

[Continue](#)

More control over the process for searching for the best fitting parameters

Different Ways of Setting Starting Values



This video was created using ADMB-IDE release 4.5.0-1 (July 15, 2011)
You may notice some minor differences if using a different version.

[Continue](#)

3 Ways to Define Starting Values

1. INITIALIZATION_SECTION
2. Pin File
3. PRELIMINARY_CALCS_SECTION

This video will show you three different ways to specify different starting values. First we will revisit the INITIALIZATION_SECTION that was covered in previous videos. Second, we will show you how to create a pin file with starting values. And third, we will introduce you to a new section called the PRELIMINARY_CALCS_SECTION.

[Continue](#)

INITIALIZATION_SECTION

Simplest way to set up starting values

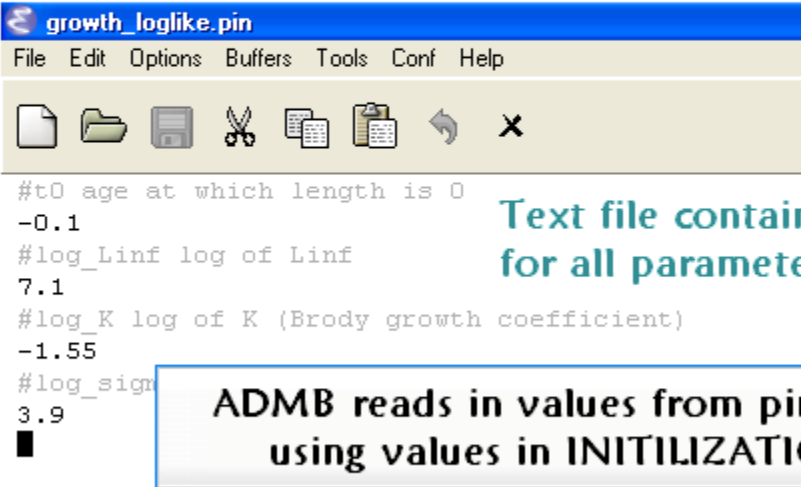
- Need to rebuild each time you change values
- Must specify starting values as a number
- All elements of a vector of parameters must have the same starting values

```
INITIALIZATION_SECTION
//Starting values for parameters
log_Linf 7.0
log_K -1.6
t0 0.
log_sigma 4.0
```

In previous videos you learned how to set starting values in an initialization section. For many simple applications, this is entirely adequate. The syntax is straightforward just give the parameter name followed by the starting value. The potential drawbacks to specifying the starting values this way include that each time you change the starting values you need to build your executable again, that you need to specify them by specific numbers rather than by a calculated variable, that for standard vectors or matrices of parameters you need to start all parameters in the vector or matrix at the same value.

Continue

pin file



```
#t0 age at which length is 0
-0.1
#log_Linf log of Linf
7.1
#log_K log of K (Brody growth coefficient)
-1.55
#log_sign
3.9
█
```

Text file containing starting values for all parameters

ADMB reads in values from pin file rather than using values in INITIALIZATION_SECTION

QFC Quantitative Fisheries Center

MICHIGAN STATE UNIVERSITY

Another way to set up starting values is to create a pin file. The pin file is a text file that contains the starting values for all parameters. ADMB basically reads in the pin file the same way as the dat file is read in. The benefit to setting up starting values with a pin file is that you don't need to re-build your program each time you change the starting values like you would if they were in the INITIALIZATION_SECTION.

