Joshua Shapiro | 2 March 2017

Module 1: Preliminaries

1.1. Are integers also real numbers? What are rational numbers and how are they different from integers or reals?

1.1.1. Yes, integers are a subset of real numbers.

1.1.2. Rational numbers are numbers that can be expressed as the division of two integers. They can be integers, have a finite number of digits after the decimial point, or continue infinitely with a repeating pattern of digits. They are not necessarily integers, however they are a subset of the real numbers.

1.2. What about rationals? Is there a rational number between every two rationals?

1.2.1. Yes. Taking the average of two rational numbers will yield another rational, since this is the same as finding two fraction's GCD and finding the average between the two new numerators. The quotient of the new numerator with the GCD will yield the rational number between the two initial rational numbers.

1.3. How do the sizes of the naturals compare with the integers? What about the size of the rationals compared to integers?

1.3.1. The size of the naturals and integers have the same cardinality.

1.3.2. The size of the rationals and integers also have the same cardinality, since the rationals are countable which implies they have a 1:1 mapping to the naturals.

1.4. CODE: In the above example, we knew that $f(a) < 0$. Re-write the pseudocode to make it more general: replace the test $f(m) < 0$ with a test to see if they are of the same sign. Modify the code in Bisection.java accordingly.

1.4.1. We want to decide if f(m) is of the same sign of a (then a = m) or not (then b = m). So…

1.4.2. If (f(m) * f(a) > 0) a=m

1.4.3. This works since the product of 2 numbers with the same sign will always be positive, while the product to 2 numbers with different signs will be negative

1.5. Examine the code from question 4

1.5.1. Use a calculator to compute $\sqrt{3}$ and compare with the output of the program. How would you increase the accuracy of the program?

1.5.1.1. Change the condition of the while loop to to be ">" than a smaller number.

1.5.2. Can you make the code a little more efficient?

1.5.2.1. Not that I can tell. Maybe calculate f(m) only once instead of twice and store in a variable?

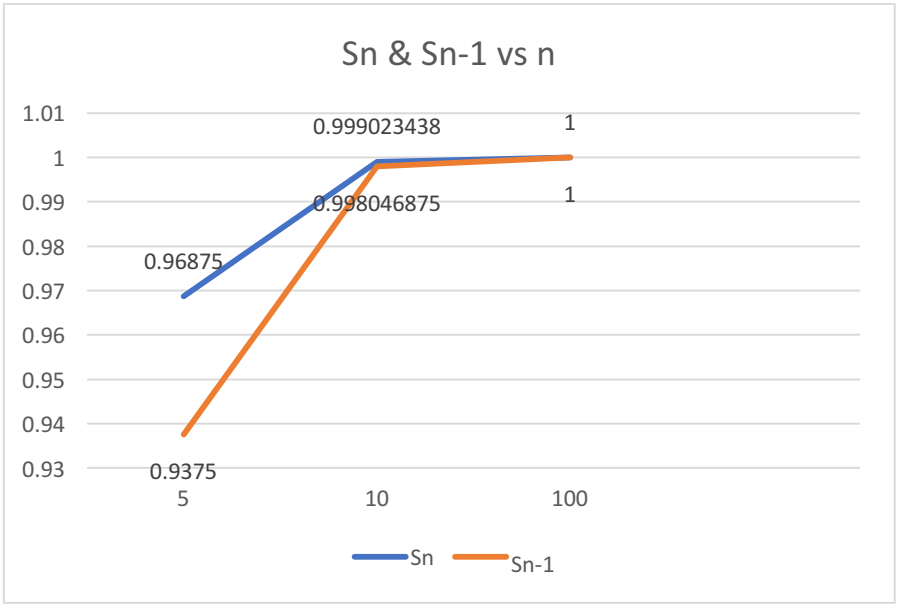
1.5.3. Can the bisection() method be written recursively? What would that do to efficiency?

1.5.3.1. Yes, however you threaten to blow the stack if your initial bounds are too far away. That wouldn't really change efficiency, though would take up more space.

