

# Approximate Find

---

A Tax Associated Best Friend

Jason Whiting

4/13/2011

## Executive summary of the project

Approximate Find is an Excel Add-in that allows for specialized numerical searches. I originally conceived approximate find when trying to help a friend, a fellow PwC associate, work through cooperate and individual tax workbooks. You can search for specific values in Excel but this only allows for string comparison searches. Many times when working with different accounts and ledgers you want to find a number near a certain value. You may also come across a line item that is the aggregate of other cells and want to find which cells add up to that amount. Approximate Find allows you to give a numeric value, different tolerances (percentage, fixed range), and the number of numeric values you would like to find that add up to your target value; these features greatly extending the search capabilities of Excel.

## Implementation documentation

In explaining my solution I would like to detail four main components.

• Add-in	• GUI	• Data Gathering	• Data Searching
----------	-------	------------------	------------------

Each of this components

### Add-in

“Approximate Find” or “Tax Assist” are two names I have given to the add-in that I have created for Excel. Approximate find is stored as an Excel add-in file or .xlam. To use it you must go to File->Options->Add-ins->Go and browse to the .xlam file. Moving this file after adding will break the functionality so it is important to store it somewhere permanent (e.g. not your desktop). Once loaded there are two ways to access the Approximate Find GUI and therefore it’s functionality.

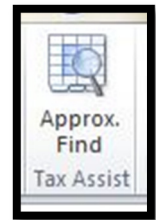


Figure 1 - Ribbon Icon

1. **Ctrl + G** - Pressing this key combination will immediately bring up the Approximate find GUI. So instead of Ctrl + F, just go over one to ‘G’ and you can do a numerical approximate find.
2. **Ribbon Icon** - The Ribbon Icon is placed under the Home tab so that it can be accessed without fumbling around between tabs.

### GUI

Once loaded, Approximate Find will present you with a GUI (Graphical Unser Interface) that will allow you to search. In the GUI you will find a box in the top left corner to insert you search term. You are then allowed to choose your search range details. You can either have a percentage or absolute search range. For example you could search for any number within 3% of your search term or 1,000 above and below you search term. You can also narrow your search scope between the current page and all worksheets. Finally, you can choose if you want to find a composition of 1, 2, or 3 numbers that add up to a value within your specified range.

In displaying your results a list box is populated on the right side. You will find first, the value located in the cell, the sheet it is on, and its cell address. For each grouping you will see either a '|' or '---'. These will alternate every other result unless there are groupings of 2 or 3. In that case, the cells are groups by either symbol.

The screenshot shows an Excel spreadsheet with a grid of values. A yellow box labeled "Sheet 3" is in the top left. The values in the grid range from 10200.00 to 76200.00. The "Approximate Search" dialog box is open, showing search criteria and results.

**Approximate Search Dialog Box:**

- Search for:** 27,000.00
- Search Range Detail:**
  - Min: 26,000.00
  - Max: 28,000.00
  - as a %: 3.00 %
  - amount: 1,000.00
- Worksheets to Search:**
  - ☒ All
  - ☐ Current Only
- Search Multiple Cell Sums:**
  - ☐ Single Cells Only
  - ☒ 2 Cells
  - ☐ 3 Cells
- Search Results:**

Value	Sheet	Cell	Sum
10000	Test Sheet 1	B6	26000
16000	Test Sheet 1	H6	26000
--- 10000	Test Sheet 1	B6	26050
--- 16050	OtherTestSheet	G7	26050
10000	Test Sheet 1	B6	27050
17050	OtherTestSheet	H7	27050
--- 10000	Test Sheet 1	B6	26200
--- 16200	Final Tests	H4	26200
11000	Test Sheet 1	C6	26000
15000	Test Sheet 1	G6	26000
--- 11000	Test Sheet 1	C6	27000
--- 16000	Test Sheet 1	H6	27000
11000	Test Sheet 1	C6	26050
15050	OtherTestSheet	F7	26050
--- 11000	Test Sheet 1	C6	27050
--- 16050	OtherTestSheet	G7	27050
11000	Test Sheet 1	C6	26200
15200	Final Tests	G4	26200
--- 11000	Test Sheet 1	C6	27200
--- 16200	Final Tests	H4	27200
12000	Test Sheet 1	D6	26000
14000	Test Sheet 1	F6	26000
--- 12000	Test Sheet 1	D6	27000
--- 15000	Test Sheet 1	G6	27000
12000	Test Sheet 1	D6	28000
16000	Test Sheet 1	H6	28000

Buttons: OK, Done. Time: 0.844 seconds.

## Data Gathering and Data Searching Methods

There are a total of six methods, one module, and one form that operate the primary functionality of the Approximate Find tool.

### Code / Approximate\_Find()

The code module is very basic. It is the module that contains that macro that initiates the GUI form. The add-in contains its own hidden worksheets and also its own hidden macros. When the Approximate Find add-in is not set to "isAddin" from the developer window then these worksheets and relevant options are exposed in the main Excel window. In the more complicated search methods I use these hidden worksheets to organize and parse through the data. Also, it is here that I assign Ctrl + G to the

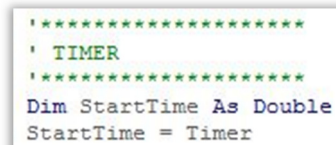
Approximate\_Find() sub-procedure (macro). By default these worksheets, more accurately, this workbook is hidden.

