Morteza Kazemi | 97243054

Amir Masoud Shaker | 97243081

12/12/2021

Homework #7

# Q1

**Dual-tone multi-frequency signaling (DTMF)** is a signaling system used in telecommunication which uses voice-frequency band over telephone lines between communication devices and switching centers. it uses **eight audio frequencies** in pairs of two to represent 16 signals (0 to 9, A to D, # and \*). These signals do not need operators on long-distance circuits because they are audible tones in voice-frequency range and can be transmitted through electrical repeaters and amplifiers, and over radio and microwave links.

**Multi-frequency signaling (MF)** is using a mix of two pure tone (pure sine wave) sounds for signaling. One of the MF signaling protocols is for in-band signaling between switching centers, where long-distance telephone operators use it to route the next destination telephone number. Based on the success with using MF to establish long-distance telephone calls, dual-tone multi-frequency signaling was developed for end-user signaling without the need of operators.

**In order to prevent interfering** consumer telephones and routing between telephone switching centers, DTMF frequencies differ from MF signaling protocols between switching centers. Also as DTMF signaling is often transmitted in-band with audio signals, it has limits for timing, frequency deviations, etc.

**The DTMF** **telephone keypad** is shown in the right. Rows and columns show the frequency components. As an illustrative example, pressing 1, produces a mix of a 697 Hz and a 1209 Hz tone that is decoded by the switching center to determine the keys pressed by the user.

Symbols #,\*,A, B, C and D were added to the keypad to access computers and automated response systems. However, in the end, A to D keys were dropped from most phones, they could be used to assert certain privilege and priority for calls, control network, select menu items, etc.

The telephone keypad system replaced rotary dial(dial pulse) with the help of DTMF. It was also used in caller ID systems to transfer caller information, in cassettes for information encoding and in television broadcasting and controlling remote transmitters.

The code for the question is in the following page:



# Q2

In the Piano\_simulation.m file, we simulated two octaves of piano (starting at middle C).

It has both white and black keys (which means all the notes in each octave) and they are placed similar to real piano.

First we create an audiorecorder object which helps us to record the sound of piano and we record for 20 seconds. We set a name for the audio file that we’re going to make and using audiowrite, we write the audio file in the directory that we are in it.

Then we start our most important part of code.

We create an empty figure which responds to the keyboard.

The write the function key, that uses a switch case structure in it to determine the right frequency for each key.

We start at C4 and end at C6# which contains a little bit more than two octaves.

We have a formula for computing the frequency for each node and that is:

freq = 2 ^ ((n-49)/12) \* 440

In the frequency\_values.m file, we compute frequencies for all the notes between C4 and C6# just to make sure that the values in our key function our correct.

Finally, we create the wave for each note and pass it to sound function which converts the signal data to sound.

Three audio files named audio1, audio2, audio3 are created.

Resources:

[Piano - File Exchange - MATLAB Central (mathworks.com)](https://de.mathworks.com/matlabcentral/fileexchange/71836-piano/)

[Piano key frequencies - Wikipedia](https://en.wikipedia.org/wiki/Piano_key_frequencies)

[Record and Play Audio - MATLAB & Simulink - MathWorks Deutschland](https://de.mathworks.com/help/matlab/import_export/record-and-play-audio.html)

Piano\_simulation: 

Frequency\_values:



# Q3

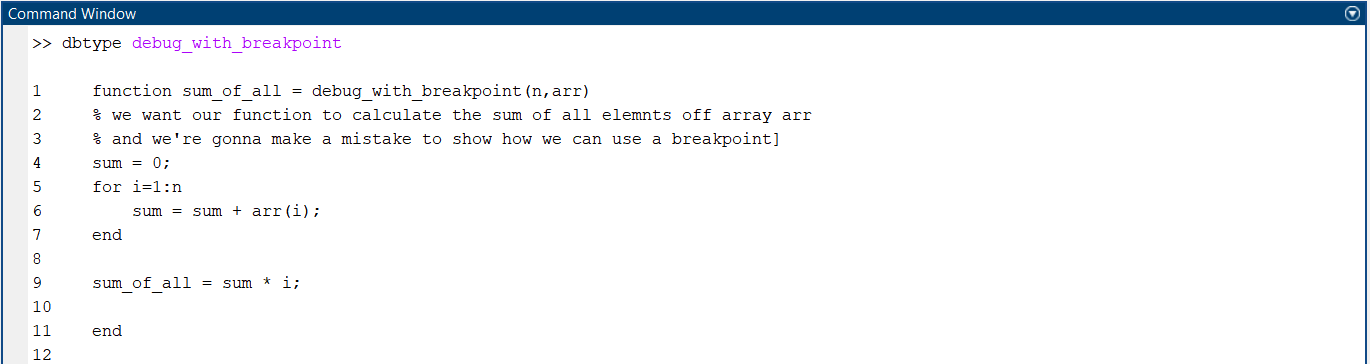
How to debug an m-file with breakpoints:

First we should know that the debugging commands work on both functions and scripts.

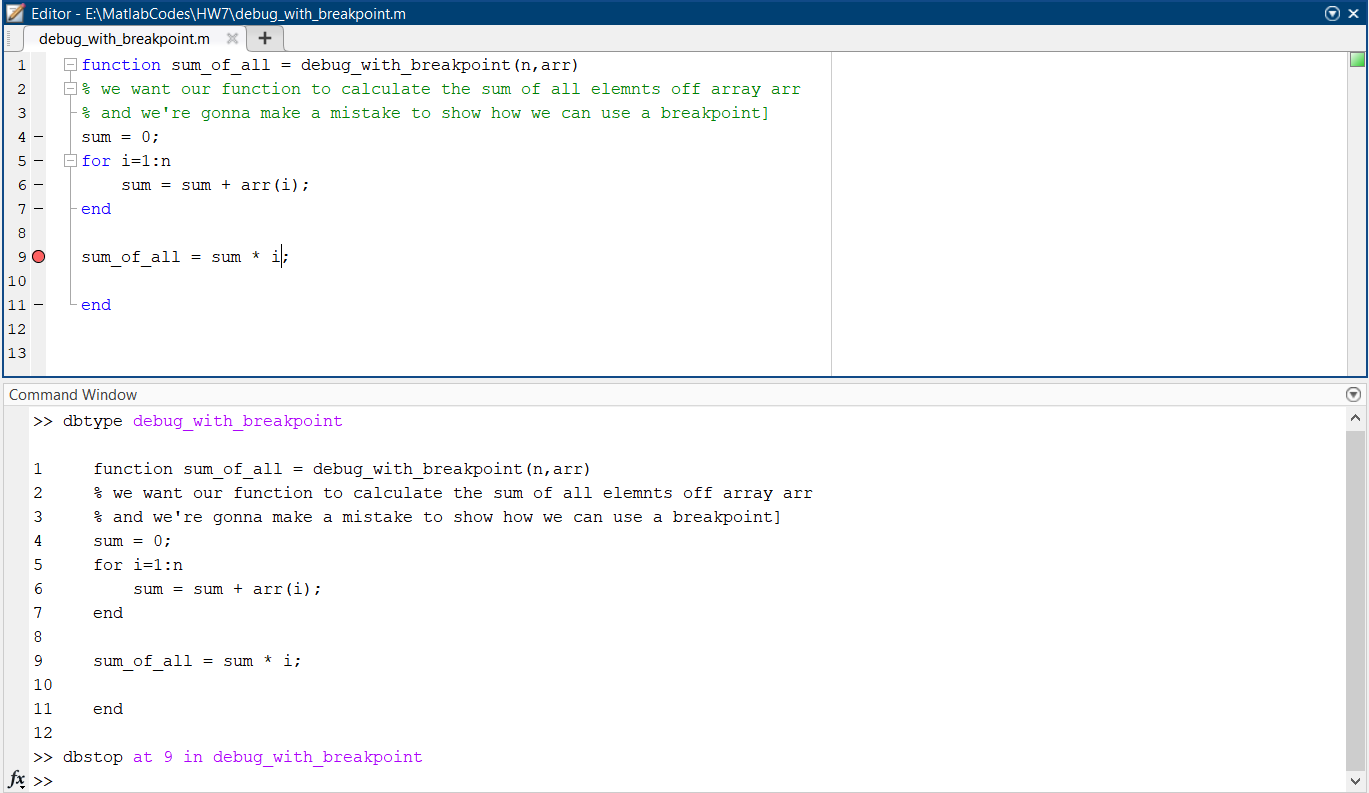
Here is our function that we want to debug :