1.6. CODE: Add code to Zeno.java to compute S_n for any value of n. Try $n = 5, 10, 100$. What do you observe?

1.6.1. The sum S_n gets closer and closer to 1 (at n=100, the result is 1).

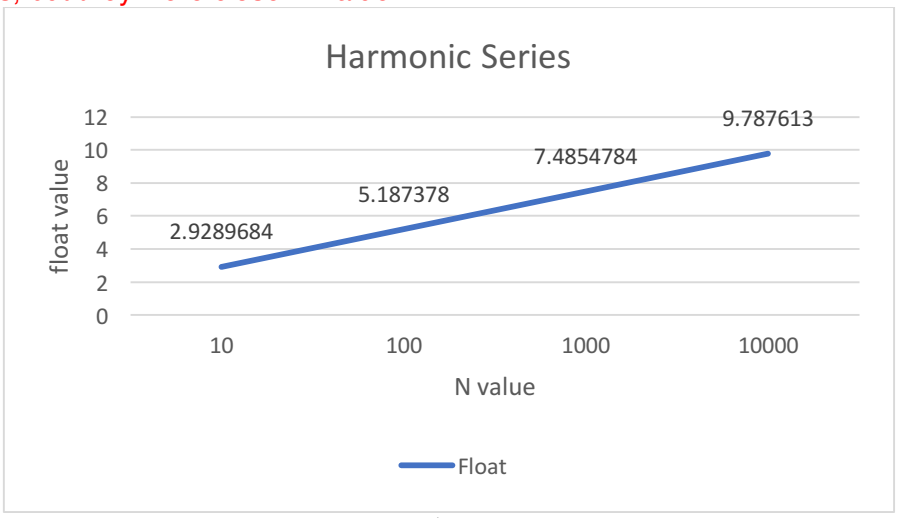
1.7. CODE: Modify the above to compute S_{n-1} for the sum in Zeno's paradox. Add the formula to Zeno.java and compare. Plot S_n vs. n.



1.7.1.

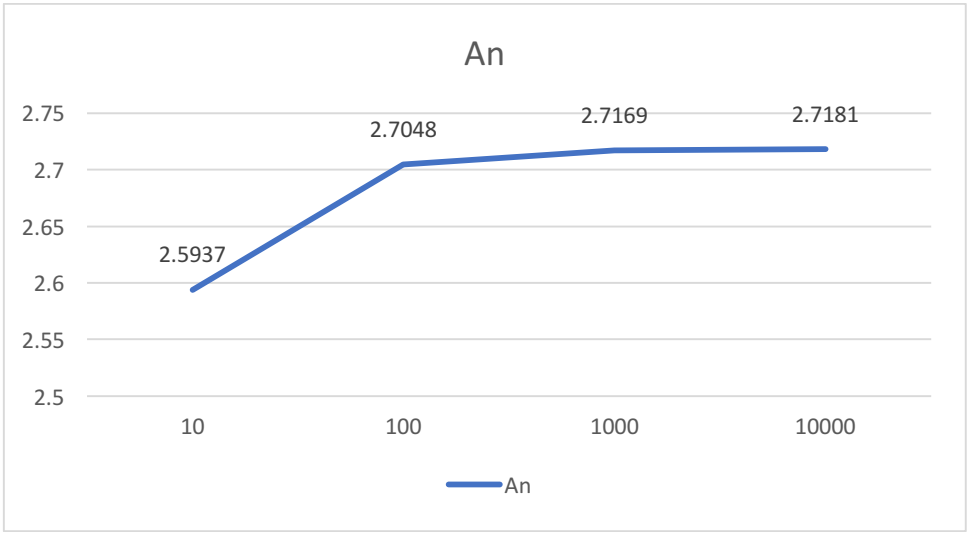
1.8. CODE: Add code to Harmonic.java and compare the output from the two computations: using double vs using float. Try different value of n. What do you think the sum is converging to? Plot H_n vs n.

1.8.1. The harmonic sum does not converge. The doubles had more digits than the floats, but they were close in value.



1.8.2.

