Lab 7: Student Assignment

Week 7, February 22

MAT 229, Spring 2021

Numerical Integration Examples

This lab begins by asking you to evaluate some “initialization code” provided by Prof. Al Hibbard. Under the “Evaluation Tab” above you’ll see “Evaluate Notebook”: select that, and allow initialization. When you’re done, come back up here to the beginning.

Numerical Integration Code: thanks to Al Hibbard -- HibbardA@central.edu

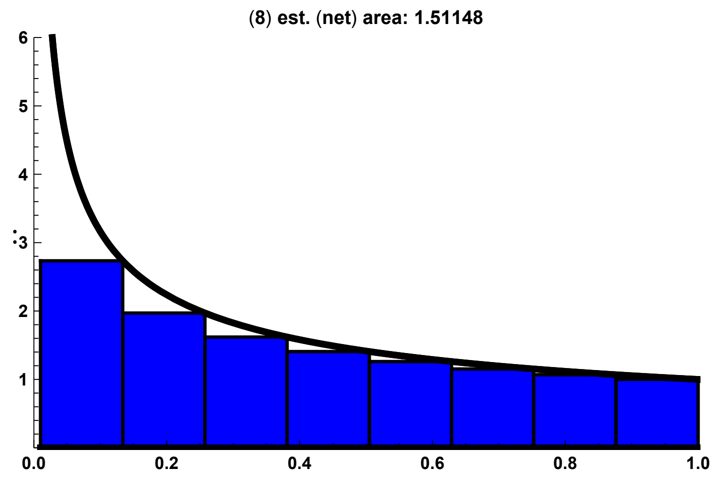
1. Improper Integral example: integrable function with a vertical asymptote

One of our questions from this week’s material asks “What makes $\int_0^1 \frac{1}{\sqrt{x}}\,dx$ an improper integral? Does it converge or not?”
It’s improper because of the vertical asymptote at the left endpoint.
It does converge, because it blows up “slowly enough” that there is actually a finite amount of area under the curve, even as the curve itself heads to infinity.

1. Define f(x)=$\frac{1}{\sqrt{x}}$, a and b as the endpoints.

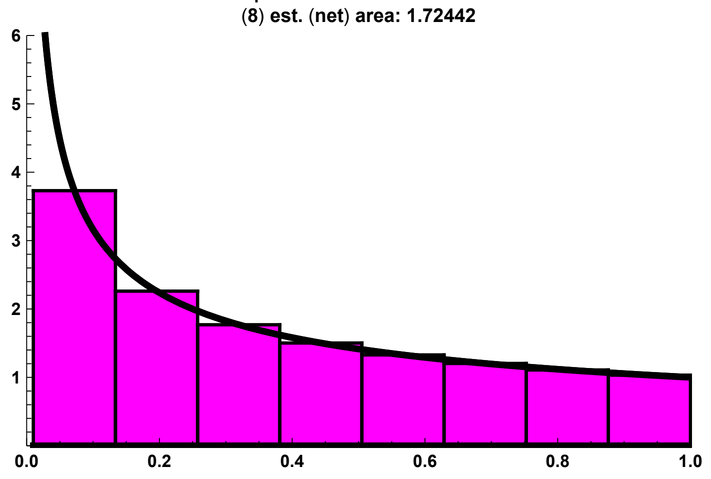
```
In[2196]:= f[x_] := 1 / Sqrt[x]
           a = 0.0;
           b = 1.0;
```

2. Use the right endpoint method to approximate this integral with n=100, using Mathematica’s Sum command. Here it a picture of the situation with n=8:



```
In[2199]:= n = 100;
           dx = (b - a) / n;
           rrr = dx * Sum[f[a + (k - 0) dx], {k, 1, n}]

Out[2201]= 1.85896
```

3. Use the midpoint method to approximate this integral with n=100, using Mathematica's Sum command. Here it a picture of the situation with n=8:



```
In[2202]:= mid = dx * Sum[f[a + (k - 0.5) dx], {k, 1, n}]

Out[2202]= 1.93951
```

4. Why are you unable to use any other of our methods for this definite integral?

Because all the other methods involve stepping on the left endpoint, at which the function not defined.

