximum microburst amplitude values), and the duration of the burst (a random number between the minimum and maximum storm duration values). the storms and microbursts are simulated in final wind simulation by combining the wind speed, storm, and burst data

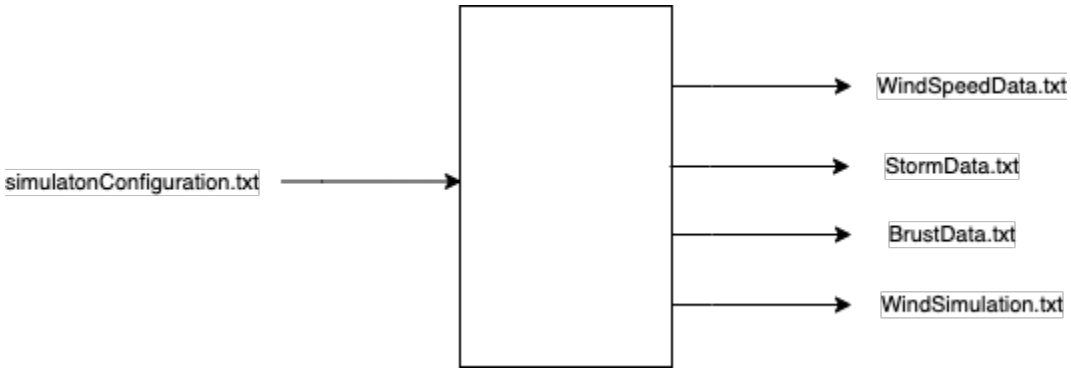
Input/Output Description

The following I/O diagram shows the user doesn't need to input data while executing the code by reading the data from file “simulationConfiguration.txt” that the first line includes the average wind speed, gust value, simulation duration, and step size. The second line from the file contains storm probability, minimum and maximum storm amplitudes values, minimum and maximum values of storm duration. The third line contains the Microburst probability, minimum and maximum values of microburst amplitude, and minimum and maximum values of microburst duration. Also, must incorporate the simulation of wind speed, storms, and microbursts during the storm.

simulationConfiguration.txt ×

```
1   30 5 15 1
2   10 10 30 2 6
3   25 15 40 1 5
4
```

The output is “WindSpeedData.txt” that contains the simulated wind speeds. The storm simulation data must be stored in the file “StormData.txt”. The microburst data must be stored in the file “BurstData.txt”. The final output file wind simulation by combining the wind speed, storm, and burst data. Save the result in a data file named “WindSimulation.txt”. A binary value to each line signifying whether a storm is active at the same time. We can use the built-in data type Double, Float, and integers for input and output objects.



Step 3: Test Cases

Test case 1: Storm occurrence:

Probability of occurrence of storm in which is used to compare the random generated value between the range of the minimum and maximum; to check if the value is greater than or equal to the randomized value. We can notice the difference by changing the storm occurrence probability by increasing the probability by increasing the duration that we can observe more storm occurrences. In contrast, decreasing that will lead to less occurrence of storms.

Test case 2: Microburst occurrence:

Microburst are simulated using three parameters: the possibility of encountering a microburst at each time step in a storm (Pburs). The Probability of occurrence of microburst in which is used to compare the random generated value between the range of the minimum and maximum; to check if the value is greater than or equal to the randomized value. We can notice the difference by changing the microburst occurrence probability by increasing the probability by increasing the duration that we can observe more storm occurrences. In contrast, decreasing that will lead to less occurrence of microburst.

Test case 3: Storm amplitude:

The storm magnitude (a random number between the minimum and maximum storm amplitude values), then start comparing the value. We can notice the difference by changing the maximum and minimum values of storm amplitude occurrence probability by increasing the probability by increasing the storm amplitude that we can observe more storm occurrences. In contrast, decreasing that will lead to less amplitude of storms.

Test case 4: Storm duration:

The simulated wind speed for a duration T (a random number between the minimum and maximum storm duration). then start comparing the value. We can notice the difference by changing the maximum and minimum values of Storm duration occurrence probability by increasing the probability by increasing the Storm duration that we can observe more storm occurrences. In contrast, decreasing that will lead to less duration of storm.

Test case 5: Microburst amplitude:

The burst amplitude (a random number between the minimum and maximum microburst amplitude values), then start comparing the value. We can notice the difference by changing the maximum and minimum values of burst amplitude occurrence probability by increasing the probability by increasing the burst amplitude that we can observe more burst occurrences. In contrast, decreasing that will lead to less amplitude of burst.