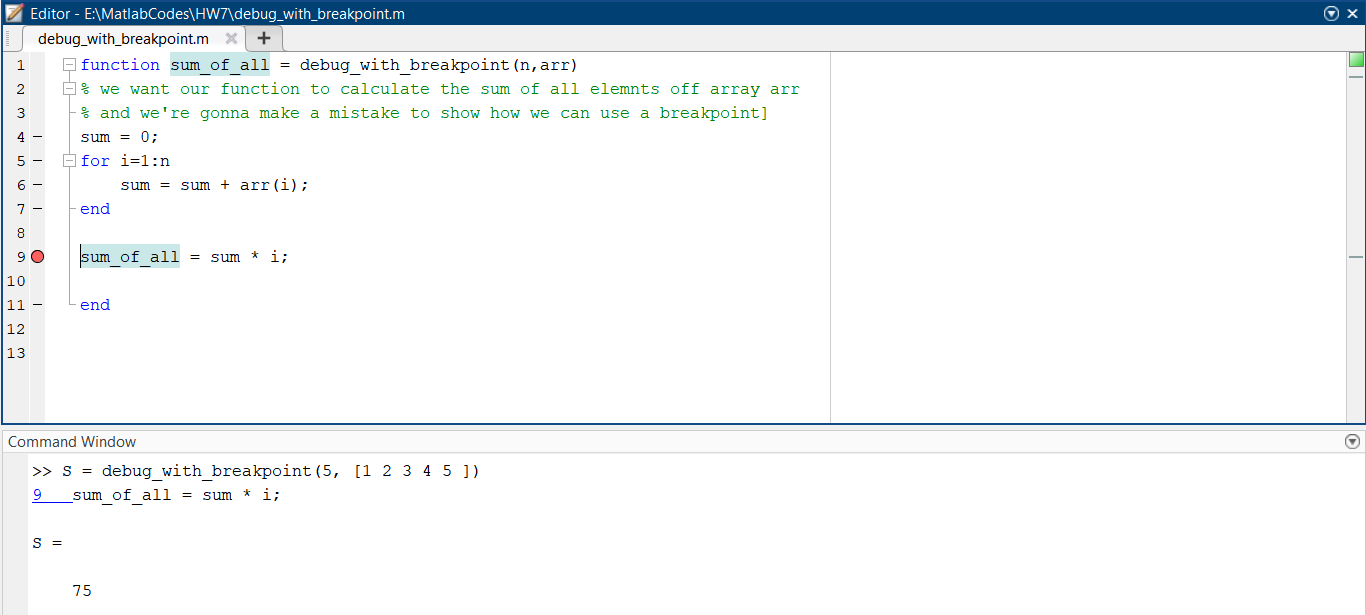


First we can use **dbtype** command to list the line numbers.