5. Why are you unable to carry out any error analysis using our error formula for midpoint?

Because the derivatives (in particular, the second derivative) are not bounded on the interval.

6. Nonetheless, you can compute this integral analytically, and so compare the absolute error of both the midpoint and right endpoint methods.

```
In[2203]:= true = Integrate[f[x], {x, a, b}]
           rrrError = Abs[rrr - true]
           midError = Abs[mid - true]

Out[2203]= 2.

Out[2204]= 0.14104

Out[2205]= 0.0604878
```

2. Numerical Integration with data:

An Exercise in Modeling Human Temperature

The Data

The following figure is reproduced from the Book “La Chaleur Animale” (Animal Heat), 1889, by Charles Richet (Nobel Prize in Medicine, 1913).

De cette régularité constante de la courbe thermique il résulte que le minimum de la température se présente toujours à peu près à la même heure, vers trois, quatre ou cinq heures du matin,

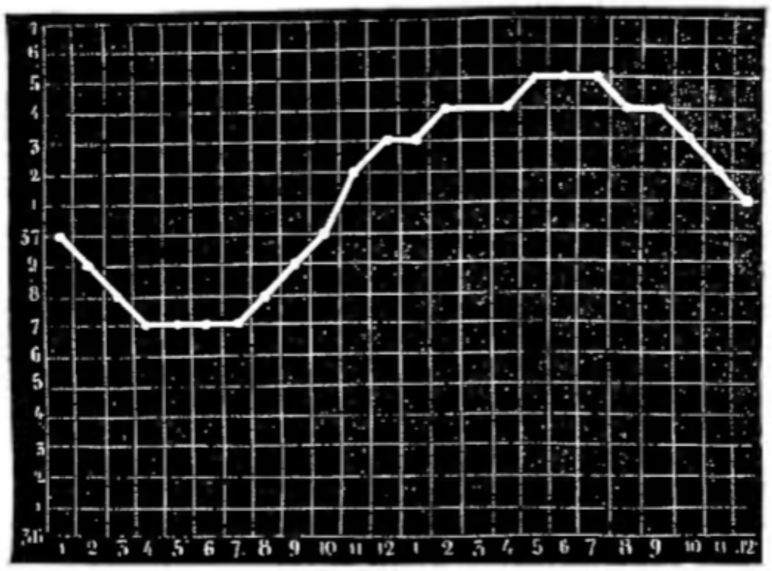


Fig. 3. — Courbe quotidienne de la température rectale chez l'homme. (Jürgensen.)

Cette courbe est déterminée par les mesures très exactes de M. Jürgensen. Elle indique nettement les phases de l'oscillation quotidienne et peut servir de point de comparaison aux observations prises sur les malades.

Maximum : 37°,5 à 5 heures, 6 heures, 7 heures du soir.
Minimum : 36°,7 à 4 heures, 5 heures, 6 heures du matin.
Écart maximum : 0°,8.

I digitized the data (pulled it off that figure) as follows (and you might check me!). Execute the following commands to define the data:

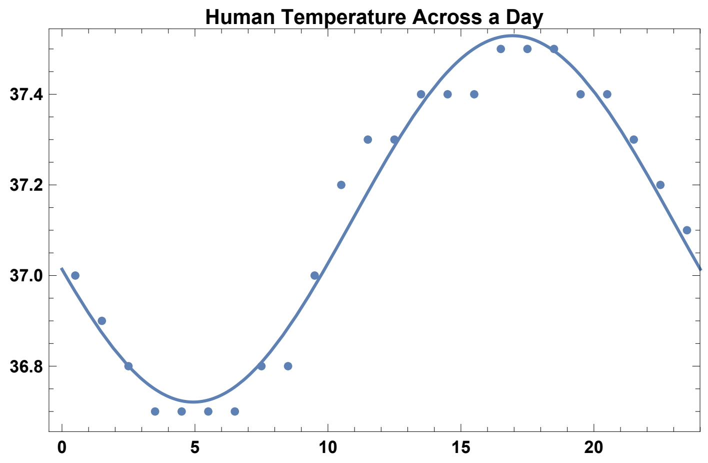
```
In[2206]:= temps = 37 + (* degrees celsius *)
              .1 {0, -1, -2, -3, -3, -3, -3, -2, -2, 0, 2, 3, 3, 4, 4, 4, 5, 5, 5, 4, 4, 3, 2, 1}
           times = {0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5, 10.5, 11.5,
              12.5, 13.5, 14.5, 15.5, 16.5, 17.5, 18.5, 19.5, 20.5, 21.5, 22.5, 23.5}
              (* half hours from midnight *)
           data = Transpose[{times, temps}];

Out[2206]= {37., 36.9, 36.8, 36.7, 36.7, 36.7, 36.7, 36.8, 36.8, 37., 37.2, 37.3,
            37.3, 37.4, 37.4, 37.4, 37.5, 37.5, 37.5, 37.4, 37.4, 37.3, 37.2, 37.1}

Out[2207]= {0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5, 10.5, 11.5,
            12.5, 13.5, 14.5, 15.5, 16.5, 17.5, 18.5, 19.5, 20.5, 21.5, 22.5, 23.5}
```

The model

A model, obtained by non-linear regression is

$$\boxed{37.125 - 0.404182 \,\mathrm{Sin}\left[\frac{1}{12}\pi(1.06355 + t)\right]}$$

and a plot of the data and the model will look like this:



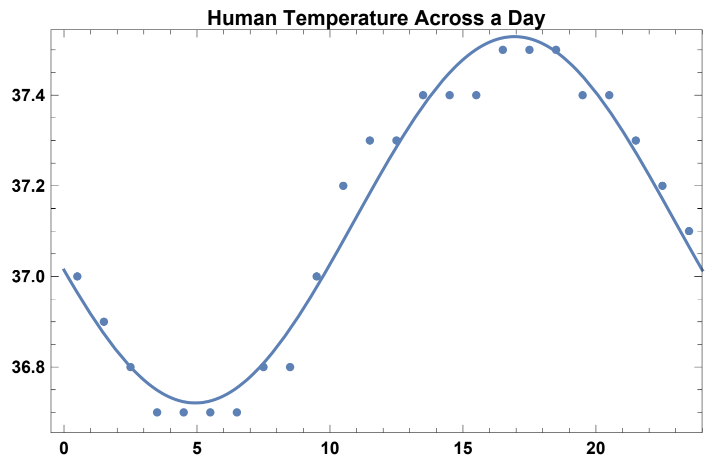
1. Define nlm(t) by the formula above (nlm stands for non-linear model).

```
In[2209]:= nlm[t_] := 37.125` - 0.40418159351544297` Sin[1/12 π (1.0635459627341117` + t)]
```

If you defined nlm(t) properly, then you'll generate the graph above with this command:

```
In[2210]:= Show[ListPlot[data], Plot[nlm[t], {t, 0, 24}],
             PlotLabel → "Human Temperature Across a Day", Frame → True]
```

Out[2210]=



The mean obtained by the non-linear regression model is 37.125 (in degrees Celsius).

a. What is this temperature in degrees Fahrenheit?   F=1.8*C+32,

```
In[2211]:= fTemp = 1.8 * 37.125 + 32
Out[2211]= 98.825
```

b. At what time of day is the temperature highest, according to the model?