1.9. CODE: Write code to compute $A_n = (1 + \frac{1}{n})^n$ for various values of n. Write your code in the compute() method of SequenceExample.java. Plot A_n vs n.



1.9.1.

1.10. CODE: Write code to print the first few terms of the sequence $C_n = \frac{\sin(n)}{n}$. How is this sequence different from the ones we've seen so far?

1.10.1. It converges to 0, but goes above and below the number it converges to over time.

1.11. CODE: Write code in RandomSequence2.java to print the first 10 terms of the sequence $V_n$above. You will also need UniformRandom.java.

1.11.1. Does the sequence U_n have a limit?

1.11.1.1. No

1.11.2. Does the sequence V_n have a limit?

1.11.2.1. Yes, 0.5

1.11.3. How are these sequences different from the ones we've seen before? After writing your code to compute V_n above, examine the code in RandomSequence3.java. What is the difference in the two approaches?

1.11.3.1. At each n value, a different random number can be generated, yet Vn still converges. In RandomSequence3, each Vn value doesn't require n new values of Un to be created. It simply tacks on to the Vn-1 value. In RandomSequence2, new Uns were created for every Vn value.

1.12. CODE: First, examine the code in RandomSequence4.java and verify that it prints out possible random values, or *samples*, of U5. Modify the code to print values of U493. Next, download Histogram.java and print out a histogram for U5with different numbers of samples: 10, 100, and 10000. What do you notice? Is it intuitive? How do the historgrams for U_5 differ from the equivalent histograms for U_{493}?

1.12.1. The number of samples doesn't affect the distribution by much. However, the larger the sample the more fine-grained the distribution is.U5 and U493 have the same distributions.

1.13. CODE: Modify Histogram.java and compute and print a histogram for V_n with different numbers of samples: 10, 100, and 10000.

1.13.1. How does the histogram for V_5 differ from the histogram for V_{493}?

1.13.1.1. V5 looks like a bell curve since there is a chance 5 values could all be low or high, while V493 only has values at bucket 4 & 5 since the numbers begin to converge at around 0.5.

1.13.2. Print the histogram for V_{10000} and explain the result.

1.13.2.1. It's close to V493. The values converge at 0.5 so only the 4 and 5

buckets get filled.

1.14. CODE: Modify Histogram.java to compute and print a histogram for Wn with different numbers of samples: 10, 100, 10000.

1.14.1. How does the histogram for W5 differ from the histogram for W493?

1.14.1.1. They both have the about the same shape.

1.14.2. Print and then plot the histogram for W10000.

1.14.3. Histogram for W10000

1.14.4. b=0 bins[b]=1369

1.14.5. b=1 bins[b]=1194

1.14.6. b=2 bins[b]=952

1.14.7. b=3 bins[b]=614

1.14.8. b=4 bins[b]=424

1.14.9. b=5 bins[b]=236

1.14.10. b=6 bins[b]=103

1.14.11. b=7 bins[b]=48

1.14.12. b=8 bins[b]=14

1.14.13. b=9 bins[b]=6

1.14.14. We'll now change the type of random generation. Create a means of generating U'n, a random variable that's different from Un.

1.14.14.1. U'n = random^random

1.14.15. Run your program to estimate what V'n is converging to.

1.14.15.1. V'n converges to 0.7, which is about 0.5^0.5

1.14.16. Plot the histograms for W'5 and W'493.

1.14.17. Histogram for W'5

1.14.18. b=0 bins[b]=1463

1.14.19. b=1 bins[b]=1346

1.14.20. b=2 bins[b]=1026

1.14.21. b=3 bins[b]=705

1.14.22. b=4 bins[b]=345

1.14.23. b=5 bins[b]=97

1.14.24. b=6 bins[b]=5

1.14.25. b=7 bins[b]=0

1.14.26. b=8 bins[b]=0

1.14.27. b=9 bins[b]=0

1.14.28. Histogram for W'493

1.14.29. b=0 bins[b]=1166

1.14.30. b=1 bins[b]=736

1.14.31. b=2 bins[b]=447

1.14.32. b=3 bins[b]=234

1.14.33. b=4 bins[b]=113

1.14.34. b=5 bins[b]=37

1.14.35. b=6 bins[b]=11

1.14.36. b=7 bins[b]=2

1.14.37. b=8 bins[b]=1

1.14.38. b=9 bins[b]=0

1.15. What types of "odd" cases do we have to worry about in real life? Can you think of a function such that the number 22 is a "bad" input value (i.e., shouldn't be allowed)?