Continue

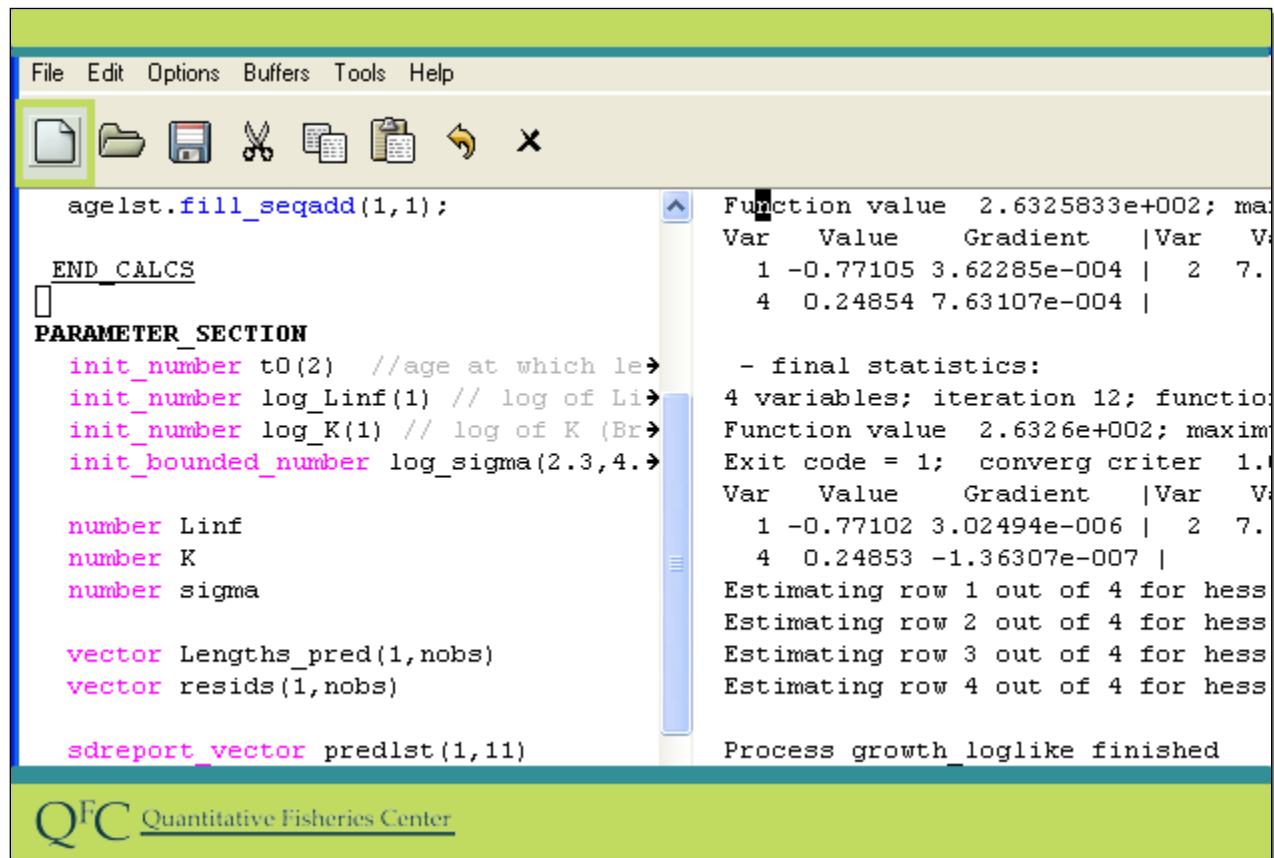
Prepare Data Files

<u>Option 1</u>	<u>Option 2</u>
<ol style="list-style-type: none">1. Open growth_loglike.tpl that you worked with in the previous videos.	<ol style="list-style-type: none">1. Create a new folder.2. Save growth_loglike.dat to the folder.3. Save growth_loglike_SV_Start.tpl to the folder.4. Change the name of the tpl to growth_loglike.tpl5. Open growth_loglike.tpl

QFC Quantitative Fisheries Center

Prepare and open your files. Click next when you are ready. Open growth_loglike.tpl that you worked with in the previous videos if you have been following along, otherwise:

1. Create a new folder.
2. Save growth_loglike.dat to the folder.
3. Save growth_loglike_SV_Start.tpl to the folder.
4. Change the name of the tpl to growth_loglike.tpl
5. Open growth_loglike.tpl



```

age1st.fill_seqadd(1,1);

END_CALCS

PARAMETER_SECTION
  init_number t0(2) //age at which le
  init_number log_Linf(1) // log of Li
  init_number log_K(1) // log of K (Br
  init_bounded_number log_sigma(2.3,4.

  number Linf
  number K
  number sigma

  vector Lengths_pred(1,nobs)
  vector resids(1,nobs)

  sdreport_vector predlst(1,11)

