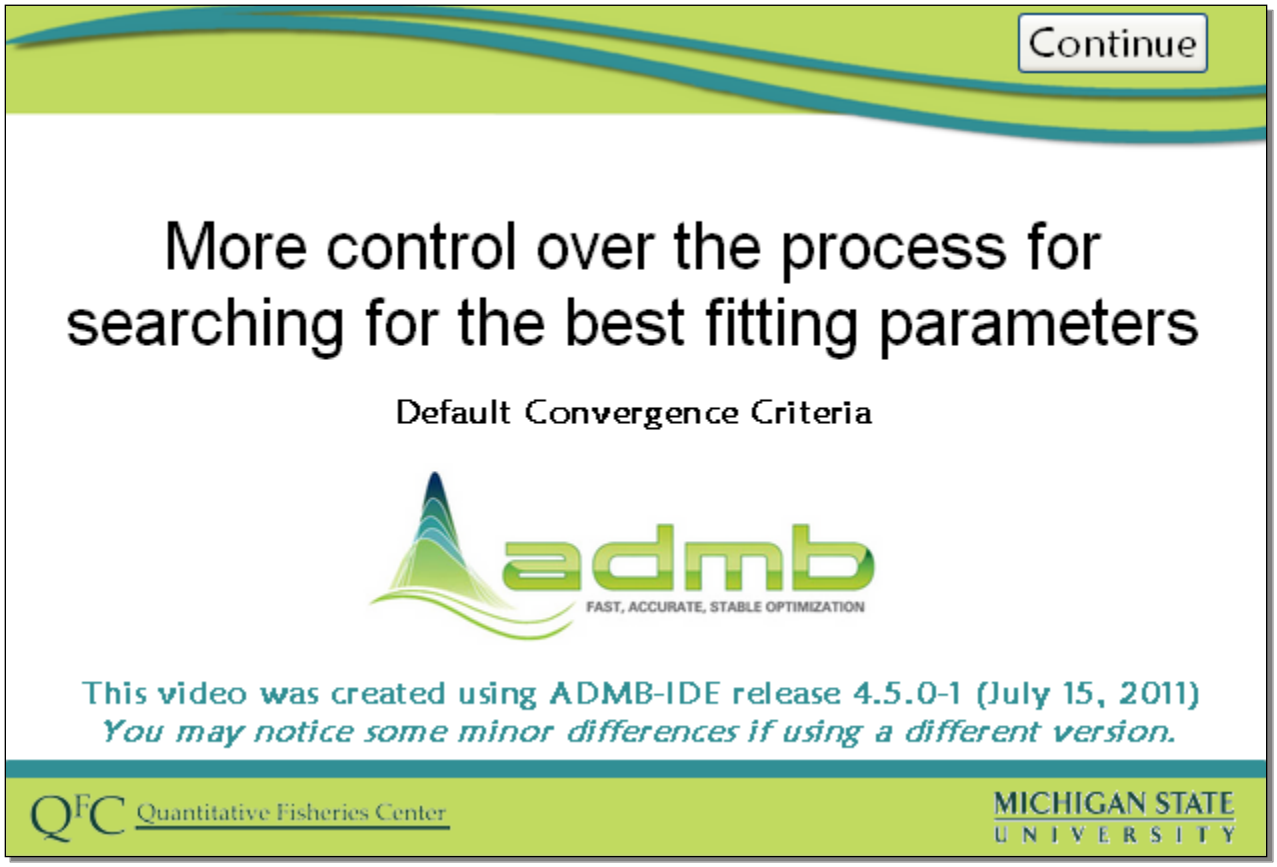
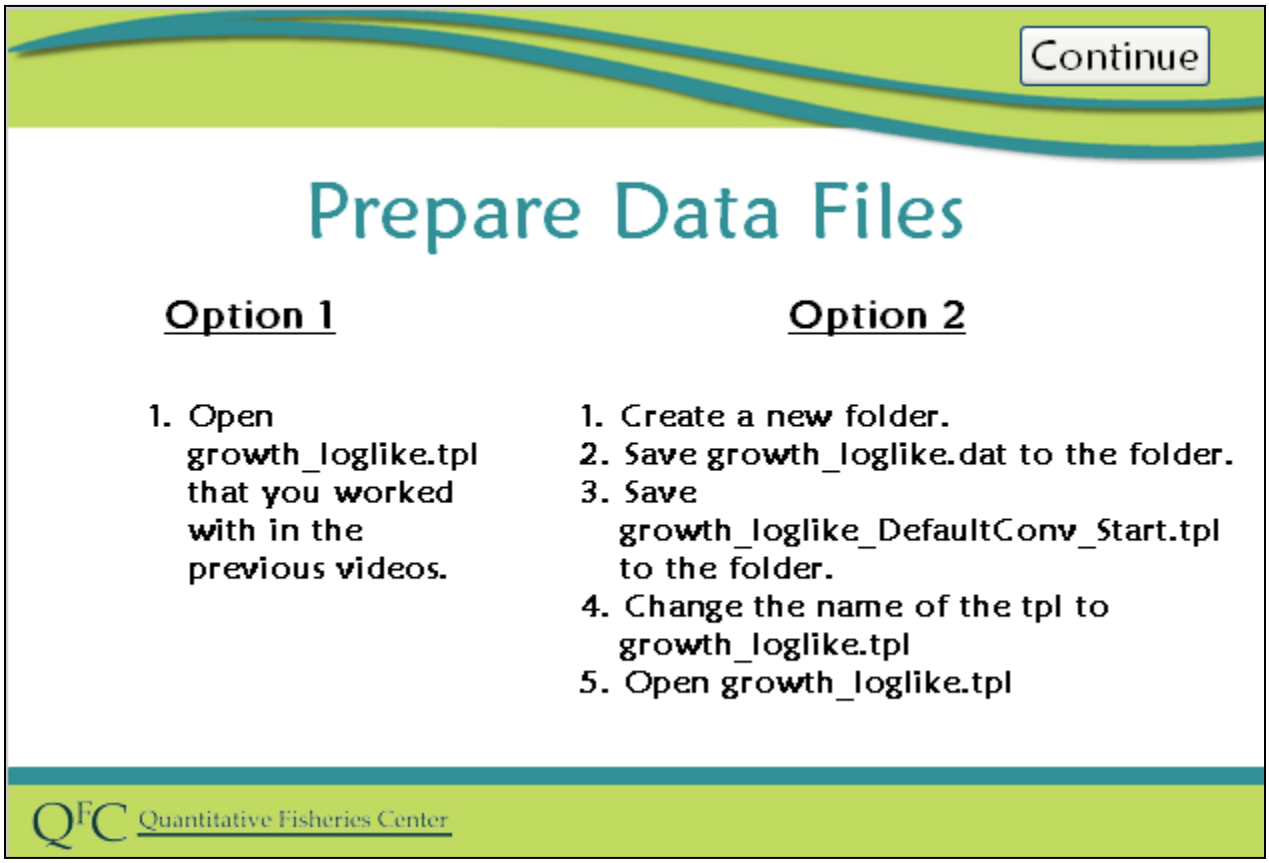
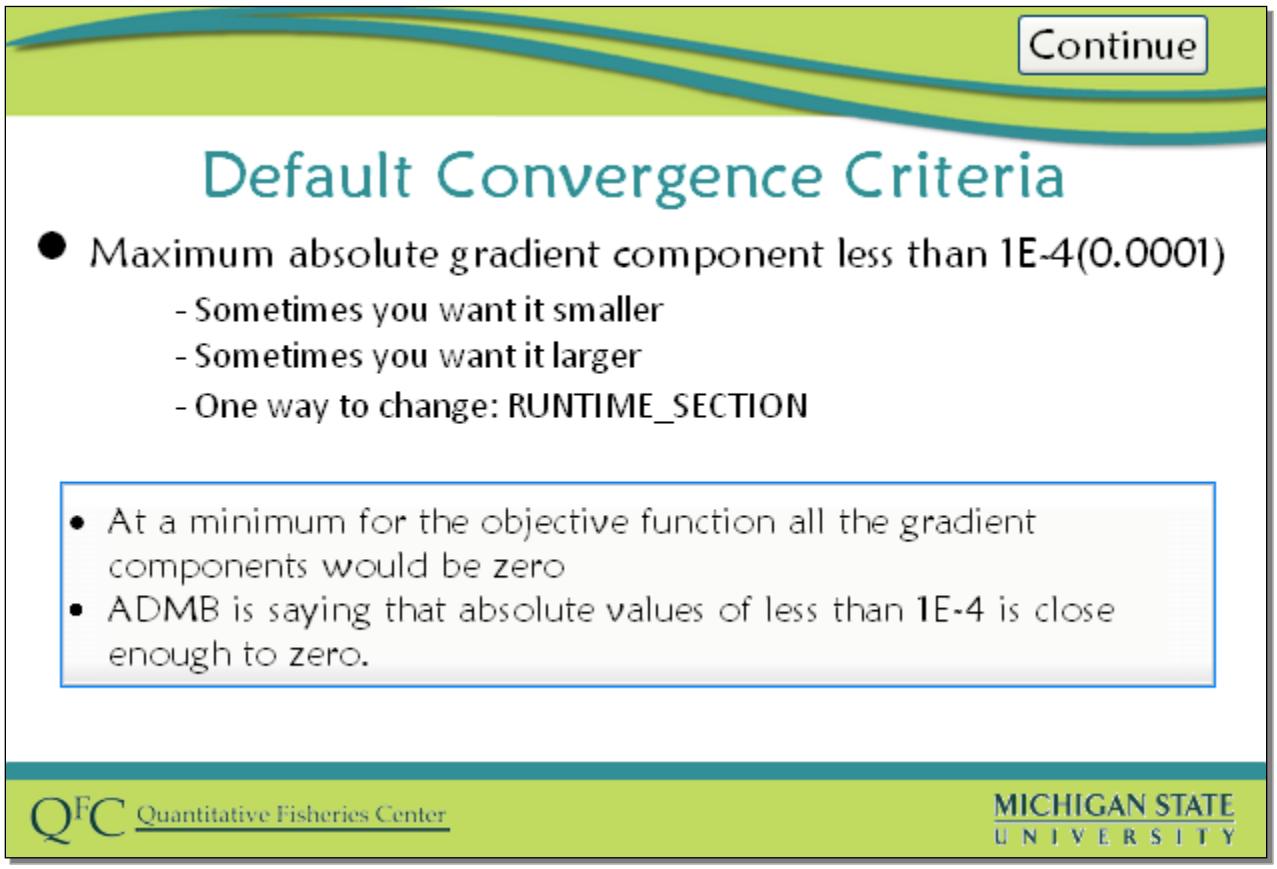