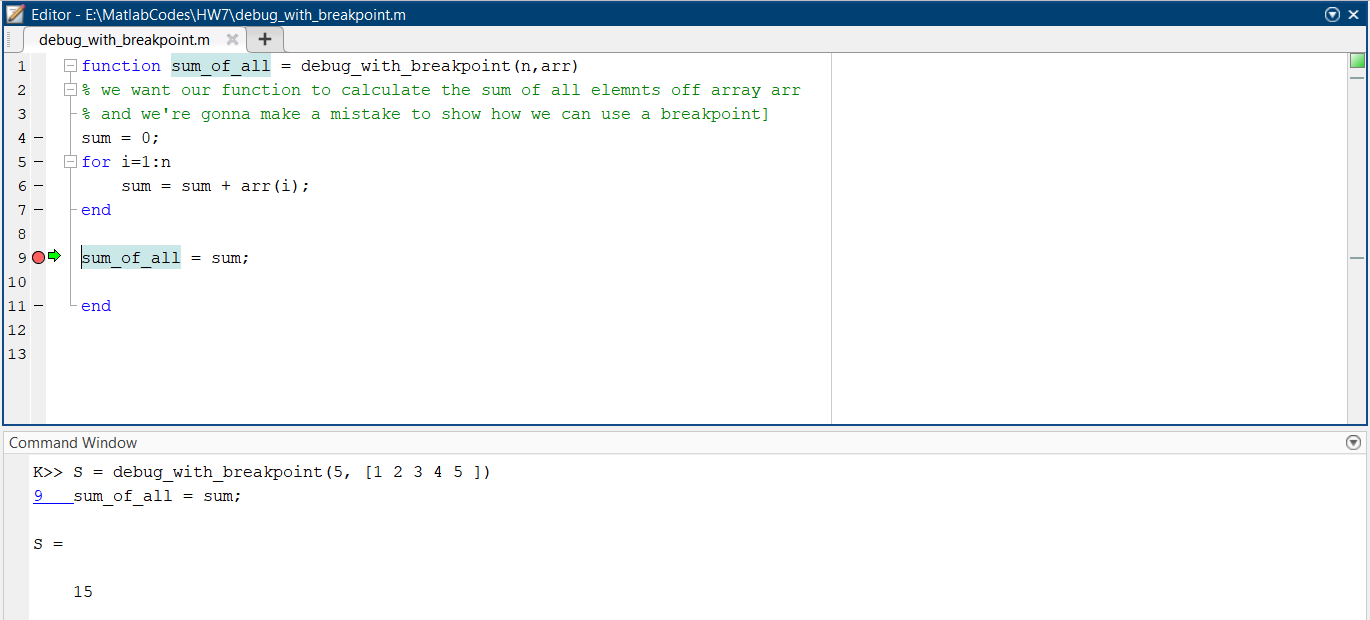
Then we set our breakpoint in line that we want using **dbstop** command:





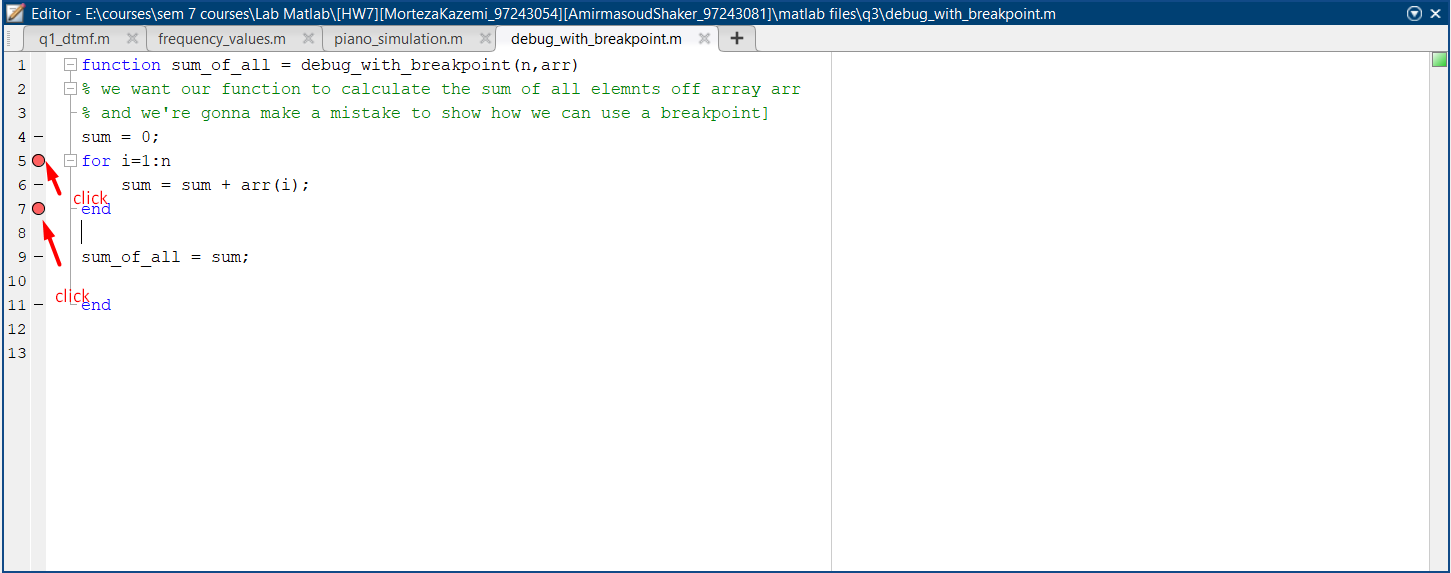
Now we can see that the multiply by i is additional and should be removed.