```

```

Function value  2.6325833e+002; ma
Var   Value      Gradient   |Var   V
  1 -0.77105  3.62285e-004 |  2  7.
  4  0.24854  7.63107e-004 |

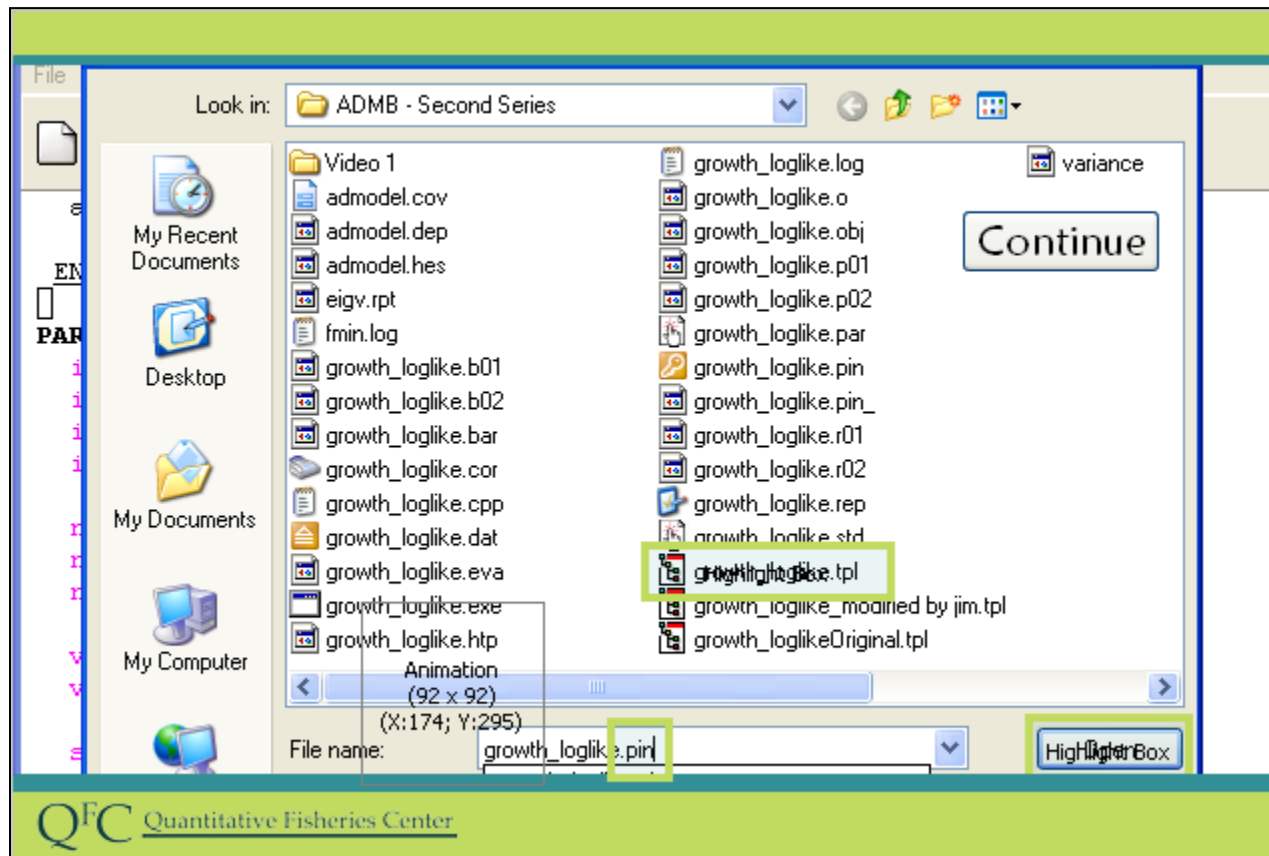
- final statistics:
4 variables; iteration 12; functio
Function value  2.6326e+002; maxim
Exit code = 1;  converg criter  1.
Var   Value      Gradient   |Var   V
  1 -0.77102  3.02494e-006 |  2  7.
  4  0.24853 -1.36307e-007 |
Estimating row 1 out of 4 for hess
Estimating row 2 out of 4 for hess
Estimating row 3 out of 4 for hess
Estimating row 4 out of 4 for hess

Process growth_loglike finished

```

Q^{FC} Quantitative Fisheries Center

Let's create a pin file together. First we will create a new document

**Slide Action:**

Now we will save this new blank file.

1. Select file
2. then save as
3. Name the file growth_loglike.pin

We need to give the file the same name as our template (growth_loglike.tpl) with a .pin extent because by default admb will read starting values from a file with the same name as the model file and extent "pin".

Comment

Continue

```

age1st.fill_seqadd(1,1);

END_CALCS

PARAMETER_SECTION

init_number t0(2) //age at which length is 0
init_number log_Linf(1) // log of Linf
init_number log_K(1) // log of K (Brody growth coefficient)
init_number log_sigma(2.3,4.6,3) // log of sigma for normal distribution

number Linf
number K
number sigma

vector Lengths_pred(1,nobs)
vector resids(1,nobs)

sdreport_vector predlst(1,11)

objective_function_value nll // The negative log likelihood

```

The parameters must be in the same order as they are in the PARAMETER_SECTION

This is our pin file that will hold parameter starting values!