Up to now we have used the default convergence rules. However you can alter some key aspects of this. The convergence criteria specify how ADMB should decide if it has found an acceptable solution and stop searching for a better solution.

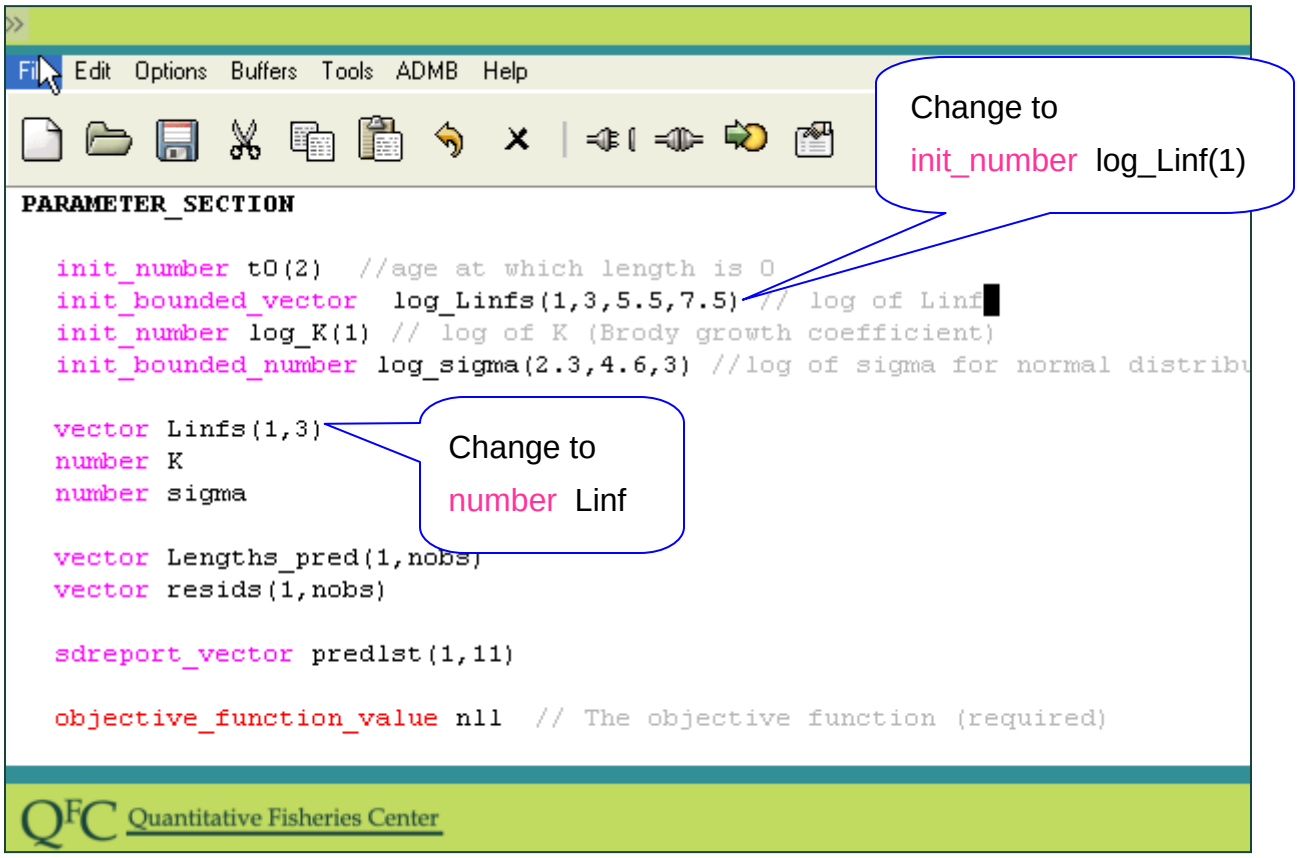


Prepare and open your files. Click continue when you are ready.

Slide Action:
Open growth_loglike.tpl that you worked with in the previous videos. Or
1. **create a new folder.**
2. **Save growth_loglike.dat to the folder.**
3. **Save growth_loglike_DefaultConv_Start.tpl to the folder.**
4. **Change the name of the tpl to growth_loglike.tpl**
5. **Open growth_loglike.tpl**



The gradient is the vector of partial derivatives of the objective function with respect to the parameters. Thus there is one gradient component for each parameter. By default ADMB stops when the maximum gradient component is less than 1 times 10 to the minus 4. At a minimum for the objective function all the gradient components would be zero and ADMB is saying that absolute values of less than one times ten to the minus 4 is close enough. This default maximum gradient component criterion generally works well but not for all problems. If you try different starting values for parameters and end up converging to different answers, and if the time to convergence is not too high, then you might want to make the convergence criterion stricter. I.e., you would require a smaller absolute value. Alternatively, if your program takes a very long time to converge and you need to refit it many times you may want to experiment and see if a less strict convergence criterion works. That is you get nearly identical estimates in less time when using it. One way to change the convergence criterion is by specifying it in your tpl in runtime section. Let us add a runtime section and change the default convergence criterion now.

