PROGRAMMING HANDOUT

ECO 613/614

FALL 2007

KAREN A. KOPECKY

Towards good programming

Here is a list of things you should always do when writing computer programs. This is especially true for any programs you write for this class.

- Programs should be well-commented.
- Variables should be given intuitive meaningful names.
- Programs should be neat and readable. Develop a good programming style and use it consistently.
- It is always best to start out with the simplest possible case and add on layers of complexity one by one.
- Don't wait until you've typed all your code up to run it for the first time. As you develop a program you should test it often. This makes it much easier to spot the source of errors as they arise.
- Your program should periodically print out some information about how much progress it's made. This is especially important for time-intensive programs and programs that are designed to run until some condition (unrelated to the total running-time) is met.

Also note that in practice the main reason that we solve problems numerically is because we can't solve them analytically. Hence in reality we don't know their exact solutions and therefore may not be sure whether or not our numerical solutions are close to their analytical counterparts. In order to gain confidence that the output of our programs are close to the true solution we should check for signs of different sources of error. Here are some checks that one should do in order to ensure that the solution he obtains is accurate/correct:

- Check equations or relationships that we know from the theory should hold true but were not used in the numerical procedure. These relations should also hold numerically and the amount by which they don't hold should decrease as the accuracy of the numerical solution is increased. For example, the competitive equilibrium of a model economy always contains one redundant relation. This relation can be checked to ensure that the solution we have found is in fact the competitive equilibrium of the economy.
- At some point, as the level of accuracy we impose on the procedure is increased the solution should not change much. If there are relatively large changes in the solution and/or these changes don't get smaller as the accuracy level is increased there may potentially be a

conditioning or stability problem.

- When writing iterative programs that require an initial guess, try different initial guesses. Some algorithms are sensitive to the initial guess and for some guesses will not converge. But if you know that your problem has a unique solution, if the algorithm does converge it should always converge to the same result.
- Make small changes in parameter values and observe the impact on the solution found. If the such small changes lead to big changes in the solution you may have conditioning or stability problems.