Test case 6: Microburst duration:

The duration of the burst (a random number between the minimum and maximum microburst duration values). then start comparing the value. We can notice the difference by changing the maximum and minimum values of burst amplitude occurrence probability by increasing the probability by increasing the burst duration that we can observe more burst occurrences. In contrast, decreasing that will lead to less duration of burst.

Test case 7: Exit message the program

The program should print the message “Done Reading Data successfully” and exit the program

Step 3: Algorithm Development

Algorithm design:

```
Declare configloader as a void

function Assigned in array1, array2, and array3 as integers

create input file and call it windspeed
open "windspeeddata.txt"
by windspeed

if openning input windspeed failed

print the error: "opening file failed", newline

exit

declare array4 as integer with assigned value of array1 element
2

print into windspeed space "duration"
space "windspeed", newline

update the random numbers by unsigned integer within time value
null

repeat as integer x is equal to 0 and it will end when x smaller
or equal to the element 3 in array1 and, increment it by one
point each time until end when more or equal element 4 in array1

assign the random value into randomwind as integer equal to
probabilty assigned values in arrays(((array1[element 0] +
array1[element 1]) - (array1[element 0] - array1[element 1])) +
1) + (array1[element 0] - array1[element 1]);

print into windspeed space "x"
space "randomwind", newline

assign randomwind equal to array4[element x]

declare storm into array4, array2, array1, array3

close windspeed
```

```
Declare storm as a void

function Assigned in array4, array7, array3, and array1 as
integers

update the random numbers by unsigned integer within time value
null

create input file and call it stormdata
open "StormData.txt"
by stormdata

if openning input stormdata failed

print the error: "opening file failed", newline

exit

declare Prand variable as a float equal to rand value to 100

Prand equal to Prand divide by 100

declare probability as a float equal to array7 first element

declare prob as a float equal to probability divide by 100
declare variable x as an integer equal to 0
declare array6 as integer assigned to third element array1

print into stormdata space "duration"
space "windspeed", newline

repeat as integer y and it will end when y smaller or equal to
the element 3 in array1 and, increment it by one point each time
until end when more or equal element 4 in array1

assign the random value into randomstorm as float
equal to assigned values in arrays(array7[1] + random number
divide by((array7[2] - array7[1] + 1))

    assign variable T as float equal to array7[3] + random value
divide by((array7[4] - array7[3] + 1) declare newtime as float
assigned to value equal to T plus X

      if condition of the Prand value smaller or equal to Prob
```

```
      repeat as integer x and x is smaller or equal to newtime
and x smaller or equal array1 element 3 and increment it by one
point each time until end when more or equal element 4 in array1

      variable newspeed equal to randomstorm plus array4[x]

      print into stormdata space x space newspeed newline

      assign array6[x] into newspeed

      print stormdata space x space array4[x] newline assign
array6[x] into array4[x] variable x greater or equal to
array1[element 4]

      Prand variable is equal to rand value to 100

      Prand equal to Prand divide by 100

      declare burst into array4, array6, array3, array1

      close file by stormdata

      Declare burst as a void
      function Assigned in array4, array6, and array7, array1 as
integers

      update the random numbers by unsigned integer within time
value null

      declare isStorm as boolean

      create input file and call it burstdata open
"BurstData.txt"
      by burstdata

      if openning input burstdata failed

      print the error: "opening file failed", newline

      exit

      declare newspeed variable as a float and Prand equal to
rand value to 100

      Prand equal to Prand divide by 100
```

```
      declare probability as a float equal to array7 first
element

      declare prob as a float equal to probability divide by 100
declare variable x as an integer equal to 0 declare array8 as
integer assigned to third element array1

      print into burstdata space "duration"
      space "windspeed", newline

      repeat as integer y and it will end when y smaller or
equal to the element 3 in array1 and, increment it by one point
each time until end when more or equal element 4 in array1

      assign the random value into randomburst as float equal to
assigned values in arrays(array7[1] + random number divide
by((array7[2] - array7[1] + 1))

        assign variable T as float equal to array7[3] + random
value divide by((array7[4] - array7[3] + 1) declare bursttime as
float assigned to value equal to T plus X

          if condition of the array6[x] greater than array4[x]

          isStorm equal true

          if isStorm equal false

          if Prand smaller than or equal prob

          repeat as integer x and isStorm equal equal true and x
is smaller or equal to bursttime and increment it by one point
each time until end when more or equal element 4 in array1

          variable newspeed equal to randomstorm plus array4[x]
newspeed variable equal to randomburst plus array6[x]

          print into burstdata space "x"
          space "newspeed", newline

          assigned array8[x] equal to newspeed value

          if the condition not applied

          print burstdata space "x"
```

```
          space array6[x] newline assign array8[x] into
array6[x] variable x greater or equal to array1[element 4]

          Prand variable is equal to rand value to 100 Prand
equal to Prand divide by 100

          assign windsim into array4, array6, array8, array1

          close file using windsim

          Declare indsim as a void
          function Assigned in array4, array6, and array8,
array1 as integers

          update the random numbers by unsigned integer within
time value null

          create input file and call it windsimulation

          open "WindSimulation.txt"
          by windsimulation

          if openning input burstdata failed

          print the error: "opening file failed", newlin exit

          print into windsimulation space "duration"
          space "windspeed"
          space "stormspeed", space "burstspeed"
          newline

          repeat as integer x equal to 0 and it will end when x
smaller or equal to the element 3 in array1 and, increment it by
one point each time until end when more or equal element 4 in
array1 windsimulation space "x"
          space array4[x] space array6[x] space array8[x];

          if condition is array6[] greater than array4[]

          print into windsimulation space "1"
          newline

          print out message "Simulation of wind speed for a
flight simulator created successfully"
          newline
```

```
          close the file using windsimulation

          declare main
          function as integer

          create input file and call it conf open
"simulationConfiguration.txt"
          by conf

          if openning input conf failed

          print the error: "opening file failed", newline

          exit

          declare array1 element 3, array2 element 4, array3
element 6 as integer

          repeat as x equal to 0 and it will end when less than
4, increment it by one point each time

          put line from con into firstline x

          repeat as x is 0 and it will end when less than 5,
increment it by one point each time

          put line from con into secondline x

          repeat as x is 0 and it will end when less than 5,
increment it by one point each time

          put line from getfile into thirdline x

          assign array1, array2, array3 into configloader

          close file using conf
```