Can be used instead of values specified in the INITIALIZATION_SECTION

You do not need to rebuild each time you change a starting value in a pin file!

QFC Quantitative Fisheries Center

ADMB reads pin files just like it reads dat files, one number after another. It assumes there will be a number for each parameter and assumes they will be in the same order as they are in the parameter section.

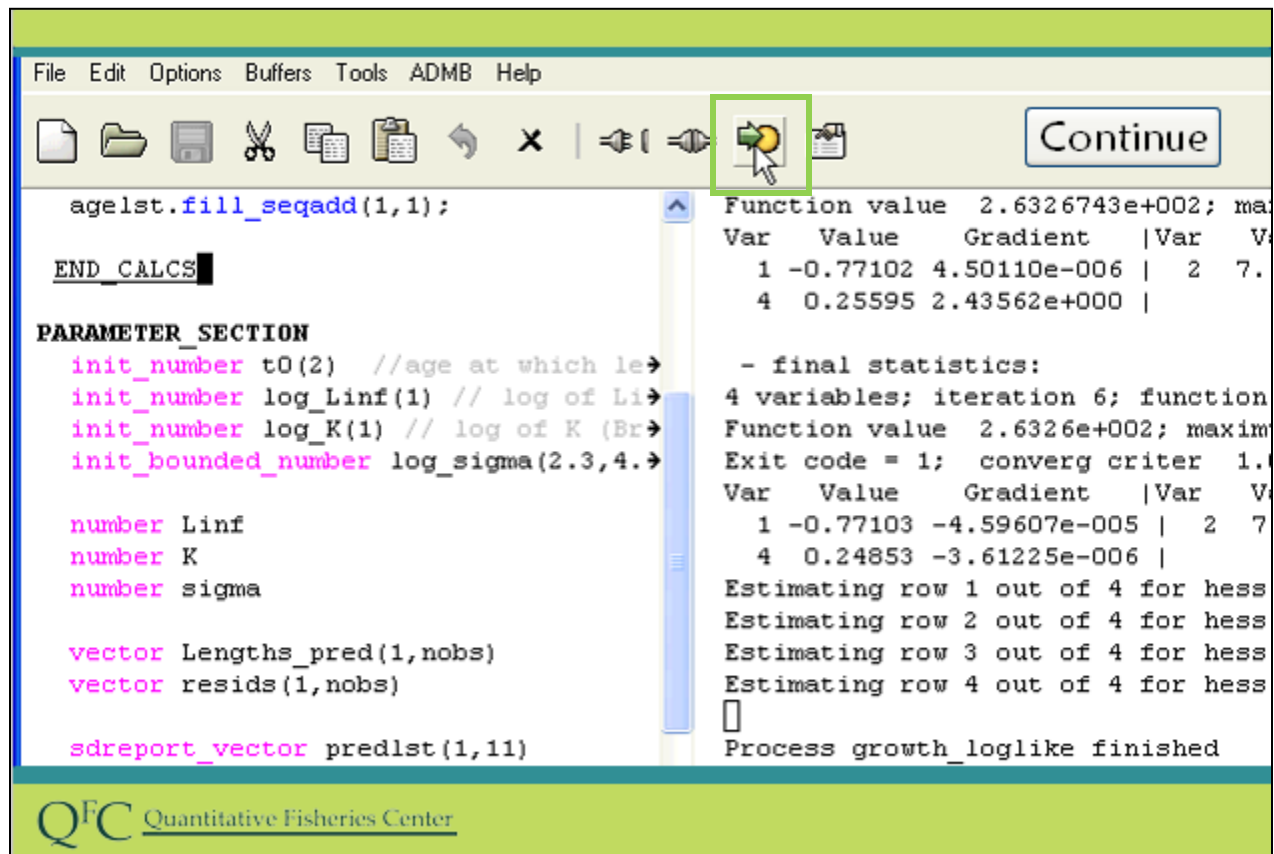
Just like in a dat file, a line starting with # is a comment. This will be our pin file for our growth model. So for each number let us include a comment saying what the parameter is, then the actual starting value we will use. Click next when you have finished typing the values in the pin file to move on.