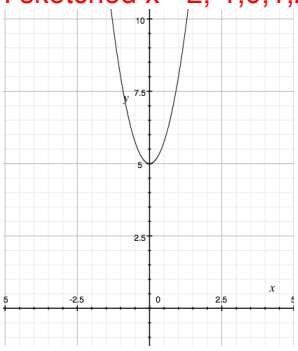
1.15.1. You would have to worry about x=2 if the function is y=1/(x-2) since you don't

want the denominator to equal 0.

1.16. Get out a piece of paper and draw (by hand) the function f(x)=3x^2+5. What are the domain and range of this function? How much of the domain and range did you sketch out?

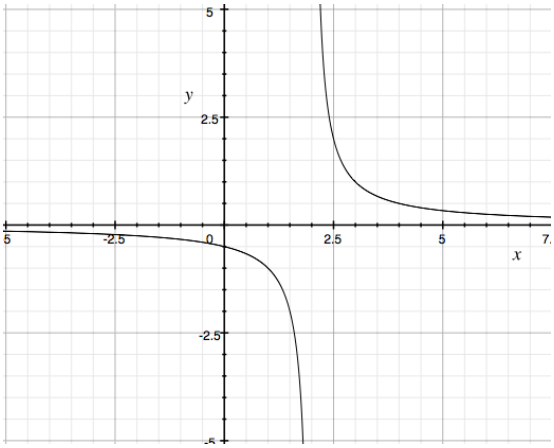
1.16.1. The domain is –infinity to +infinity. The range is 5 to +infinity.

1.16.2. I sketched x=-2,-1,0,1,2.

1.16.3. 

1.17. CODE: Download, compile and execute the above program. Change the last line to make the output friendlier.

1.18. CODE: Modify the above program to compute the function f(x) = 1/(x-2). Use the program to generate some values and draw a graph of the function. What happens when you enter x = 2?

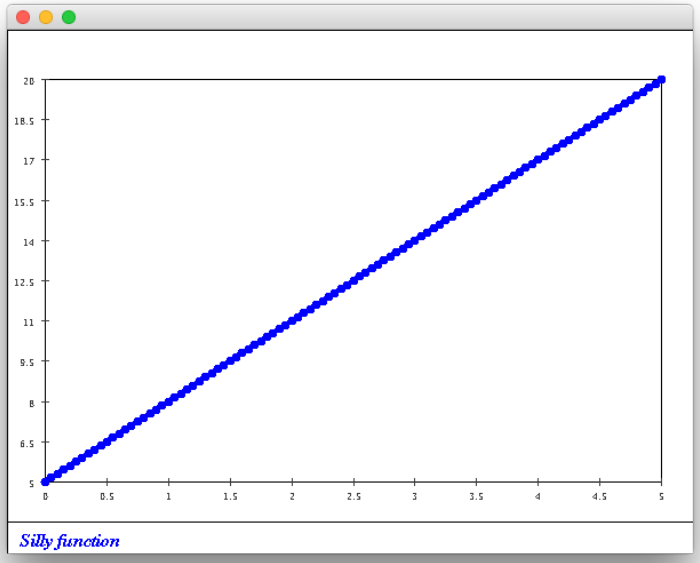
1.18.1. 

1.18.2. x = 2 returns infinity.