In[2212]:= soln = Solve$\left[\frac{1}{12}\pi\,(1.0635459627341117\grave{} + t) == -\text{Pi}/2, t\right]$

```
           highTime = t /. soln[[1]]
           highTime = highTime + 24 (* Almost 5 pm *)
Out[2212]= {{t → -7.06355}}
Out[2213]= -7.06355
Out[2214]= 16.9365
```

c. At what time does the body temperature equal its mean?

```
           meanTime = -1.0635459627341117 (* plus or minus 12 hours, so around 11 am/pm *)
Out[2215]= 1.06355
```

d. How much does the temperature vary across the day?

```
In[2216]:= amplitudeC = 0.40418159351544297` (* Varies by plus-or-minus .4 degrees C *)
           amplitudeF = 1.8 * 0.40418159351544297`
           (* Varies by plus-or-minus .73 degrees F *)
           tempRangeC = 2 * amplitudeC
           tempRangeF = 2 * amplitudeF

Out[2216]= 0.404182

Out[2217]= 0.727527

Out[2218]= 0.808363

Out[2219]= 1.45505
```

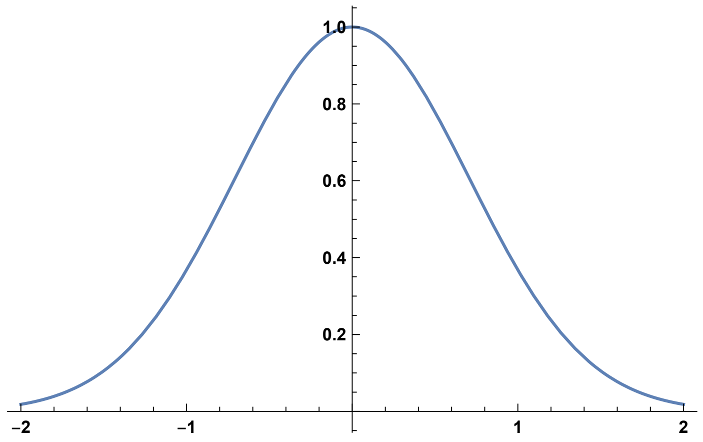
2. Use the original data and the Trapezoidal Rule, to compute the average value of temperature. How does it compare to the average obtained by the model?

```
In[2220]:= n = 23;
           lrr = 1 / n * Sum[temps[[k]], {k, 1, n}]
           rrr = 1 / n * Sum[temps[[k + 1]], {k, 1, n}]
           trap = (lrr + rrr) / 2 (* pretty close: 37.125 was the average from the model *)

Out[2221]= 37.1261

Out[2222]= 37.1304

Out[2223]= 37.1283
```

3. An example problem from section 7.7:

This function cannot be evaluated analytically by any of our methods -- the famous “bell curve”, so essential for statistics:

```
In[2224]:= f[x_] := E^(-x^2)
           a = -2.0;
           b = 2.0;
           Plot[f[x], {x, a, b}]

Out[2227]=
```



Our objective is to compute the integral $A = \int_{-1}^{1} f(x)\, dx$.

1. Compute the integral $\int_0^1 f(x)\, dx$ using Mathematica, and call this value *true* (Even though Mathematica is also doing a numerical approximation!).

```
In[2228]:= a = 0;
b = 1;
true = Integrate[f[x], {x, a, b}]
```

Out[2230]= $\frac{1}{2}\sqrt{\pi}$ Erf[1]

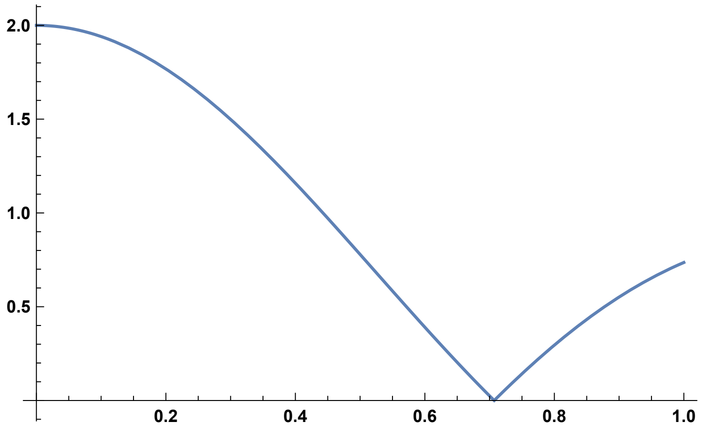
2. Why can we focus all of our energy and attention on the interval [0,1]?