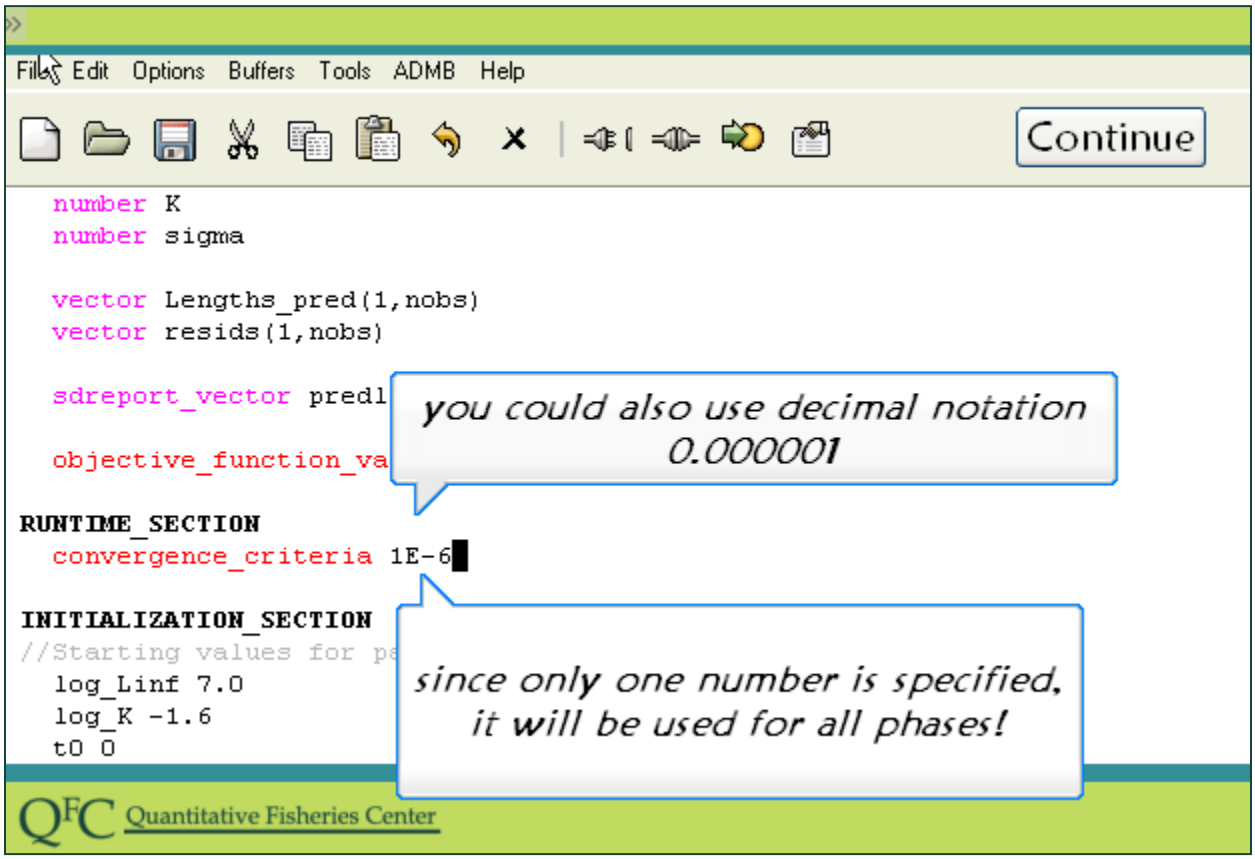


The first thing we need to do is to return log_Linfs and Linfs back to unbounded numbers. Make changes as shown above.

Slide Code:

Change vector Linfs (1,3) **to** number Linf

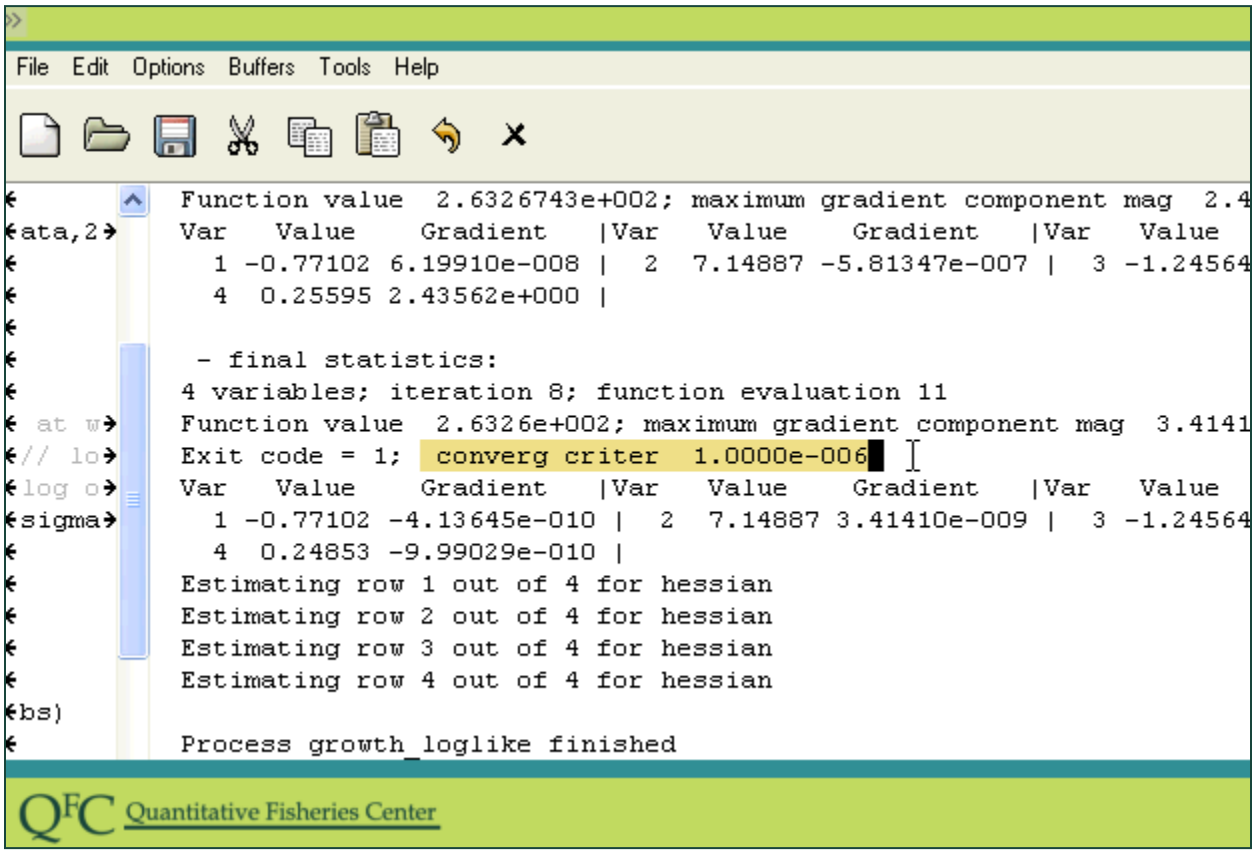
Change init_bounded_vector log Linfs (1,3,5.5,7.5) **to** init_number log_Linf(1)



We will put this (Runtime) section right below the parameter section. It cannot be defined before the parameter section but it could be put after other later sections. We will set the maximum gradient component criterion to 1E-6.