Step 4: Implementation

```
//imports libs
#include <iostream>
#include <cmath>
#include <iomanip>
#include <fstream>
#include <cstdlib>

using namespace std;

//functions prototype
void configloader (int array1[], int array2[], int array3[]);
void storm (int array4[], int array5[], int array1[], int
array3[]);
void burst (int array4[], int array6[], int array7[], int
array1[]);
void windsim (int array4[], int array6[], int array8[], int
array1[]);

//function configloader
void
configloader (int array1[], int array2[], int array3[])
{
  // create and open windspeeddata.txt to store data in
  ofstream windspeed;
  windspeed.open ("WindSpeedData.txt", ios::out);

  //check file opening
  if (windspeed.fail ())
    {
      cerr << "opening  file failed \n";
      exit (0);
    }
  int array4[array1[2]];

  //store data in file
  windspeed << left << setw (10) << "duration" << left << setw
(10) <<
    "windspeed" << endl;
  srand (static_cast < unsigned int >(time (nullptr)));
  for (int x = 0; x <= array1[2]; x += array1[3])
```

```
    {

      int randomwind =
    rand () % (((array1[0] + array1[1]) - (array1[0] -
array1[1])) + 1) +
    (array1[0] - array1[1]);
      windspeed << left << setw (10) << x << left << setw (10)
<< randomwind
    << endl;
      array4[x] = randomwind;

    }
  storm (array4, array2, array1, array3);

  //close file
  windspeed.close ();
}

//function storm
void
storm (int array4[], int array7[], int array1[], int array3[])
{
  srand (static_cast < unsigned int >(time (nullptr)));

  //create and read stormdata.txt
  ofstream stormdata;
  stormdata.open ("StormData.txt", ios::out);

  //check file opening
  if (stormdata.fail ())
    {
      cerr << "opening  file failed \n";
      exit (0);
    }

  //declaring variables
  float newspeed;
  float Prand = (rand () % 100);
  Prand = Prand / 100;
  float probability = array7[0];
  float prob = probability / 100;
  int x = 0;
  int array6[array1[2]];

  //write data in file if it passed true the conditions
```

```
  stormdata << left << setw (10) << "duration" << left << setw
(10) <<
    "windspeed" << endl;

  //generate random value

  for (int y; y <= array1[2]; y += array1[3])
    {
      float randomstorm = array7[1] + rand () % ((array7[2] -
array7[1] + 1));
      float T = array7[3] + rand () % ((array7[4] - array7[3] +
1));
      float newtime = T + x;

      if (Prand <= prob)
    {
      for (x; x <= newtime && x <= array1[2]; x += array1[3])
        {
          newspeed = randomstorm + array4[x];
          stormdata << left << setw (10) << x << left << setw
(10) <<
        newspeed << endl;
          array6[x] = newspeed;
        }
    }
      else
    {
      stormdata << left << setw (10) << x << left << setw (10)
<<
        array4[x] << endl;
      array6[x] = array4[x];
      x += array1[3];
    }

      Prand = (rand () % 100);
      Prand = Prand / 100;

    }
  burst (array4, array6, array3, array1);

  //close the file
  stormdata.close ();
}

//function burst
void
```

```
burst (int array4[], int array6[], int array7[], int array1[])
{
  srand (static_cast < unsigned int >(time (nullptr)));
  bool isStorm;

  //create filestream and open burstdata.txt