Because the integral is symmetric -- there's as much area from -1 to 0 as there is from 0 to 1.

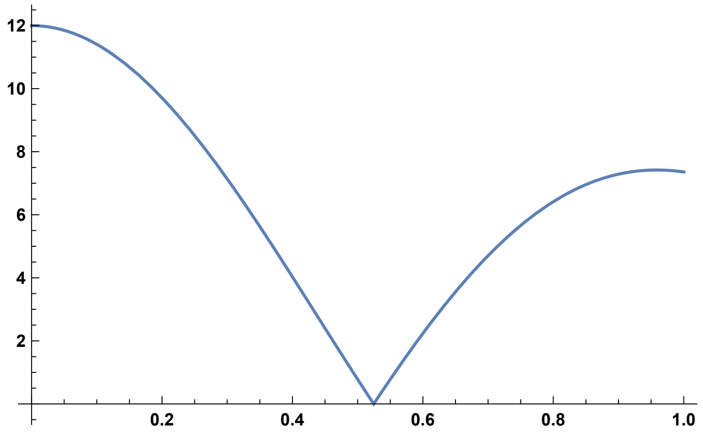
3. Graph the absolute values of the second and fourth derivatives, and obtain estimates for the error constants K for midpoint, trapezoidal and Simpson's rules.

```
In[2231]:= Plot[Abs[f''[x]], {x, a, b}]
Plot[Abs[f''''[x]], {x, a, b}]
```

Out[2231]=



Out[2232]=



```
In[2233]:= k2 = 2;
k4 = 12;
```

4. Given the following estimates for midpoint (M_8), trapezoidal (T_8), and Simpson's (S_{16}) rules.

```
In[2235]:= mid = NMidpointApprox[f[x], {x, a, b}, 8]
           trap = NTrapezoidApprox[f[x], {x, a, b}, 8]
           simp = NSimpsonApprox[f[x], {x, a, b}, 16]

Out[2235]= 0.747304

Out[2236]= 0.745866

Out[2237]= 0.746824
```

4.1. Verify that the actual absolute errors of each of the three methods satisfy the error bounds associated with the given numbers of subintervals on [0,1].

```
In[2238]:= n = 8
           k2 (b - a) ^3 / (12.0 n^2)
           Abs[trap - true]
           k2 (b - a) ^3 / (24.0 n^2)
           Abs[mid - true]
           n = 16
           k4 (b - a) ^5 / (180.0 n^4)
           simpError = Abs[simp - true]

Out[2238]= 8

Out[2239]= 0.00260417

Out[2240]= 0.000958518

Out[2241]= 0.00130208

Out[2242]= 0.000479446

Out[2243]= 16
```

Out[2244]= 1.01725×10^{-6}

Out[2245]= 1.24623×10^{-7}

4.2. Verify that Simpson’s approximation is the appropriate average of the midpoint and trapezoidal approximations.

```
In[2246]:= simp
           (trap + 2 mid) / 3

Out[2246]= 0.746824

Out[2247]= 0.746824
```

5. How do we extend both the approximations **and** the error estimates to *A*, over the entire interval [-1,1]?

We double the area, and double the expected error:

```
In[2248]:= integralGiven = 2 * simp
           2 * simpError
Out[2248]= 1.49365
Out[2249]= 2.49247 × 10^-7
```

