**Summary:**

The final project is a GUI that can be used to track the calories burnt by a person while walking/running. It is called Workout Tracker. The interactive user interface accepts and accounts for the height and weight of the user. Accelerometer data is collected from the user’s mobile device and used to calculate the velocity of the individual and thus the calories burned per minute.

**GUI Design:**

The Android Support Package was used to stream accelerometer data recorded using MATLAB Mobile from an Android mobile device to MATLAB R2018b on a PC on the same network.

Creating an object of **mobiledev** allows us to use a host of supporting functions that allow for streaming of data in real-time. This enhances the usability of the application as we do not need to generate, move and then import .csv/.mat files retrospectively. Establishing the data stream allows for instantaneous results from data gathered in real time. This makes the GUI user friendly and more applicable to real-life scenarios.

The acceleration data (along with the timestamps) collected, processed and converted to the velocity of the individual by integrating. An efficient way to do this is by using the **cumtrapz()** function. This method quickly returns the integral of a function by performing numerical integration via the trapezoidal method. This method approximates the integration over an interval by breaking the area down into trapezoids with more easily computable areas (see figure 1).

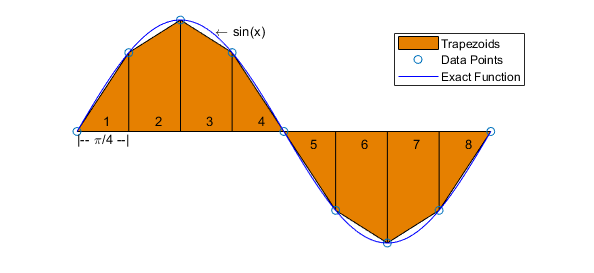


Figure . Integration of sine curve using trapezoidal method

On the GUI a plot of calories burned per minute is displayed (see figure 2). This plot is continuously refreshed and adapts according to incoming acceleration data. This is achieved using a **recursive function**. A recursive function is a function that calls itself thus creating a loop. The loop is exited only when the user presses the STOP button on the GUI, thus maintaining user-control.

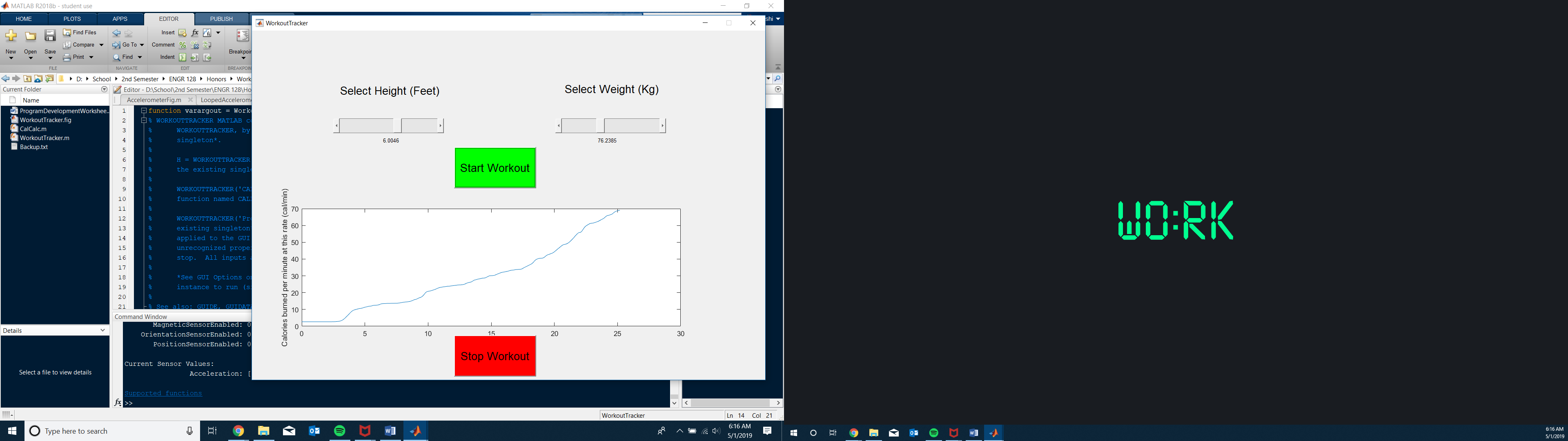


Figure . The GUI that the user interacts with

**Sliders** were created in GUIDE to collect the height and weight values of the individual which are also required to accurately calculate the calories the user is burning.

The values set using the sliders were shared among the callback functions of the GUI by utilizing the methods **setappdata()** and **getappdat()**. These can be used together to effectively move data to and from callback functions defined in a GUI. The function setappdata() is used to store data collected in a variable that is created for an object that is initialized (syntax: setappdata(object\_name, variable\_name, value.).

This data can then be retrieved from within a different callback function using getappdata() ( syntax: value = getappdata(object\_name, variable\_name) ).

This is a quick and easy way of sharing data between callback functions defined in GUI and has the added benefit of allowing sharing of data not only between different callbacks but also between separate GUIs.

This sharing of height and weight values collected at the slider callbacks has multiple benefits. It allows us to display in a static text box the actual value to which each slider has been dragged. Otherwise users would be blindly dragging the sliders, not knowing the value being inputted, drastically increasing error. Most importantly, it allows the START button callback to retrieve the height and weight values and pass them as arguments when calling CalCalc(height, weight) (the subroutine which actually calculates and plots calorie data).

**Design conclusions / Possible improvements:**

The design meets the requirements of providing the user with accurate information that expresses calories burnt in real time using accelerometer data while accounting for the heigh and weight of the individual.

However, there are improvements that can be made to improve the product in future iterations:

* Instead of using the exit command to interrupt the recursive loop, a flag can be passed using the setappdata() and getappdata() functions, thus performing a cleaner termination of the GUI.
* The GUI could allow the user to input a “target value” and then let the user compare their performance to an ideal situation (calculated using “target value”) by plotting both on the graph.
* Develop the application for other common platforms other than MATLAB in order to boost popularity and cut costs.