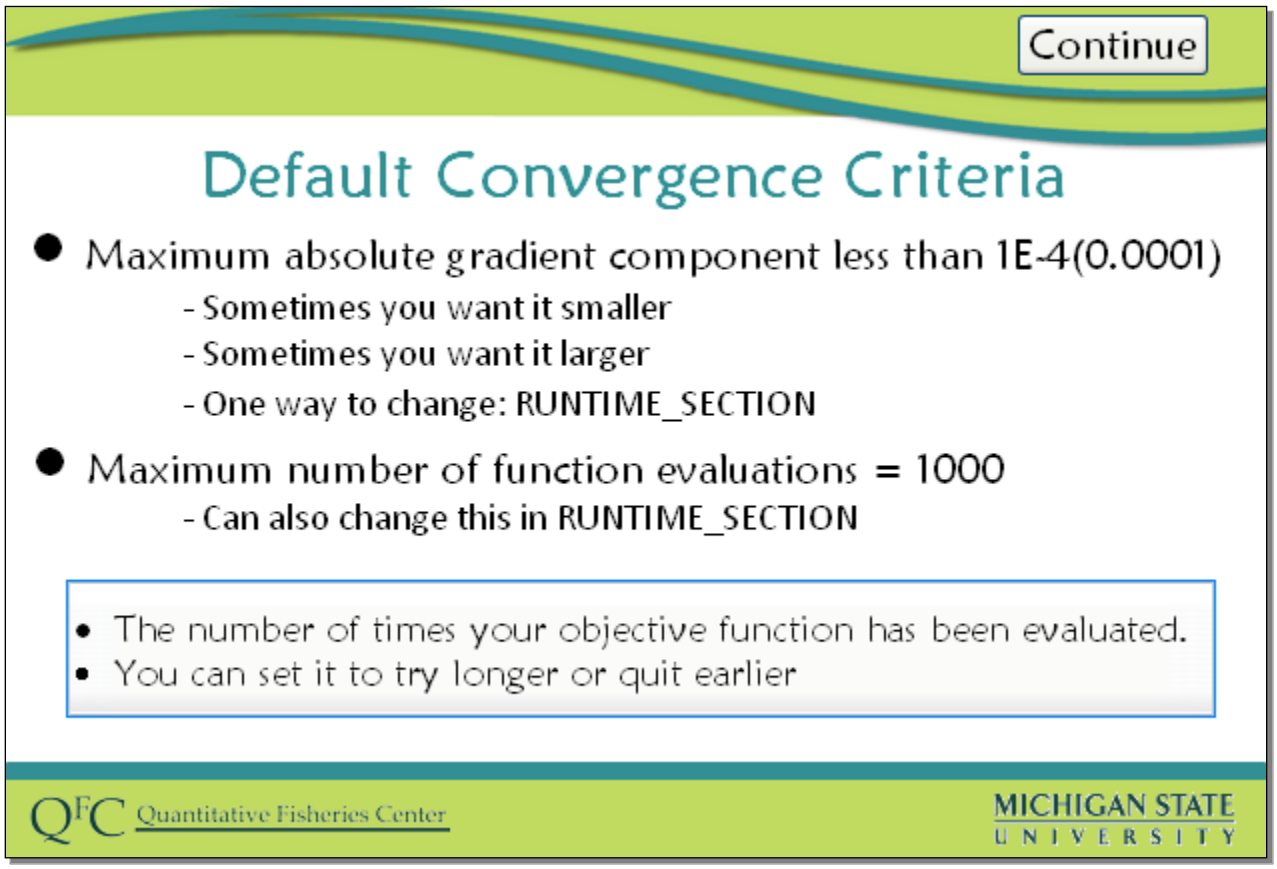
Slide Code:

```
RUNTIME_SECTION
    convergence_criteria  1E-6
```

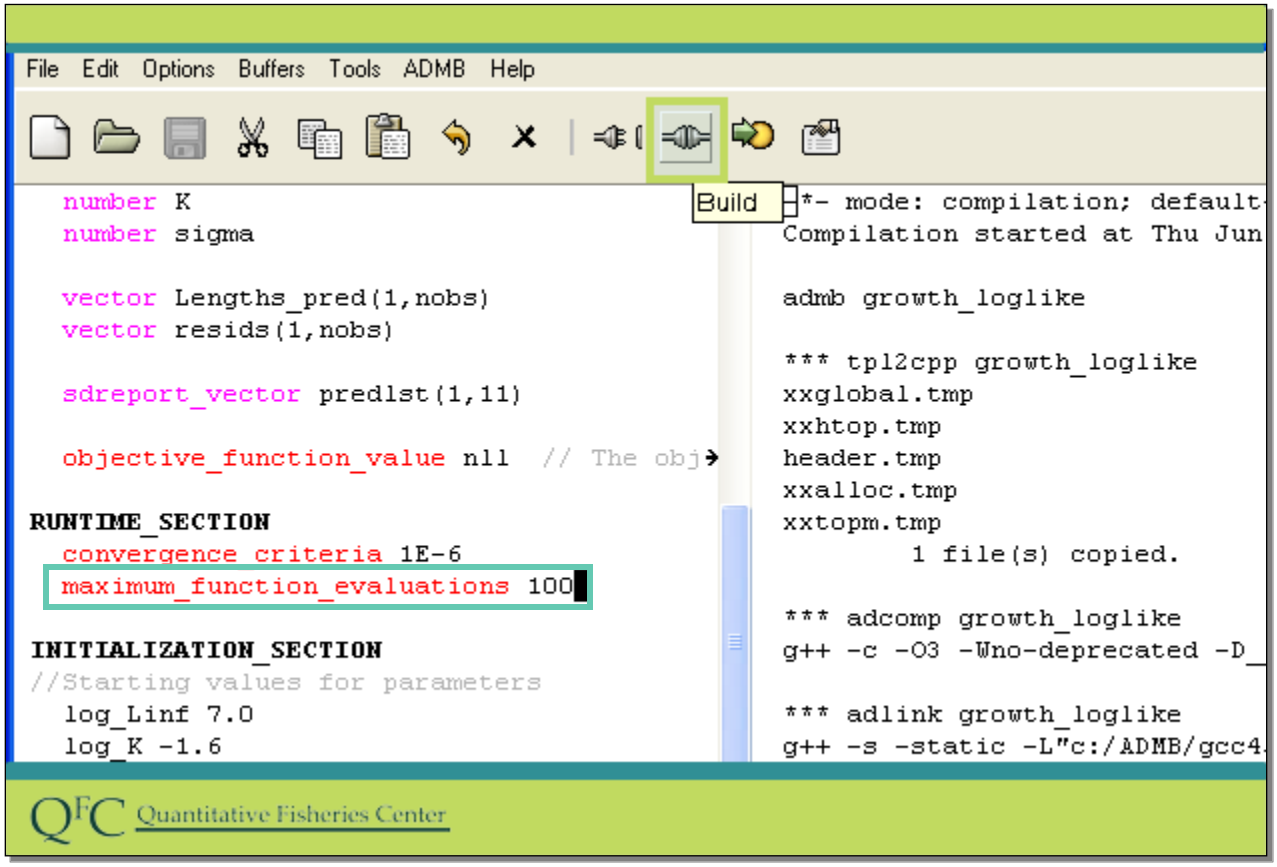


1. We now build
2. and run the program

Notice in the output that the convergence criteria is now given as 1E-6. If you compared the parameter estimates obtained with this criterion with those from the default you would see they are identical to 8 significant digits. So clearly in this case the default criterion was strict enough.