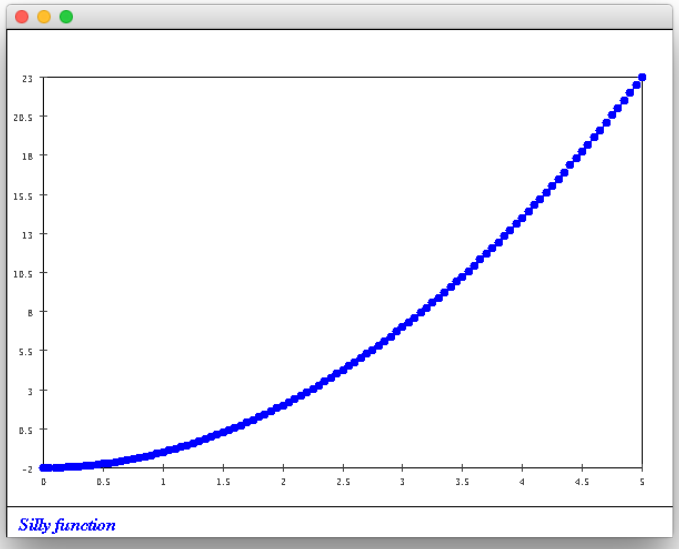
1.19. CODE: Modify the above program to print out function values for x in the range of -10 to -1.

1.20. CODE: Modify the above program to show the shape of the function f(x) = 1/(x-2) in the range [0,5].

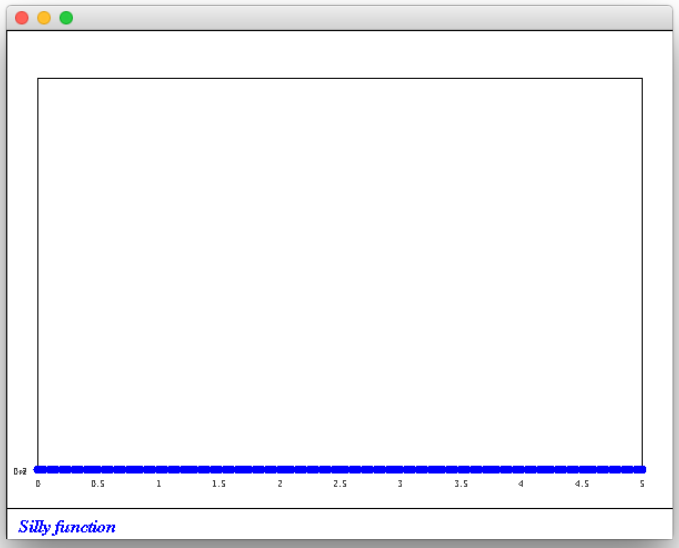
1.21. CODE: For each of the functions below, generate 100 values in the range [0,10] and feed them into a Function object. Then display the result.



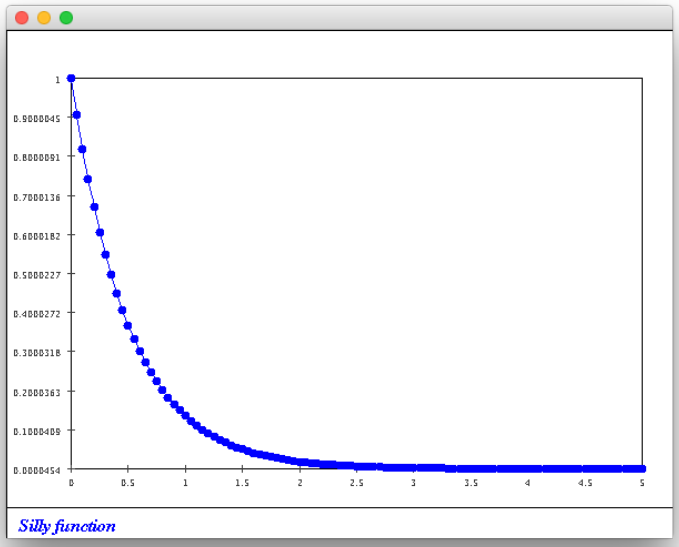
1.21.1. 3x+5



1.21.2. x^2-2



1.21.3. 5/(x^2)



1.21.4. e^(-2x)

1.22. CODE: Modify the above code to draw f(x) = 3x+5 and g(x) = 3x+10. Use only 50 points.

1.23. Did it matter whether we used g-f or f-g above? What is the distance between the functions f(x) = 3x+5 and h(x) = 20? What happens when you use f-h vs h – f?