4. Error Analysis: you compute the number of subdivisions needed.

Compute the integral $\int_0^4 x^3 \sqrt{x^2+9}\, dx$

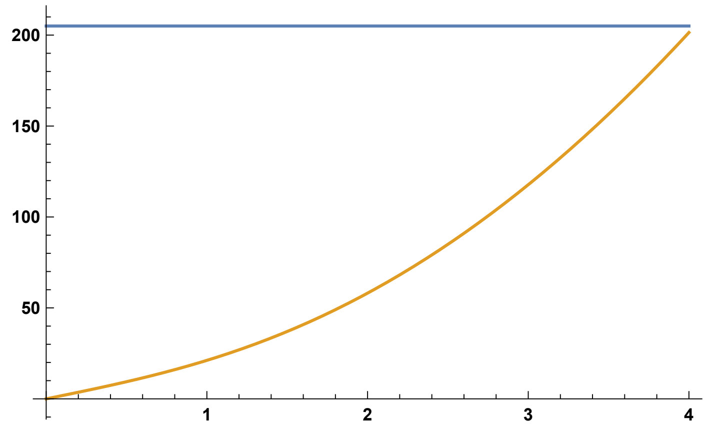
1. analytically.

```
In[2250]:= f[x_] := x^3 Sqrt[x^2 + 9]
           a = 0.0;
           b = 4.0;
           true = Integrate[f[x], {x, a, b}]
Out[2253]= 282.4
```

2. Determine what the minimal value of n should be for the midpoint rule to have accuracy within 0.0001, and compute the midpoint estimate with that value of n. Verify that the absolute error of your estimate is within the error bound.

```
In[2254]:= k2 = 205;
            Plot[{k2, Abs[f''[x]]}, {x, a, b}]
            Clear[n]
            Solve[k2 (b - a) ^3 / (24.0 n^2) == 0.0001]
            n = 2339
            mid = NMidpointApprox[f[x], {x, a, b}, n]
            Abs[mid - true]
```

Out[2255]=

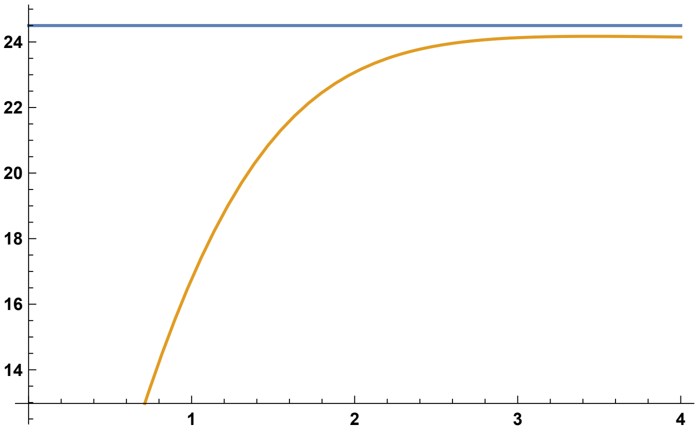


```
Out[2257]= {{n → -2338.09}, {n → 2338.09}}

Out[2258]= 2339

Out[2259]= 282.4

Out[2260]= 0.0000354846
```

3. Determine what the minimal value of n should be for Simpson’s rule to have accuracy within 0.0001, and compute the Simpson’s rule estimate with that value of n. Verify that the absolute error of your estimate is within the error bound.

```
In[2261]:= k4 = 24.5;
           Plot[{k4, Abs[f''''[x]]}, {x, a, b}]
           Clear[n]
           Solve[k4 (b - a) ^5 / (180.0 n^4) == 0.0001]
           n = 36
           simp = NSimpsonApprox[f[x], {x, a, b}, n]
           Abs[simp - true]
```

Out[2262]=



```
Out[2264]= {{n → -34.3596}, {n → 0. - 34.3596 i}, {n → 0. + 34.3596 i}, {n → 34.3596}}

Out[2265]= 36

Out[2266]= 282.4

Out[2267]= 0.0000658654
```