  ofstream burstdata;
  burstdata.open ("BurstData.txt", ios::out);

  //check file opening
  if (burstdata.fail ())
    {
      cerr << "opening  file failed \n";
      exit (0);
    }

  //declaring variables
  float newspeed, Prand = (rand () % 100);
  Prand = Prand / 100;
  float probability = array7[0];
  float prob = probability / 100;
  int array8[array1[2]];
  int x = 0;

  //write data in file if it passed true the conditions
  burstdata << left << setw (10) << "duration" << left << setw
(10) <<
    "windspeed" << endl;

  //generate random value

  for (int y; y <= array1[2]; y += array1[3])

    {
      float randomburst = array7[1] + rand () % ((array7[2] -
array7[1] + 1));
      float T = array7[3] + rand () % ((array7[4] - array7[3] +
1));
      float bursttime = T + x;
      if (array6[x] > array4[x])
    {
      isStorm = true;
    }
      else
    {
      isStorm = false;
```

```
    }
      if (Prand <= prob)
    {

      for (x; isStorm == true && x <= bursttime; x += array1[3])
        {
          newspeed = randomburst + array6[x];
          burstdata << left << setw (10) << x << left << setw
(10) <<
        newspeed << endl;
          array8[x] = newspeed;
        }
    }
      else
    {
      burstdata << left << setw (10) << x << left << setw (10)
<<
        array6[x] << endl;
      array8[x] = array6[x];
      x += array1[3];
    }

      Prand = (rand () % 100);
      Prand = Prand / 100;

    }
  windsim (array4, array6, array8, array1);

  //close the file
  burstdata.close ();
}

  //function windsim
void
windsim (int array4[], int array6[], int array8[], int array1[])
{
  //create and open WindSimulation.txt
  ofstream windsimulation;
  windsimulation.open ("WindSimulation.txt", ios::out);

  //collecting values from the files and store data in file if
it passed true
  windsimulation << left << setw (10) << "duration" << left <<
setw (10) <<
    "windspeed" << left << setw (12) << "stormspeed" << left <<
setw (10) <<
```

```
    "burstspeed" << endl;
  for (int x = 0; x <= array1[2]; x += array1[3])
    {
      windsimulation << left << setw (10) << x << left << setw
(10) <<
    array4[x] << left << setw (12) << array6[x] << left << setw
(10)
    << array8[x];
      if (array6[x] > array4[x])
    {

      windsimulation << left << setw (10) << "1" << endl;

    }
      else
    {
      windsimulation << endl;
    }
    }
  cout <<
    "Simulation of wind speed for a flight simulator created
successfully" <<
    endl;
  //close the file
  windsimulation.close ();
}

//main function
int
main ()
{
  //open and read values from configFile
  ifstream conf;
  conf.open ("simulationConfiguration.txt", ios::in);

  //check file opening
  if (conf.fail ())
    {
      cerr << "open file failed \n";
      exit (0);
    }
  //store data of each line in array
  int array1[4], array2[5], array3[5];
  for (int x = 0; x < 4; x++)
    {
      conf >> array1[x];
```

```
    }
  for (int x = 0; x < 5; x++)
    {
      conf >> array2[x];
    }
  for (int x = 0; x < 5; x++)
    {
      conf >> array3[x];
    }

  configloader (array1, array2, array3);

  //close file
  conf.close ();
  return 0;
}
```