1.23.1. Yes. Since we are taking the difference instead of absolute value, the distances will be different. H-f = -1.63*10^-13. F-h = -1.63*10^-13.

1.24. CODE: Modify the above example so that the loop has only one line of code (and is therefore more compact).

1.25. CODE: Modify the example to use 100 points instaed of 50. What do you notice? What does this tell you about the method we're using to compute distance?

1.25.1. The distance doubles. We should probably divide distance by the number of

points sampled to get an average, so the distance isn't dependent on the range and the sample number.

1.26. CODE: Modify the above example as follows:
  1.26.1. First, use 100 points insetad of 50. Then 1000. What do you observe?
    1.26.1.1. 100 yields 5.05, 1000 yields 5.005. These are fairly close together, which is good.
  1.26.2. Next, change the range from [0,10] to [0,5]. What is the distance between f and g using this range?
    1.26.2.1. 2.55 and 2.505 for 100 and 1000 respectively.
  1.26.3. Repeat the above for f and h(x) = 20. Does the distance measure for functions make sense?
    1.26.3.1. 3.82 and 3.75 for 100 and 1000 respectively with a range from 0 to 5. 7.64 and 7.51 for 100 and 1000 respectively with a range from 0 to 10. This looks generally correct for a distance measure, since it changes with range but not much with sample number.

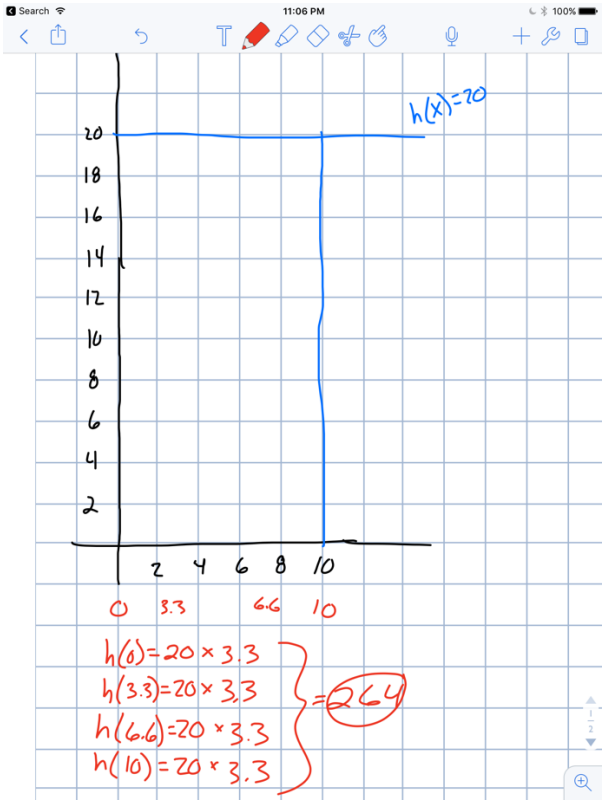
1.27. CODE: Now modify the above example to use [0,5] as the range:
  1.27.1. What is the distance between f and g using this range?
    1.27.1.1. 51.0
  1.27.2. Next, use 100 points instead of 50. Then use 1000 ponts. Is our modified measure severely affected by the number of points?
    1.27.2.1. 100: 50.5. 1000: 50.05. No, the measure is not severely affected.

1.28. Why does this code produce exactly the same result?
  1.28.1. It is simply doing the same calculation but including the interval division in the addition to the sum line, not separately. It's like adding 1/10 to itself 10 times instead of doing 10/10.

1.29. CODE: Change the second line to double interval = (10-0) / 50;. What do you obvserve? Explain.
  1.29.1. The distance becomes 0. This is because the division of 10/50 is the quotient of 2 integers, which Java will round down to the nearest integer. This makes 10/50 = 0. When the sum is calculated, we multiply f-g by 0 resulting in a distance of 0.