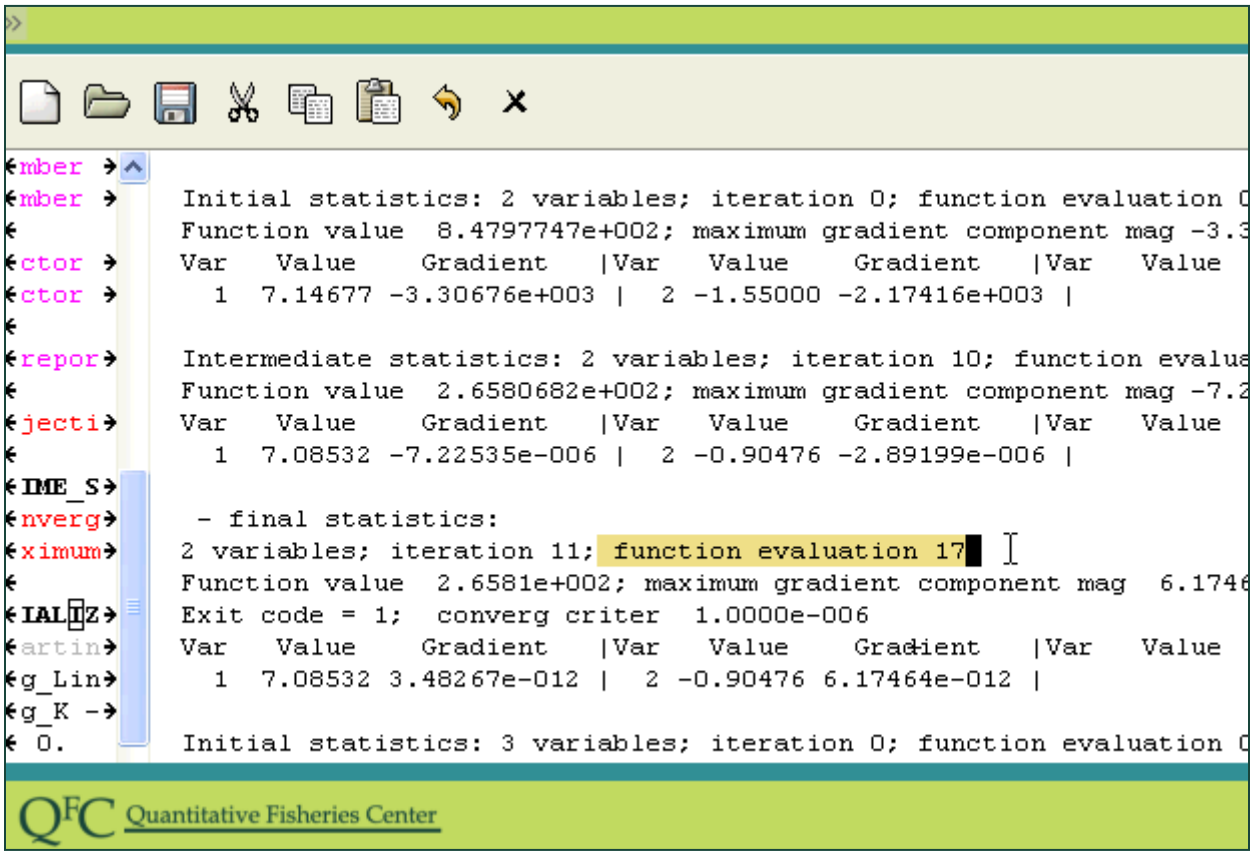
Another reason an ADMB application might stop even when the criterion for derivatives has not been met is that it has evaluated the objective function too many times. ADMB refers to the number of times your objective function has been calculated during the search as the number of function evaluations. The default is 1000 and ADMB essentially assumes that enough time has been spent and the derivative criterion will never be reached. You can ask for ADMB to either try longer or quit earlier by changing the maximum number of function evaluations. This is dead easy. We just need to add a line to our runtime section.



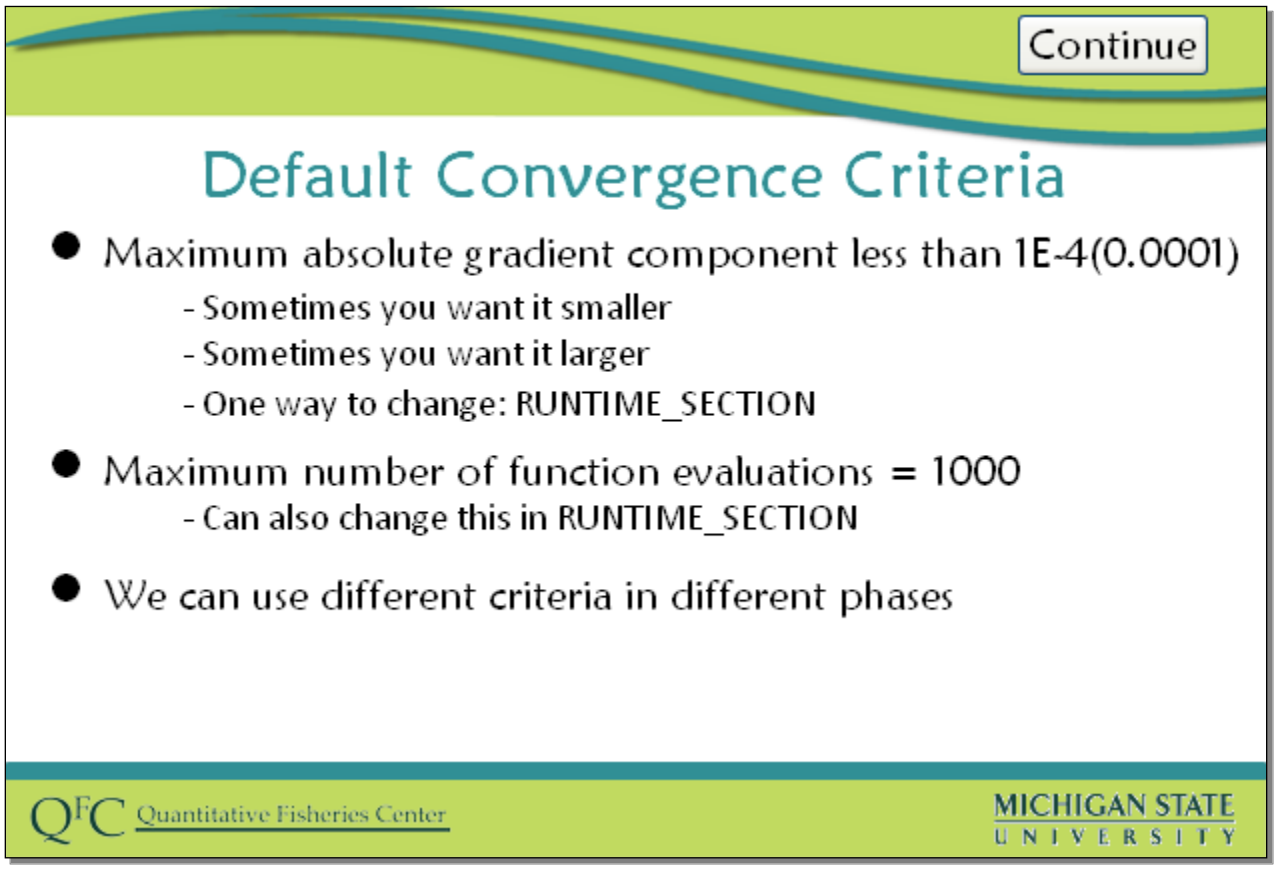
1. Here we will set the maximum function evaluation to 100.
2. Again we build
3. and run the program

Slide Code:

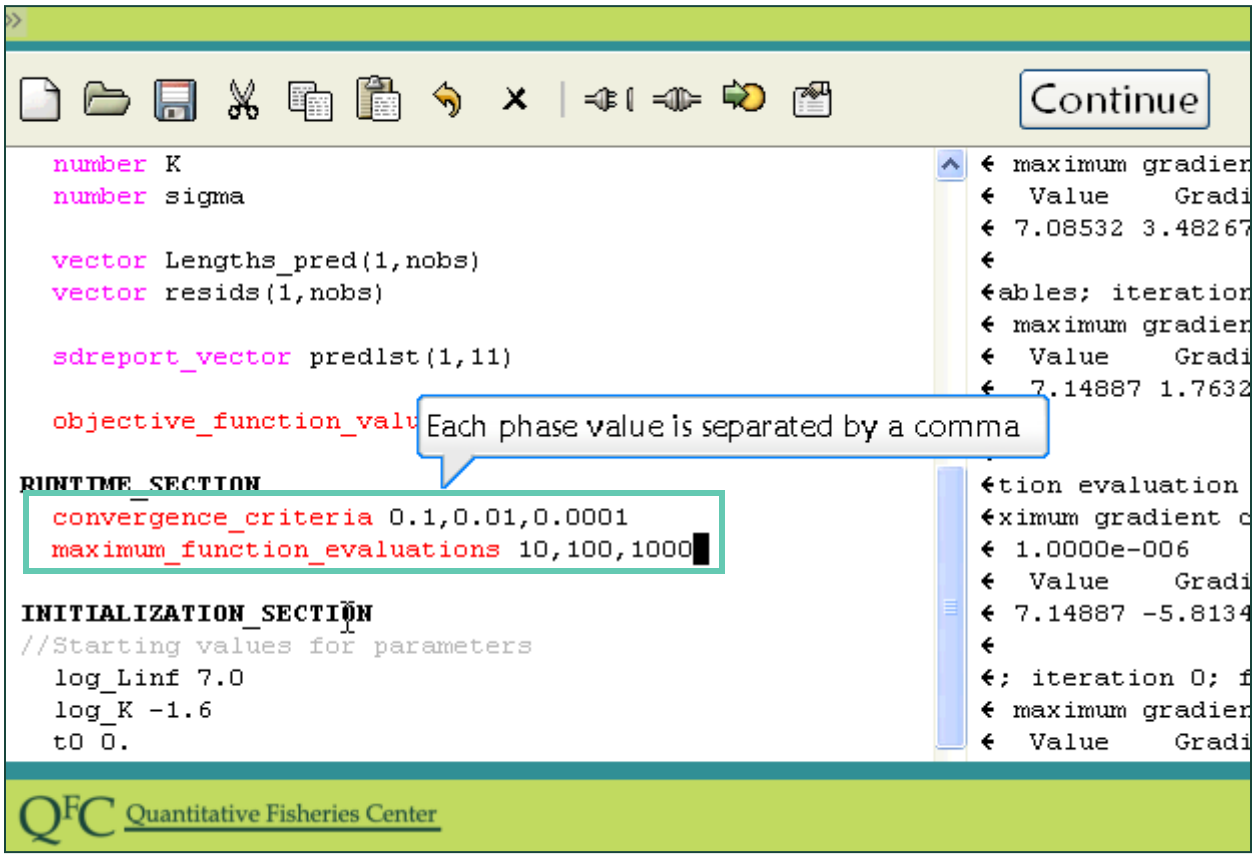
```
maximum_function_evaluations  100
```



If we scan through the output window notice that the actual number of function calls for phase 1 was 17, for phase 2 was 14, and for phase 3 was 11. So using this lower value had no effect in this case. In more complex problems you might know that the time it would take to do 1000 function calls is excessive and that if convergence takes more than say 200 calls there is likely a problem with how you set up the problem. In such cases you would want to change the default number of function calls.



Up to now we have used the same criterion for the gradient and the for the number of function evaluation for each phase. However we only need to be rigorous about getting to the absolute best estimates for the last phase. Remember the earlier phases are just a fancy way to get reasonable starting values for the last phase. Sometimes you can save lots of time by using more relaxed criteria for the earlier stages.



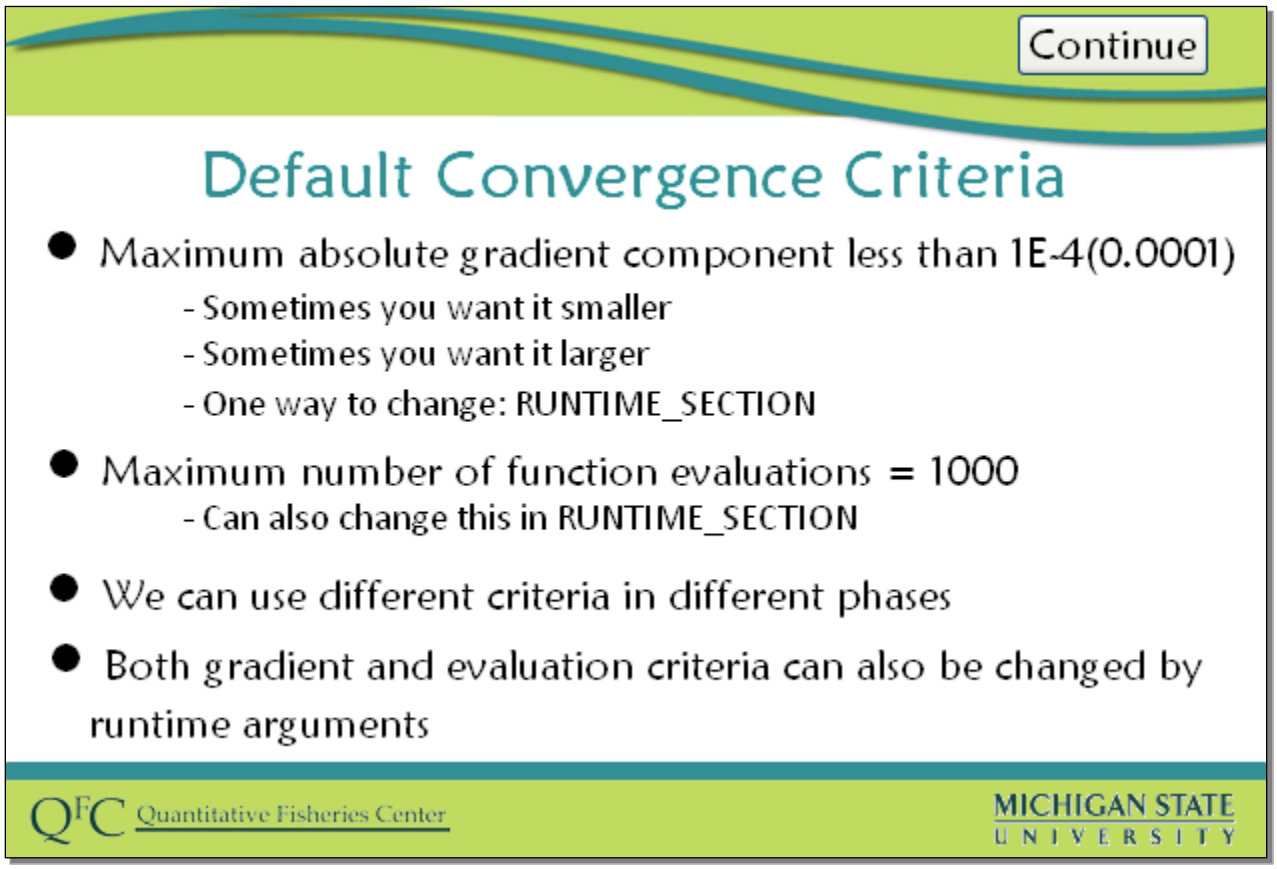
Let's modify our code by specifying criteria for each phase. Notice we just list the criteria for each phase, both for convergence criteria (0.1, 0.01, 0.0001) and number of function evaluations (10, 100, 1000) as numbers separated by commas on the appropriate line.

This code uses a gradient criterion of 0.1 in the first phase and will stop after 10 function calls. In the second phase it requires 0.01 for the maximum absolute value of the derivatives and no more than 100 function calls. And so on. Note that if there are more phases than values specified the last specified value will be used in later phases. This trick of using less strict criteria for earlier phases can sometimes speed up fitting substantially with little cost.