Step 5: Test and Verification

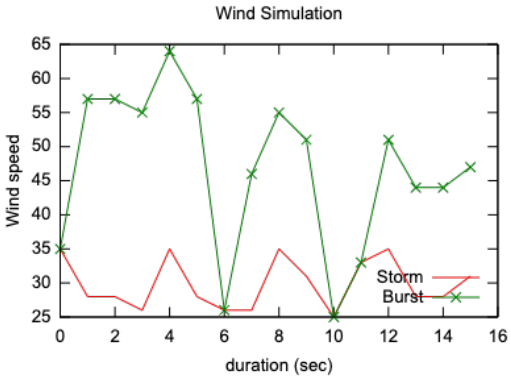
Test case: Increasing Storm Occurrence probability

WindSimulation.txt

	duration	windspeed	stormspeed	burstspeed	
1	duration	windspeed	stormspeed	burstspeed	
2	0	35	35	35	
3	1	28	57	57	1
4	2	28	57	57	1
5	3	26	55	55	1
6	4	35	64	64	1
7	5	28	57	57	1
8	6	26	26	26	
9	7	26	46	46	1
10	8	35	55	82	1
11	9	31	51	78	1
12	10	25	25	52	
13	11	33	33	60	
14	12	35	51	78	1
15	13	28	44	71	1
16	14	28	44	44	1
17	15	31	47	47	1

simulationConfiguration.txt

```
1  30 5 15 1
2  55 10 30 2 6
3  25 15 40 1 5
```



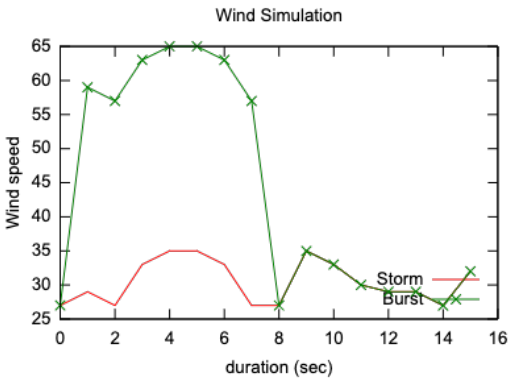
Test case: Decreasing Storm Occurrence probability

WindSimulation.txt

	duration	windspeed	stormspeed	burstspeed	
1	duration	windspeed	stormspeed	burstspeed	
2	0	27	27	27	
3	1	29	59	79	1
4	2	27	57	77	1
5	3	33	63	83	1
6	4	35	65	85	1
7	5	35	65	85	1
8	6	33	63	83	1
9	7	27	57	57	1
10	8	27	27	27	
11	9	35	35	35	
12	10	33	33	33	
13	11	30	30	30	
14	12	29	29	29	
15	13	29	29	29	
16	14	27	27	27	
17	15	32	32	32	

simulationConfiguration.txt

```
1  30 5 15 1
2  10 10 30 2 6
3  25 15 40 1 5
```



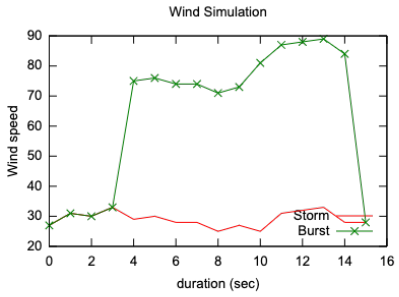
Test case: Increasing Storm amplitude

WindSimulation.txt

	duration	windspeed	stormspeed	burstspeed	
1	0	27	27	27	
2	1	31	31	31	
3	2	30	30	30	
4	3	33	33	33	
5	4	29	75	95	1
6	5	30	76	96	1
7	6	28	74	94	1
8	7	28	74	94	1
9	8	25	71	91	1
10	9	27	73	93	1
11	10	25	81	101	1
12	11	31	87	107	1
13	12	32	88	108	1
14	13	33	89	129	1
15	14	28	84	124	1
16	15	28	28	68	

simulationConfiguration.txt

```
1  30 5 15 1
2  10 10 70 2 12
3  15 20 60 1 9
```