## Search Form

The SearchForm is where the GUI and all of its elements are created and laid out. By going to the code view while looking at this form you will find all the methods that run this tool. When looking at the GUI interface the most critical and complicated piece is the list box on the right. Search results are displayed in the list box as separate list items. Each list item comprises a row in the list with five possible columns. The best feature of the box is that on clicking a list item it has Excel make the current active cell the address listed in the list item. This features allows you to quickly browse to the results in the context of the workbook.

## Do\_Search

The “Do\_Search” method is what is launched when the “OK” button is pressed. In this method everything is prepared for the searching process. It first sets up a timer so that the time it takes to process the search can be displayed. This method also reads all the values submitted in the GUI form. Based on the user’s inputs one of the following three methods are called: “appx\_find,” “appx\_find\_2\_cells,” or “appx\_find\_3\_cells.”



```
*****  
' TIMER  
*****  
Dim StartTime As Double  
StartTime = Timer
```

## Appx\_find

“Appx\_find,” this is the meat of the approximate find tool. Though it isn’t the most complicated of the methods, it provides the central functionality of the add-in. “Appx\_find” first collects all the numerical values spread throughout either the current worksheet or all worksheets depending on the user’s specifications. Using the range object method “specialCells” I was able to pull out all constants and formula values from the worksheets without having to parse through each cell. The constants and formula results are stored in two separate ranges. I then check to see if either is empty. If so, I just use the one with values to do my searching. If both have data I perform a union on the two data sets and work from there.

The next part of the code is to perform either a search with a percentage or absolute range. It runs a “for each” loop in either case. The logic it checks the current value to see if it is within the search range. If it is, a new item is added to the list box in the GUI specifying the value, the worksheet, and cell address; if no results are found it will populate the list box with the appropriate notification.

## Appx\_find\_2\_cells and Appx\_find\_3\_cells

The “appx\_find2\_cells” and “appx\_find\_3\_cells” were probably the most fun to program. They are very similar to the “appx\_find” method; however, they have nested loops and optimization for finding multiple cells that add up to the desired range. The most interesting part here is that I work with hidden worksheets belonging to the add-in so that I can have the powerful information navigation features of a worksheet, yet work transparently to the user. For each of these I copy all the information out of the range object into a worksheet called “data.” I could not find a way to easily parse through the range in memory. With the 2 cell search, for each value in the list I run another loop to add it to all the following

values in the list. Having the information in a worksheet gave me the simple data addressing functionality you would find in an array that I could not find in a range object. In the 3 cell search, it is basically the same thing; however, there is third loop to check all following values with the first value pairing. I was able to increase performance significantly by cutting out the third loop if the first to numbers were already larger than the specified range.

## Discussion of learning and conceptual difficulties encountered

### Gathering the Data

The first thing that I tackled with my project was how I was going to search all the data. I knew I needed a way to gather all the numerical data, whether explicit or derived (e.g. a formula) and aggregate it in some form that was easily parsed for searching. Some of my first experimentation, with my basic knowledge of VBA, was of course to read cell by cell on each worksheet and evaluate whether or not the value was numeric. This method, as expected, was very inefficient. The culmination of my efforts is the few lines of code you see in figure 2. The key is that you can assign an entire worksheet to a range and then use the SpecialCells method to pull out the numeric values for constant and formula cells. Once the constant and formula cells are gathered into a range a union can be ran on the two ranges forming one single range with all numeric data.

```
Set rAcells = WS.Range("A1").EntireColumn.EntireRow
Set rCcells = Nothing
Set rFcells = Nothing

'In case of no formula or constants.
On Error Resume Next

Set rCcells = rAcells.SpecialCells(xlCellTypeConstants, 1)
Set rFcells = rAcells.SpecialCells(xlCellTypeFormulas, 1)

If rCcells Is Nothing And rFcells Is Nothing Then
    'no numbers at all in the worksheet
    Exit For
ElseIf rCcells Is Nothing Then
    'only formulas
    Set rAcells = rFcells
ElseIf rFcells Is Nothing Then
    'only constants
    Set rAcells = rCcells
Else
    'formulas and constants
    Set rAcells = Application.Union(rFcells, rCcells) 'Both
End If
```

Figure 2 - Numeric Data Gathering Algorithm

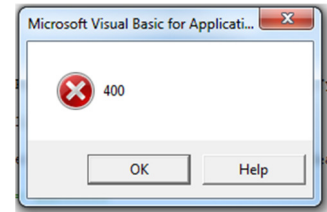
### Creating a GUI

Creating the GUI was an interesting challenge. For the most part it went pretty smoothly. I was able to figure out man what I needed with little effort. Making Excel move the active cell to the selected item

from the list box was the most useful. Speaking of the list box, it was quite possibly the most annoying thing to work with. My number one frustration was that I wanted to color different row backgrounds so that I could show groupings. Unfortunately, as far as I could find there is no way to set the background of an individual item in a list box—super annoying.

## The Button that Pushes my Buttons

This is still my biggest gripe. I have approached the issue from many angles and I still can't remove the error. I have created a ribbon icon to launch my application. However, each time I launch it, I get this "400" error. I have found the error means that Excel is trying show a form that is already visible. I have done some testing and for some reason when clicking the button it makes Excel try to launch its referenced method more than one time causing this error. Running the method it references separately does not generate the same error. In theory, pressing "CTRL + G" is doing the exact same thing.



**Update:** I finally solved the problem finding a working xml file and tweaking it. I found the special sauce. I need to have "(control As IRibbonControl)" in my macro's parameters.

## Wrap-up

This project has been a really fun exploration of Excel for me. I started maybe five or six different side projects before resting on this one. I really like making tools that solve an immediate problem. Additionally I was able to learn two things that I wanted to know better—how to create an add-in and how to customize the ribbon.