

MATH 246, WORKSHEET 3: INTRODUCTION TO PROGRAMMING

1. TOPICS

This continues the last exercise on worksheet 2, introducing all the most basic programming concepts except loops (repetition).

- Interactive input and output in M-file programs.
- Programming style: improved user interface and documentation within M-files.
- Decision making: the `if`, `else` and `elseif` commands, arithmetic comparisons and logical statements.
- Repetition (looping): the `while` command.
- MATLAB functions (as opposed to M-file scripts.)

2. EXERCISES

As you develop your program, save major new versions under new names `quadroots1`, `quadroots2`, ... and then submit them all through the file server (or by email), along with diaries (command `diary`) and at least one graph.

Start with your basic M-file script `quadroots` for solving a quadratic equation using the quadratic formula, and successively refine it as follows.

- (1) Write a new more *flexible* script, `quadroots2`, allowing the user to solve different quadratics without needing to edit the file each time. Do this by using interactive input of the quadratic's coefficients, using the MATLAB command `input`].
- (2) Write a new more *user friendly* script `quadroots3`, that prints messages to explain what it does, what the input variables mean, and what the output values signify. Use commands like `disp`, or explore the more sophisticated `fprintf`.
- (3) Write a new more *readable and understandable* version `quadroots4`: the same user interface as version 3, but with the file itself more readable by another person (or by you if you reread it some months later). Add *comments* in the script, explaining what the script does, how it does it, what the variables mean and so on. Also try to choose intuitive, "self-documenting" names for variables and such.
- (4) Add *help documentation* to version `quadroots5` the script, so that the command `help quadroots` does something useful. (You might already have done this in the previous steps without realizing!)
- (5) Write a new more *robust* version `quadroots6`: test it for hard cases (a single root, complex roots, $a = 0$) and make sure that it either gives correct answers or a message explaining why it did not give an answer.
- (6) Write version `quadroots7` of the script, that can solve a succession of quadratics, going on until the user says to stop.
- (7) Convert the script to a Matlab *function*, `quadsolve` so that it is used like this

```
>> [root1, root2] = quadsolve(2, -8, 6)
```