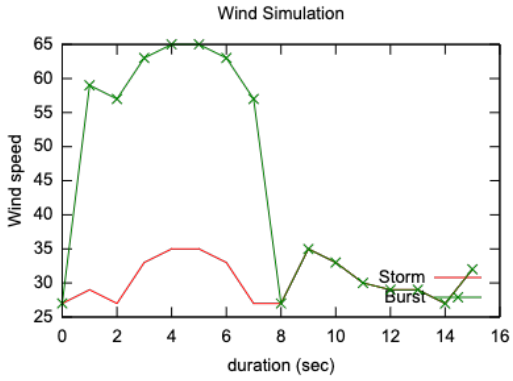
Test case: Decreasing Storm amplitude

WindSimulation.txt

	duration	windspeed	stormspeed	burstspeed	
1	0	27	27	27	
2	1	29	59	79	1
3	2	27	57	77	1
4	3	33	63	83	1
5	4	35	65	85	1
6	5	35	65	85	1
7	6	33	63	83	1
8	7	27	57	57	1
9	8	27	27	27	
10	9	35	35	35	
11	10	33	33	33	
12	11	30	30	30	
13	12	29	29	29	
14	13	29	29	29	
15	14	27	27	27	
16	15	32	32	32	

simulationConfiguration.txt

```
1  30 5 15 1
2  10 10 30 2 6
3  25 15 40 1 5
```



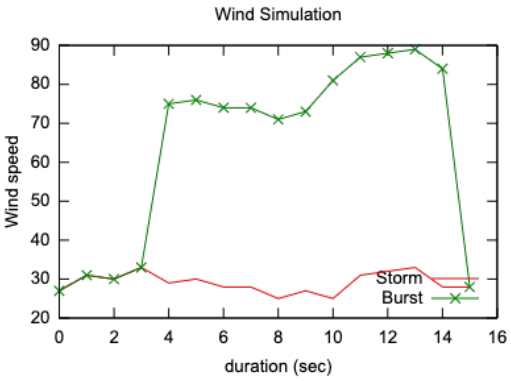
Test case: Increasing Storm duration

WindSimulation.txt

	duration	windspeed	stormspeed	burstspeed	
2	0	27	27	27	
3	1	31	31	31	
4	2	30	30	30	
5	3	33	33	33	
6	4	29	75	95	1
7	5	30	76	96	1
8	6	28	74	94	1
9	7	28	74	94	1
10	8	25	71	91	1
11	9	27	73	93	1
12	10	25	81	101	1
13	11	31	87	107	1
14	12	32	88	108	1
15	13	33	89	129	1
16	14	28	84	124	1
17	15	28	28	68	

simulationConfiguration.txt

```
1  30 5 15 1
2  10 10 70 2 12
3  15 20 60 1 9
```



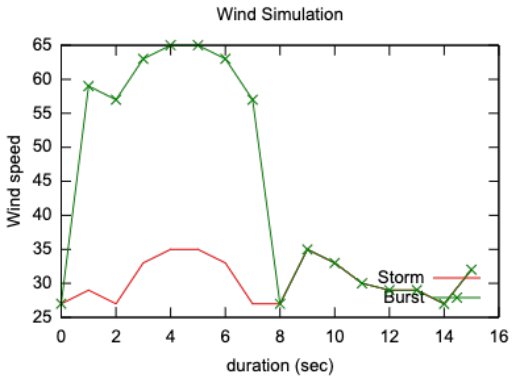
Test case: Decreasing Storm duration

WindSimulation.txt

	duration	windspeed	stormspeed	burstspeed	
2	0	27	27	27	
3	1	29	59	79	1
4	2	27	57	77	1
5	3	33	63	83	1
6	4	35	65	85	1
7	5	35	65	85	1
8	6	33	63	83	1
9	7	27	57	57	1
10	8	27	27	27	
11	9	35	35	35	
12	10	33	33	33	
13	11	30	30	30	
14	12	29	29	29	
15	13	29	29	29	
16	14	27	27	27	
17	15	32	32	32	

simulationConfiguration.txt

```
1  30 5 15 1
2  10 10 30 2 6
3  25 15 40 1 5
```