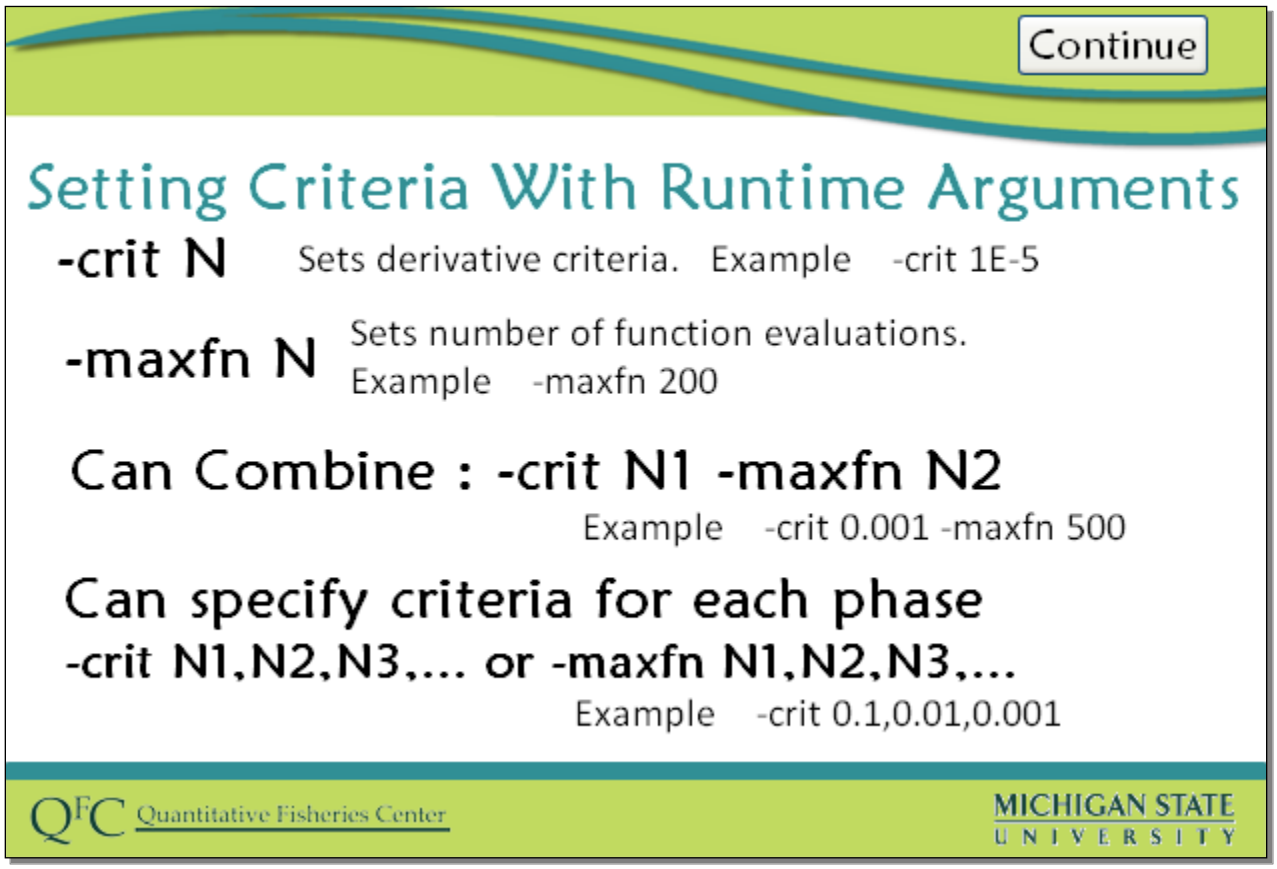
Slide Code:

convergence_criteria 0.1,0.01,0.0001

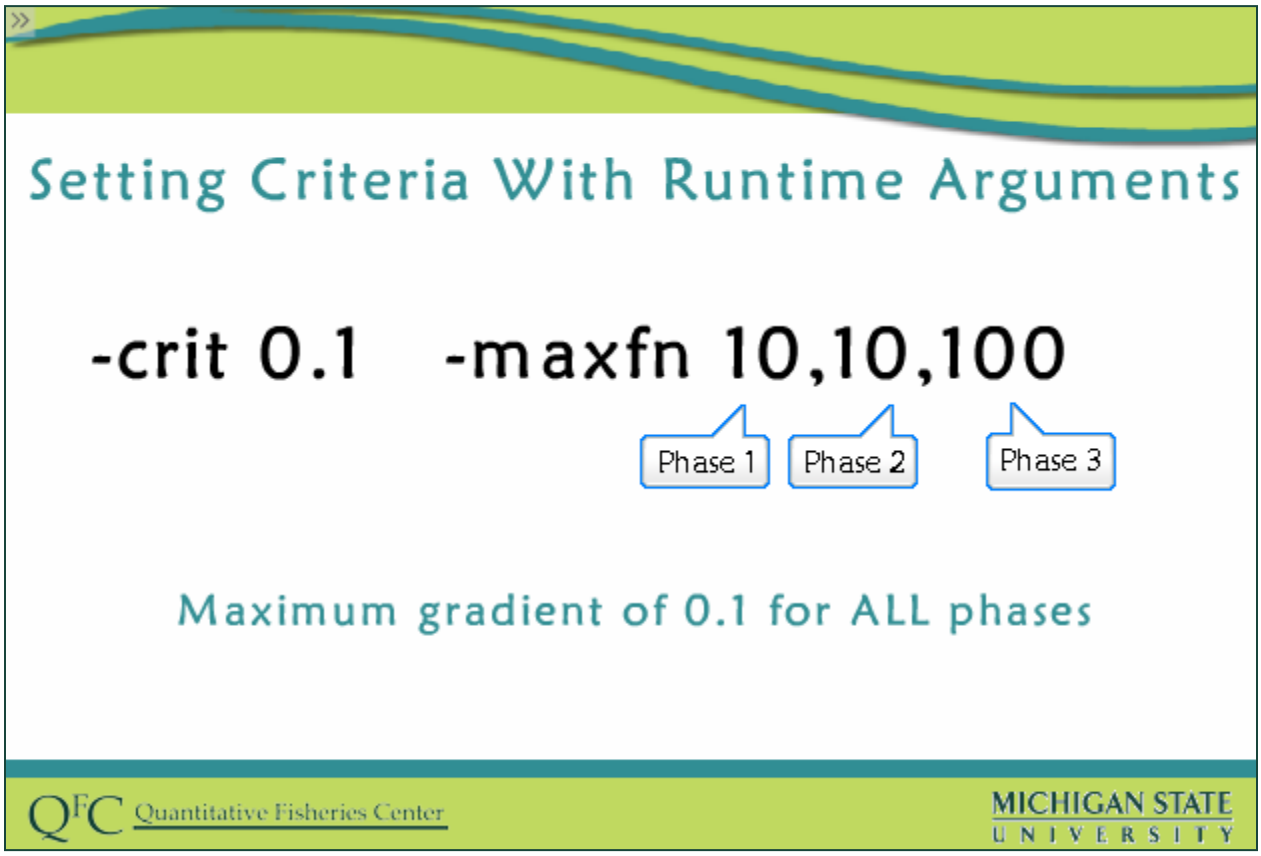
maximum_function_evaluations 10,100,1000



It is nice to include the convergence criterion and maximum number of function evaluations in your tpl because this documents what the criteria were. However, it can be tedious to keep editing the tpl and rebuilding your application when experimenting with different values. For such experimentation, you can actually set the criteria as arguments when you run your program.