1.30. Consider the function h(x) = 20 in the range [0,10]. Draw this on paper, you should have a rectangle of width 10 and height 20. Next, use 4 equally-spaced x-values and hand-execute the above program with h(x) instead of f(x). Can you relate the calculations to the area of the rectangle?
  1.30.1. The sum is the area plus 64. This is because 4 x values yields 3 intervals. But then the calculation on 4 x values sums 4 intervals, adding an extra interval to the final answer.

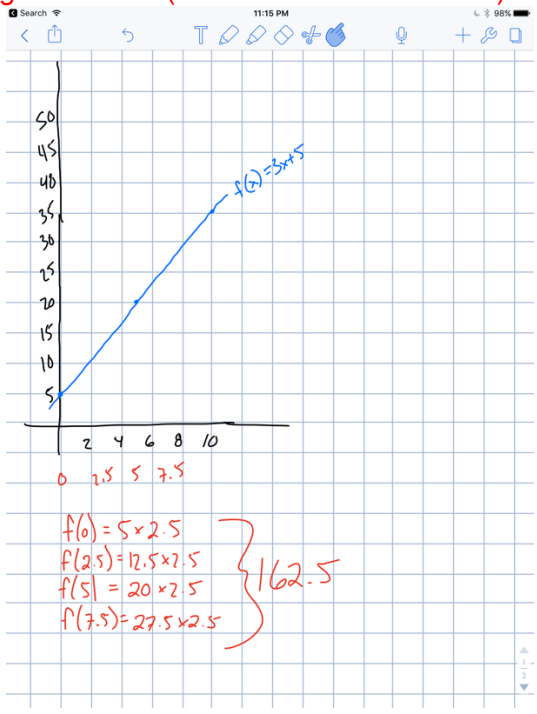
1.30.2.



1.31. Now consider the function f(x) = 3x + 5 in the range [0,10]. Draw this on paper and repeat the above steps (using 4 intervals) to compute the distance to the x-axis. How does this relate to the area? What happens when we use more intervals? Try 50 intervals, then try 1000. Can you calculate the area exactly by hand?

1.31.1. The more intervals used the closer to the area we get. The picture only shows using 4 intervals (not 4 x values like above).

1.31.2.



1.32. Draw the two functions f(x) = 3x+5 and g(x) = 20 on paper. What is the relationship between our distance measure and area?

1.32.1. Our distance measure is the absolute value of the area of f(x) – the area of g(x).

1.32.2. 

1.33. Why is this true?

1.33.1. We know that the ratios of the 3 sides of a right triangle must stay the same if the angles stay the same. So if we double one side and we want to keep the angles the same, the other two sides have to double as well.

1.34. CODE: Modify the above program to print out sin(4pi/3) and answer the following:

1.34.1. Why is the result negative? What does it have to do with actual angles and our earlier definition of the sin function?

1.34.1.1. The result is negative because sine is a periodic function alternating between -1 and 1. Since 4pi/3 is 240 degrees and sine is negative between 180 and 360 degrees, this is not surprising.

1.34.2. Convert the angle from radians to degrees (perhaps by modifying the program). Draw the result on paper. See if this helps answering the above question.

1.34.2.1. Since 4pi/3 is 240 degrees and sine is negative between 180 and 360 degrees, this is not surprising.

1.34.3. Modify the code to print out sin(2pi + 4pi/3). Can you explain the result?

1.34.3.1. It has the same result as 4pi/3. This is because the period size of sine is 2pi. So adding it to 4pi/3 is equivalent to not adding it at all.

1.35. Explain why both functions are periodic. What is the period?

1.35.1. One way to look at sine and cosine is the x and y components of a triangle in a unit circle. As the triangle traverses this circle, the x and y components vary. However, they never get larger than 1 or smaller than -1. This is why the function are periodic. The period is 2pi.

1.36. CODE: Use the above approach to find the minimum value of cos(x) for x in the range of [0,2pi]. What could we do to make the result more accurate?