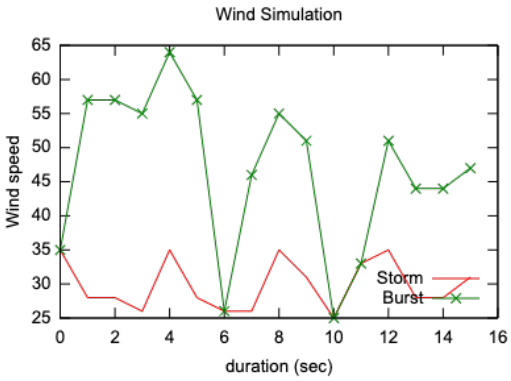
Test case: Increasing Microburst Occurrence probability

WindSimulation.txt

	duration	windspeed	stormspeed	burstspeed	
1	0	35	35	35	
2	1	28	57	57	1
3	2	28	57	57	1
4	3	26	55	55	1
5	4	35	64	64	1
6	5	28	57	57	1
7	6	26	26	26	
8	7	26	46	46	1
9	8	35	55	82	1
10	9	31	51	78	1
11	10	25	25	52	
12	11	33	33	60	
13	12	35	51	78	1
14	13	28	44	71	1
15	14	28	44	44	1
16	15	31	47	47	1

simulationConfiguration.txt

```
1  30 5 15 1
2  55 10 30 2 6
3  25 15 40 1 5
```



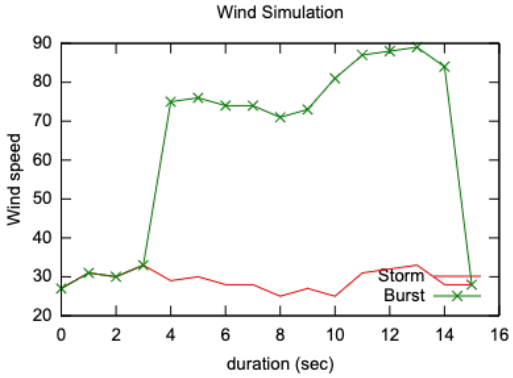
Test case: Decreasing Microburst Occurrence probability

WindSimulation.txt

	duration	windspeed	stormspeed	burstspeed	
1	0	27	27	27	
2	1	31	31	31	
3	2	30	30	30	
4	3	33	33	33	
5	4	29	75	95	1
6	5	30	76	96	1
7	6	28	74	94	1
8	7	28	74	94	1
9	8	25	71	91	1
10	9	27	73	93	1
11	10	25	81	101	1
12	11	31	87	107	1
13	12	32	88	108	1
14	13	33	89	129	1
15	14	28	84	124	1
16	15	28	28	68	

simulationConfiguration.txt

```
1  30 5 15 1
2  10 10 70 2 12
3  15 20 60 1 9
```



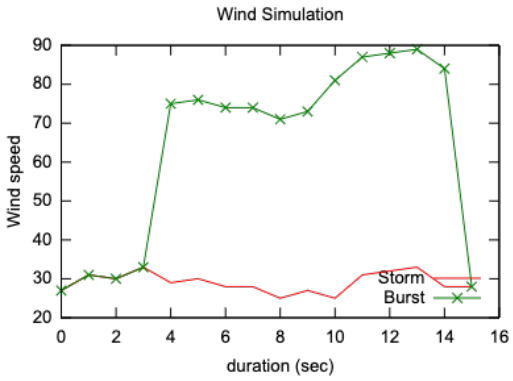
Test case: Increasing Microburst amplitude

WindSimulation.txt

	duration	windspeed	stormspeed	burstspeed	
1	duration	windspeed	stormspeed	burstspeed	
2	0	27	27	27	
3	1	31	31	31	
4	2	30	30	30	
5	3	33	33	33	
6	4	29	75	95	1
7	5	30	76	96	1
8	6	28	74	94	1
9	7	28	74	94	1
10	8	25	71	91	1
11	9	27	73	93	1
12	10	25	81	101	1
13	11	31	87	107	1
14	12	32	88	108	1
15	13	33	89	129	1
16	14	28	84	124	1
17	15	28	28	68	

simulationConfiguration.txt

```
1   30 5 15 1
2   10 10 70 2 12
3   15 20 60 1 9
```