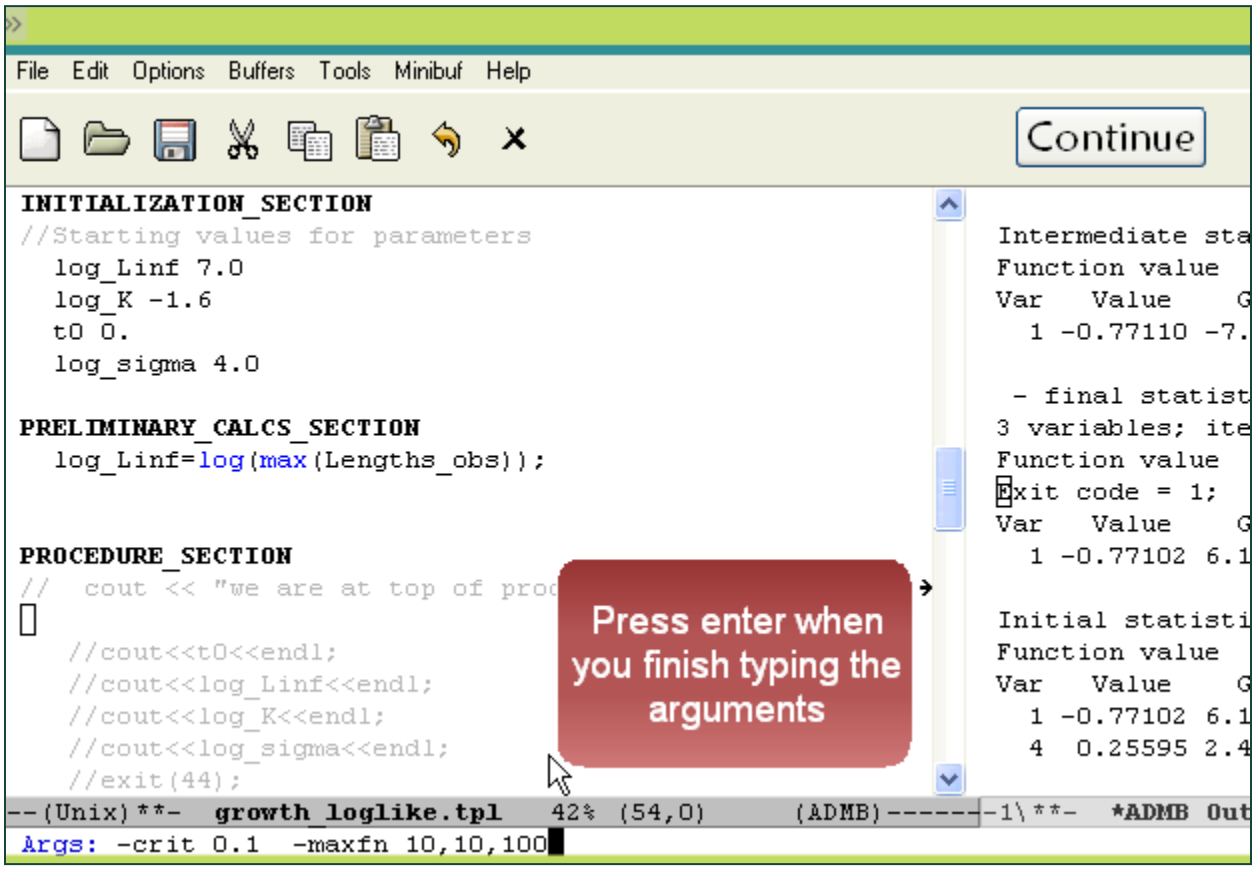


The argument used to set the derivative criteria is dash crit followed by the numerical value and the argument used to set the maximum number of function evaluations is dash maxfn followed by the numerical value you want. Like other runtime arguments, you can specify more than one at a time and thus you could specify both the derivative criterion and maximum function evaluations criterion at the same time with run time arguments such as this. You can follow dash crit or dash maxfn by a series of numbers with commas if you want different criteria for different phases



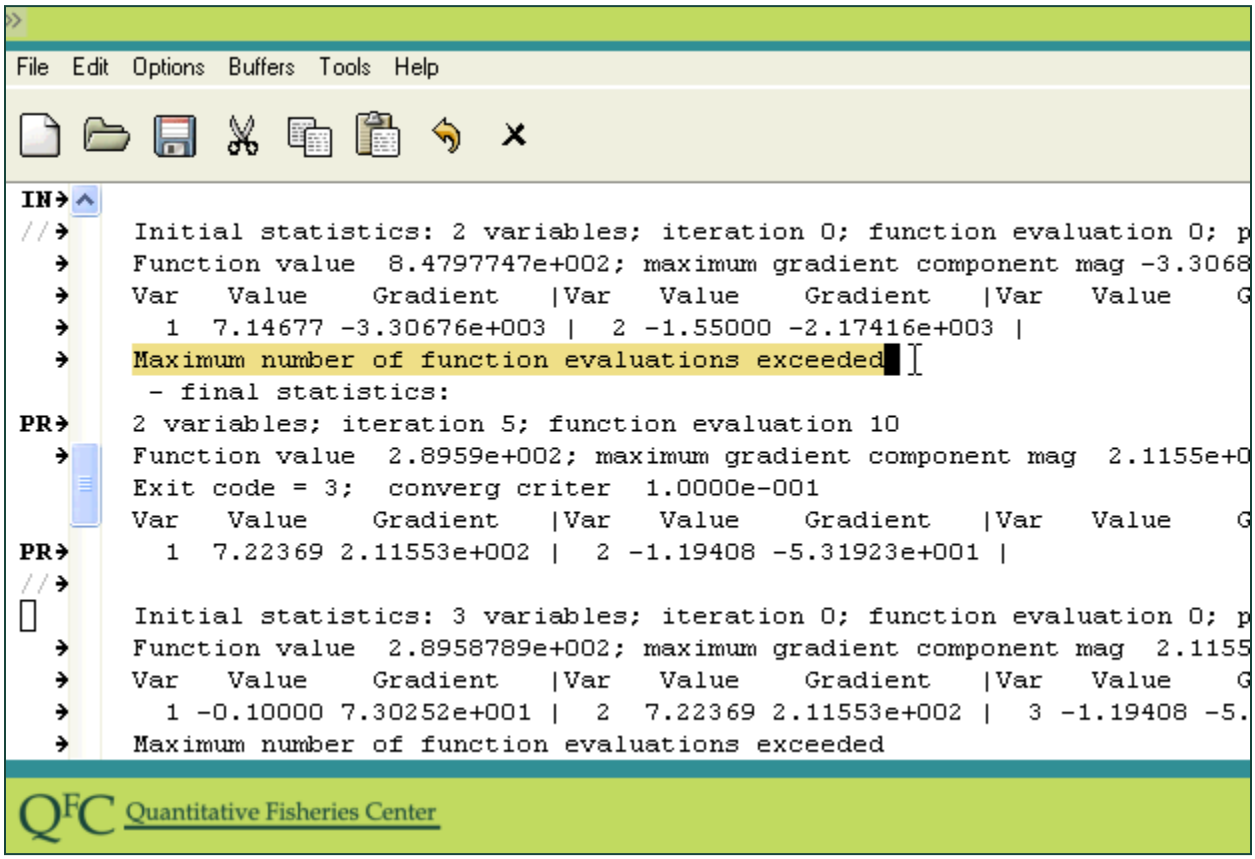
For example we will use the arguments dash crit zero point 1 dash maxfn 10 comma 10 comma 100 this would run our application with a maximum gradient of 0.1 for all phases and maximum numbers of function evaluations of 10 in the first two phases and 100 in the third phase. Now let's see what happens when we use these example arguments.

ADMB – Default Convergence Criteria Video



Slide Code:

1. Go to the **ADMB** menu
2. down to **run with args**.
3. Type **–crit 0.1 -maxfn 10,10,100**
4. press **enter** when finished.



Now let us examine the screen output.

Notice that in phases 1 and 2 it stopped because the maximum number of function evaluation has been exceeded, whereas the fitting stopped because the derivative criterion we specified was met in the last phase. Although information on the maximum gradient component convergence criterion and why estimation ended is shown in the output at run time and the actual maximum gradient component is recorded in the par file, the gradient and number of function call criteria are not recorded automatically in results files. Thus, for a final version of your analysis, if you want to document the convergence criteria used by your program you may want to specify these in your tpl rather than as runtime arguments. For this very simple example, fussing with convergence criteria by phase or even using phases is an overkill but this is good to know about as you move to more complex models for your own work.