1.36.1. The minimum value is -1. It occurs around 3.1. By making the interval smaller in the for loop, we would get a more accurate result.

1.37. What good are such “rules”? Why are they useful? Why not just use the data directly?

1.37.1. Rules act as a simplified way to explain a lot of information. Instead of having to list every value in a data table, a function can be used as a rule to convert any input into a desired output.

1.38. Can you guess the function that produced this data:

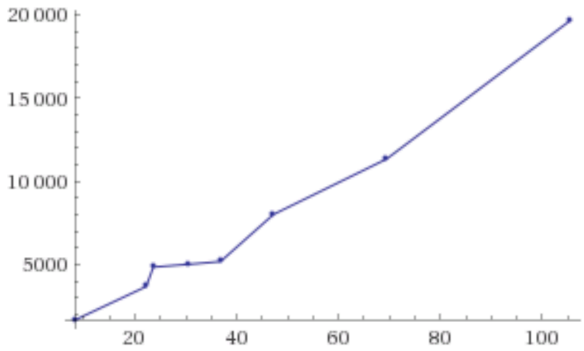
1.38.1. It is linear, but hard to get the exact function

1.39. Identify the linear function in the previous exercise.

1.39.1. The function ends up being f(x) = pi*x + 2.72

1.40. Draw the graph for this data. Is the curve linear or non-linear? The data is from a historic 1929 paper.

Plot:

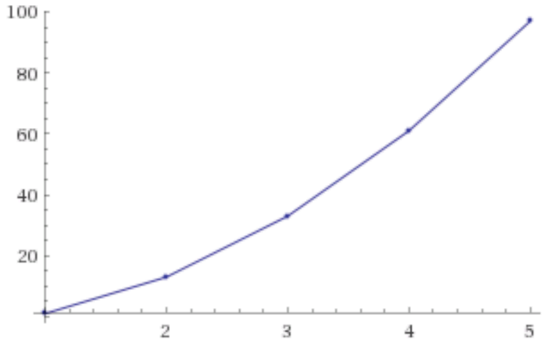


1.40.1.

1.40.2. It is nonlinear

1.41. Draw the graph for this data. Is the curve linear or non-linear?

Plot:



1.41.1.

1.41.2. It is nonlinear

1.42. CODE: Write a small program to compute x^2 + f(x)^2

1.43. CODE: Execute the above program. Does the function g(x) look familiar? Modify the above code to fill in the f and g values into Function objects and display them.

1.43.1. G is linear, F is non linear. G looks like the derivative of F.

1.44. CODE:

1.44.1. Use d = 0.01 and plot the derivative function g(x) along with f(x) when f(x) = 3x^2 + 5.

1.44.2. Compare this derivative function nwith that of f(x) = 3x^2. Explain what you observe.

1.44.3. Use two points on the derivative function and write down the function as a

formula.

1.44.4. Explore what happens with different d values, for example try d=1, d=0.1, d= 0.0001, d=0.00001.

1.45. CODE: Suppose the derivative of f(x) is g(x) = -sin(x) and that we know f(0) = 1. Modify the above code to compute f(x) in the interval [0,2pi] and display the result.

1.46. CODE: Donwload Simulator.class and UnknownFunctionDerivative.java and exdecute. What is the relationship between x and f(x)?

1.47. CODE: How can you simplify and speed up the code above? What does the Java library use to compute sin(x)? Compare the accuracy of the above function with Java's sin function.

1.48. CODE: Download AccelCar.java, Function.java, and SimplePlotPanel.java. This is a simple model of an accelerating vehicle – the goal is to start at the left and reach the right in the least time possible but also such that the velocity at the end is as close to zero as possible.

1.48.1. Compile and execute AccelCar. Then try to determine a good acceleration "function" by hand.

1.48.2. Download MyController,ava, compile and execute. Examine the code – you should see a sample acceleratаion function.

1.48.3. Implement a better acceleration function in MyController.

1.48.4. How would you systematically search for the optimal acceleration function?

1.49. Construct a sequence of distinct functions so that the limit is f(x) = x.