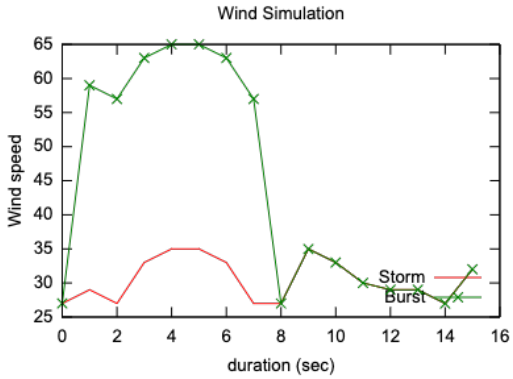
Test case: Decreasing Microburst amplitude

WindSimulation.txt

	duration	windspeed	stormspeed	burstspeed	
1	duration	windspeed	stormspeed	burstspeed	
2	0	27	27	27	
3	1	29	59	79	1
4	2	27	57	77	1
5	3	33	63	83	1
6	4	35	65	85	1
7	5	35	65	85	1
8	6	33	63	83	1
9	7	27	57	57	1
10	8	27	27	27	
11	9	35	35	35	
12	10	33	33	33	
13	11	30	30	30	
14	12	29	29	29	
15	13	29	29	29	
16	14	27	27	27	
17	15	32	32	32	

simulationConfiguration.txt

```
1   30 5 15 1
2   10 10 30 2 6
3   25 15 40 1 5
```



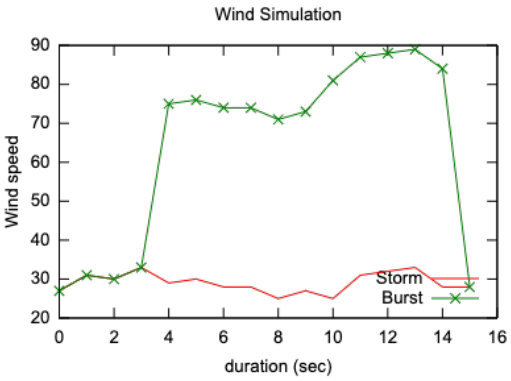
Test case: Decreasing Microburst duration

WindSimulation.txt

	duration	windspeed	stormspeed	burstspeed	
2	0	27	27	27	
3	1	31	31	31	
4	2	30	30	30	
5	3	33	33	33	
6	4	29	75	95	1
7	5	30	76	96	1
8	6	28	74	94	1
9	7	28	74	94	1
10	8	25	71	91	1
11	9	27	73	93	1
12	10	25	81	101	1
13	11	31	87	107	1
14	12	32	88	108	1
15	13	33	89	129	1
16	14	28	84	124	1
17	15	28	28	68	

simulationConfiguration.txt

```
1  30 5 15 1
2  10 10 70 2 12
3  15 20 60 1 9
```



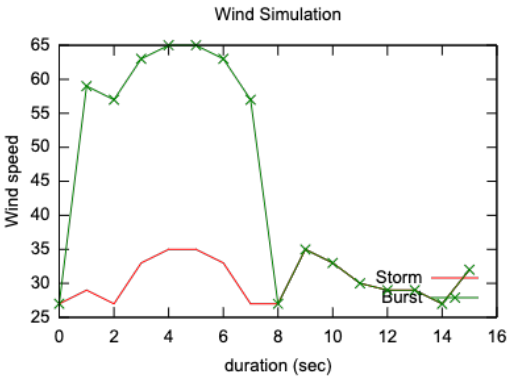
Test case: Increasing Microburst duration

WindSimulation.txt

	duration	windspeed	stormspeed	burstspeed	
2	0	27	27	27	
3	1	29	59	79	1
4	2	27	57	77	1
5	3	33	63	83	1
6	4	35	65	85	1
7	5	35	65	85	1
8	6	33	63	83	1
9	7	27	57	57	1
10	8	27	27	27	
11	9	35	35	35	
12	10	33	33	33	
13	11	30	30	30	
14	12	29	29	29	
15	13	29	29	29	
16	14	27	27	27	
17	15	32	32	32	

simulationConfiguration.txt

```
1  30 5 15 1
2  10 10 30 2 6
3  25 15 40 1 5
```



User guide

Determining a simulation of wind speed to a flight simulator. The software incorporates the simulation of storms and microbursts during the storm. The user will be able to see four files. Windspeed, storm, microburst and the last one contains all the values and binary value to each line signifying whether a storm is active at the same time. If the simulation is executed successfully, you will get a message "Done Reading Data successfully".