Lab 3: Left and Right Sum convergence

■ Pictures of left and right sums

In this notebook we want to illustrate the convergence of left and right sums. In the last lab we did this with a table. Now we want to look at it graphically. First load the Graphics`FilledPlot` package and then execute the following cell. This defines two (fairly crude) commands, LeftSumPlot[ ] and RightSumPlot[ ], designed to be used in much the same way as LeftSum[ ] and RightSum[ ] themselves, only now the output is a filled graph showing the actual rectangles being used in calculating the corresponding Riemann sum.

```
<< Graphics`FilledPlot`
```

```
LeftSumPlot[f_, a_, b_, n_, color_] :=
  FilledPlot[f[Floor[x * n / (b - a)] (b - a) / n], {x, a, b}, Fills → {{{1, Axis}, color}}];
RightSumPlot[f_, a_, b_, n_, color_] :=
  FilledPlot[f[(Floor[x * n / (b - a)] + 1) (b - a) / n], {x, a, b}, Fills → {{{1, Axis}, color}}];
```

The only difference in usage is that we now also specify a color. (Because these are rather crude commands, you must actually specify a color. You cannot leave the color blank to get the default color as you can with FilledPlot itself.) For example, to show the rectangles making up the left Riemann sum for Sin[x] on the interval $[0, \pi/2]$ for $n = 10$ and color them blue we would use the following.

```
LeftSumPlot[Sin, 0, Pi / 2, 10, RGBColor[0, 0, 1]]
```

■ Specifying Colors

You have seen by example that the standard way to specify a color in *Mathematica* is to do so with RGBColor[r, g, b], where r, g, b, are numbers between 0 and 1 that specify the amount of red, green, and blue to mix for the color. Therefore, RGBColor[0, 0, 1] is pure blue. Since most people would not know what proportions of red, green and blue to get, for example, magenta, there are alternate means of getting colors. One is with the "Color Selector". Put your cursor in the empty last entry of the RightSumPlot below. Go to the Input menu and pick "Color Selector..." There are multiple ways to pick a color. Select one and then click OK. The appropriate RGBColor[ ] command for the color you picked will be inserted where you left the cursor.

```
RightSumPlot[Sin, 0, Pi / 2, 10,    ]
```

A second method is to load the package Graphics`Colors`. This defines many colors by name:

```
<< Graphics`Colors`
```

```
RightSumPlot[Sin, 0, Pi / 2, 10, Pink]
```

To find out what color names are defined in this package (and remember to capitalize!), you can execute the command AllColors (after the package has been loaded).

```
AllColors
```

■ Illustrating Convergence

To illustrate how the left and right sums approximate the area under a curve we want to place a picture of the rectangles and the graph of the function together on the same plot. We can do this with the Show command if we first “name” our pictures. For example, suppose we want to compare the left sum for Sin[x] on [0, π] to the graph of the function. We can execute commands for the LeftSumPlot and Plot, but assign then to names (in this case, ls and fp) so we can refer to them later:

```
ls = LeftSumPlot[Sin, 0, Pi, 10, Blue]
fp = Plot[Sin[x], {x, 0, Pi}, PlotStyle→ Thickness[.01]]
```

We can then show them together by:

```
Show[ls, fp]
```

Show[ ] will draw the graphics objects in order with later objects drawing over earlier ones. Therefore, sometimes you need to be careful with the order you specify things. For example, look what happens is we reverse the order of the two pictures in the previous example:

```
Show[fp, ls]
```

Notice that because Sin[x] is not monotone (either only increasing or only decreasing), the rectangle that make up the left sum are sometimes inside the region and sometime out. The same hold for the right sum.

```
rs = RightSumPlot[Sin, 0, Pi, 10, Red]
Show[rs, fp]
```

However, this is different if the function is monotone. Consider Sin[x] on [0, $\pi/2$] and 1/x on [1, 2]:

```
ls = LeftSumPlot[Sin, 0, Pi / 2, 10, Blue]
fp = Plot[Sin[x], {x, 0, Pi / 2}, PlotStyle→ Thickness[.01]]
rs = RightSumPlot[Sin, 0, Pi / 2, 10, Red]
Show[rs, ls, fp]
```

```
f[x_] := 1 / x
ls = LeftSumPlot[f, 1, 2, 10, Blue]
fp = Plot[f[x], {x, 1, 2}, PlotStyle→ Thickness[.01]]
rs = RightSumPlot[f, 1, 2, 10, Red]
Show[ls, rs, fp]
```

Note that in both cases the right sum is red and the left sum is blue. Which is over/under estimating in each case? The difference between the left and right sums is the area of the region where they are **not** overlapping (the visible red part for Sin[x] and the visible blue part for 1/x). Notice from the pictures that this difference is actually the sum of the two individual errors: $|R_n - L_n| = |A - R_n| + |A - L_n|$, if A is the true area under the graph. Therefore, if we use either the right or the left sum to approximate the area under a **monotonic** function, then $|\text{error}| \leq |R_n - L_n|$. The same does not hold for non-monotonic functions. (Why?) For Sin[x] on [0, π] we have the right sum equal to the left sum (for the same n), but neither equals the true value.

To finish this section we have an animation to illustrate the convergence. Double click on the graph to run the animation and then control the play with the icons in the lower left margin of the notebooks window.

```
myFrame[n_] := Show[RightSumPlot[Sin, 0, Pi / 2, n, Red],
  Plot[Sin[x], {x, 0, Pi / 2}, PlotStyle→ Thickness[.01]], LeftSumPlot[Sin, 0, Pi / 2, n, Blue]]
Needs["Graphics`Animation`"]
Animate[myFrame[2^i], {i, 2, 6, 1}]
```

■ Assignment

For the function y = f(x) =1 - cos($e^{3x/4}$) on the interval [-1,1.5] do the following:
1. Plot the function and determine if it is monotone.
2. For n =10, 20, 40, 80 equally spaced subintervals compute the corresponding left sums, right sums, and error bounds. Display your results in a table with column headings (look up TableHeadings in the *Mathematica* help): n, leftsum, right-sum, error bound.
3. Plot the rectangles corresponding to n =20 for both the right and left sums on the same plot as the function f(x) and use a different color for each of the three graphs.
4. By what factor does the error bound change when the number of subintervals is doubled?