So we fix the error easily:

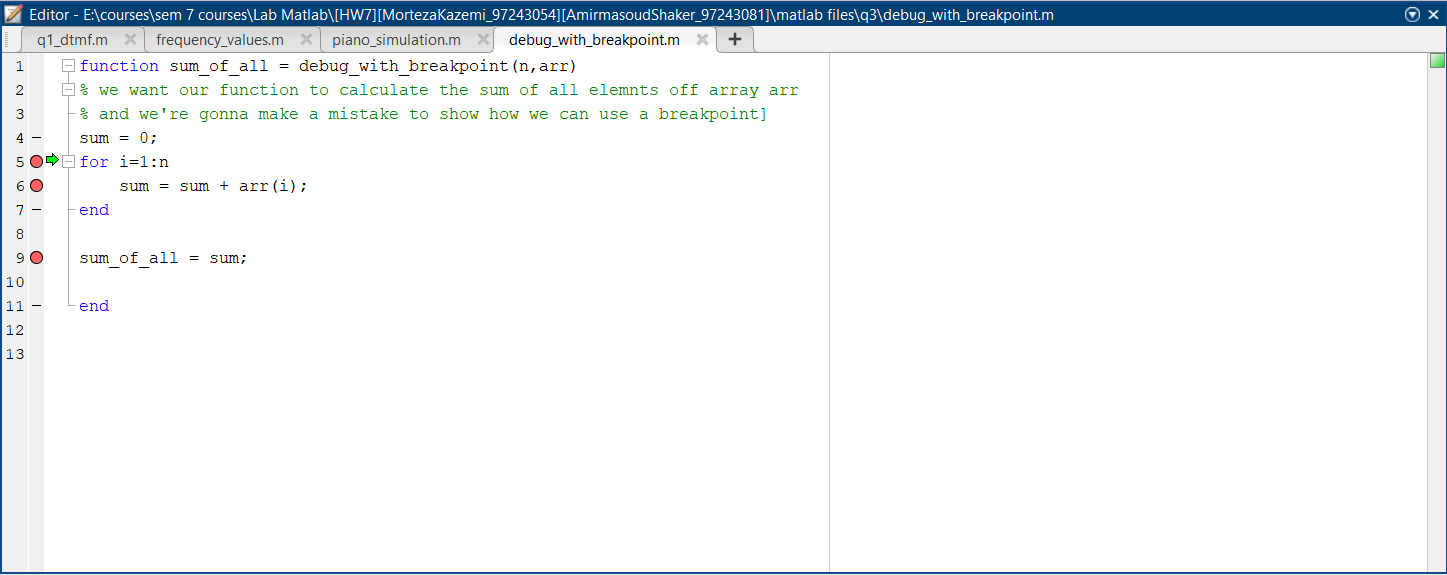


We can also use the Matlab GUI to debug. This is an easier way of debugging.

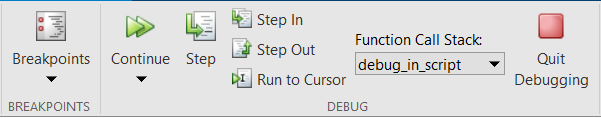
First, we can put the breakpoints by clicking on the left side of the editor pane. We can have multiple breakpoints in one file.



Then we run the program and see that it stops at a certain line of the code shown by a green arrow.



When in debug mode, we can see the following tab in the Matlab toolbar:



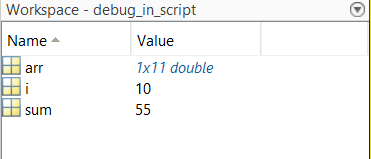
By clicking on Continue, we run the program to the next breakpoint.

By clicking on Step, the program executes one line of the code.

By clicking step in we can see through the functions we call and by clicking step out we can get out of the current scope in which we are.

Run to cursor could be an option to move faster. We need to click on a line of our code so that the cursor would be set there. Then, we can click on this option to execute the program upto that line of the code.

On the right side of the Matlab window, we can see the workspace pane:



We can see our variables values by double clicking on it or writing its name in the command window(or even writing more complex scripts to show a desired dimension of our variables).