Riemann Sums

One of the most important and useful concepts of calculus is the idea of computing accumulation. In many situations, there is information about a rate of change (e.g., acceleration, velocity, concentration, flow, power) and calculus presents tools for calculating the accumulation that occurs (e.g., velocity, position, amount, volume, energy, etc.).

Table of Contents

[Calculating Solar Energy Generation](#MW_H_25AC406F)   
 [Background](#MW_H_BF44E87F)   
 [Select a day](#MW_H_AA5911A6)   
 [Continuity of Datasets](#MW_H_DEECCC87)   
[Application: Riemann Sums with Rectangular Approximations](#MW_H_2B736D71)   
 [Visualizing Error in Riemann Sum Approximations](#MW_H_DA43D88F)   
[Riemann Sum with Trapezoidal Approximation](#MW_H_20526209)   
 [Application: Solar Energy Data](#MW_H_3D8ED0F6)   
[Helper Functions](#MW_H_5546E545)

# Calculating Solar Energy Generation

## Background

Consider the power generated by a fixed solar array with a maximum rated output of 643.545 kW. To convert this power (in kilowatts) into energy (kilowatt-hours) requires accumulating this power over time.



Data from the MathWorks Lakeside Campus parking garage solar array, March 14-20, 2021.

### Select a day

Choose a day to analyze more carefully.

Focus on the period when it wasn't dark, so the panels had a chance of generating electricity.

## Continuity of Datasets

This graph looks continuous, but in fact it is linearly interpolated between each adjacent datapoint, generated by sensor reports every minute across the day. Consider the scatterplot representation:

 **Reflect**

1. How easy is it to fit a simple curve to this data? Does it matter which day of the week is chosen?
2. Are most real-world datasets continuous or discrete? Why? What implications does this have for using calculus in applications?
3. Is the underlying data in this situation, where solar panels are generating power, continuous or discrete?

# Application: Riemann Sums with Rectangular Approximations

This data provides sensor readings at one minute time slices across the entire [day](#MW_H_C15C875A). To calculate an approximation of the amount of energy generated, it is necessary to add up an approximation of the power generated in each time slice by the length of the time slice. Visually, if we select a left-endpoint approximation:

Notice that many of these rectangles are overestimating (or underestimating) the value of the energy generated over that slice of time.

 **Try**. By default, this plot is generated with a step size of 1 hour (60 minutes) and uses the left-endpoint to approximate the power generated over the entire interval. Try selecting both the right-endpoint approximation and the midpoint approximation as well as shortening the step size.

 **Reflect**.

1. Which method (left-endpoint, right-endpoint, or midpoint) generates the most accurate approximation to the total energy generated?
2. What impact does shortening the step size have on the total energy estimate? Does it matter whether it is using a left-endpoint, right-endpoint, or midpoint approximation?

### Visualizing Error in Riemann Sum Approximations

Visualizing the error at each step requires tracking how much over-estimation and under-estimation contributes to each estimate. The red area visualizes the overestimation of energy production, while green area visualizes the underestimation of energy production.

 **Try**.

1. Change the [method of approximation](#MW_H_5A646D9D) to right endpoint, midpoint, maximum value, and minimum value.
2. Change the [day of the week](#MW_H_C15C875A) that you are studying.
3. Use the [slider](#MW_M_7C08D964) or watch the animation to observe the impact of increasing/decreasing the step size.

 **Reflect**.

1. What do you notice about the error for the different approximation methods?
2. What impact does shortening the step size have on the total energy estimate? Does it matter which type of estimate you are using? Why or why not?

# Riemann Sum with Trapezoidal Approximation

Constant value approximations are frequently clearly bad approximations, even to the naked eye. Continuing the theme of approximating functions by polynomials, after a constant (0th degree polynomial) the 1st degree polynomial approximation is a line. Using the endpoints of each interval this gives us a trapezoidal approximation of the area under the curve.

## Application: Solar Energy Data

Compare using trapezoidal approximation on the solar energy data with different step sizes to the best possible estimate given the resolution of the data.

# 

# Helper Functions

If you wish to see the details of the code, select the **View** tab and switch to **Output Inline**. Alternately, select **Output Inline** using the icon  at the top right of the Live Editor pane.

The function chooseDay returns the chosen timeslice of the overall data file.

The function viewTrapBetween sets up the trapezoidal approximation graphs.

The function viewRiemannBetween plots the signed between fdata and the line y=height.

The function displayTrapezoidMethod computes the approximate area and displays a plot of the trapezoidal approximation

The function drawColoredPolys determines the regions where the approximation is too large or too small and colors them accordingly.