Slide Code:

```

#t0 age at which length is 0
-0.1
#log_Linf log of Linf
7.1
#log_K log of K (Brody growth coefficient)
-1.55
#log_sigma log of sigma for normal distribution
3.9

```

Now we will save our pin file and run our growth model. It will use the starting values we just entered from the pin file, rather than using any starting values defined in the `INITIALIZATION_SECTION` where we put original starting values. Just looking at the program run it's not so obvious we are using a different set of starting values

Slide Action:

Save .pin file – Go to file to save

Run the program

Continue

```
PROCEDURE_SECTION  
// cout << "we are at top of procedure  
// exit(44);  
  
cout<<t0<<endl;  
cout<<log_Linf<<endl;  
cout<<log_K<<endl;  
cout<<log_sigma<<endl;  
exit(44);  
  
Linf=exp(log_Linf);  
K=exp(log_K);  
sigma=exp(log_sigma);
```

Check to see
which starting
values are being
used

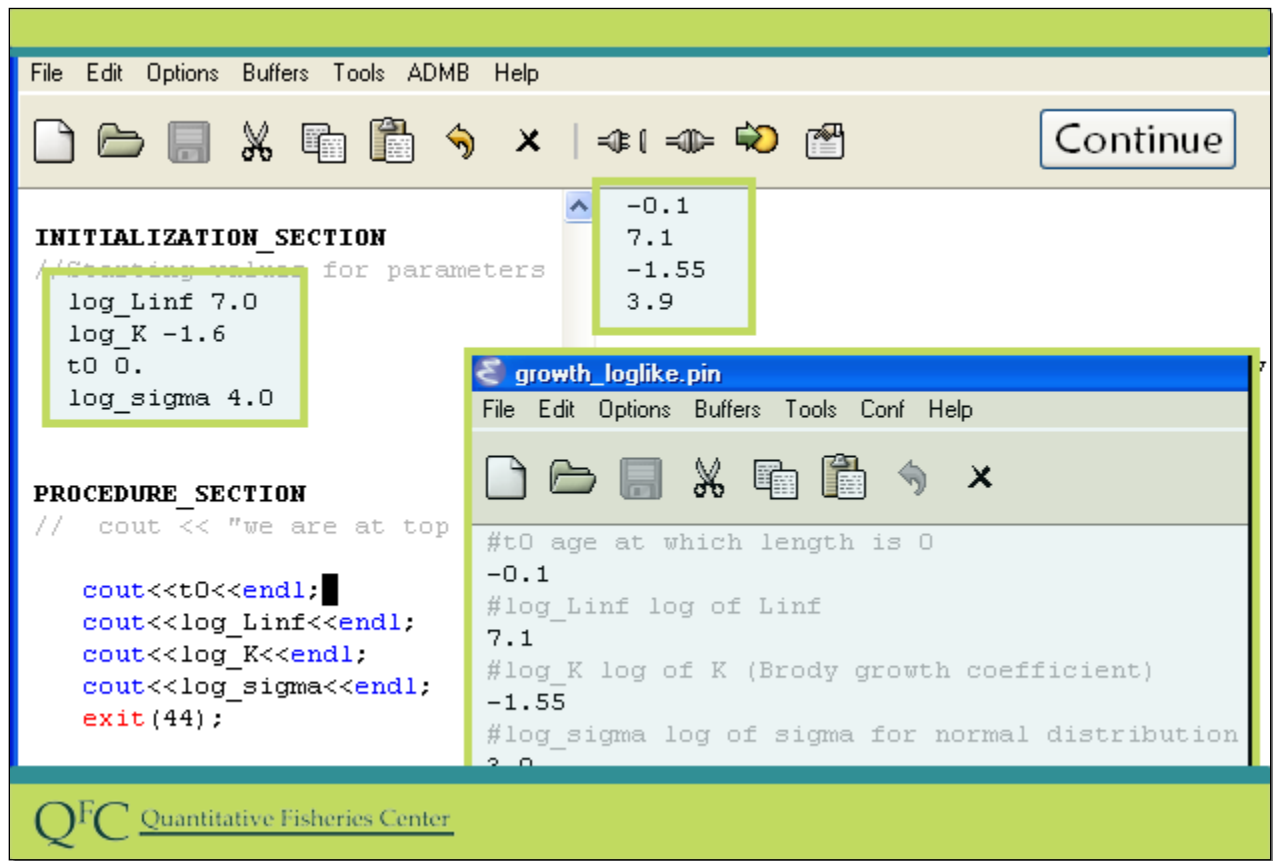
QFC Quantitative Fisheries Center

MICHIGAN STATE
UNIVERSITY

If you ever are unsure what starting values are being used you can “cout” the values right at the top of the PROCEDURE_SECTION and then exit. **Add these lines to your growth_loglike.tpl** if you are following along then press next to continue.

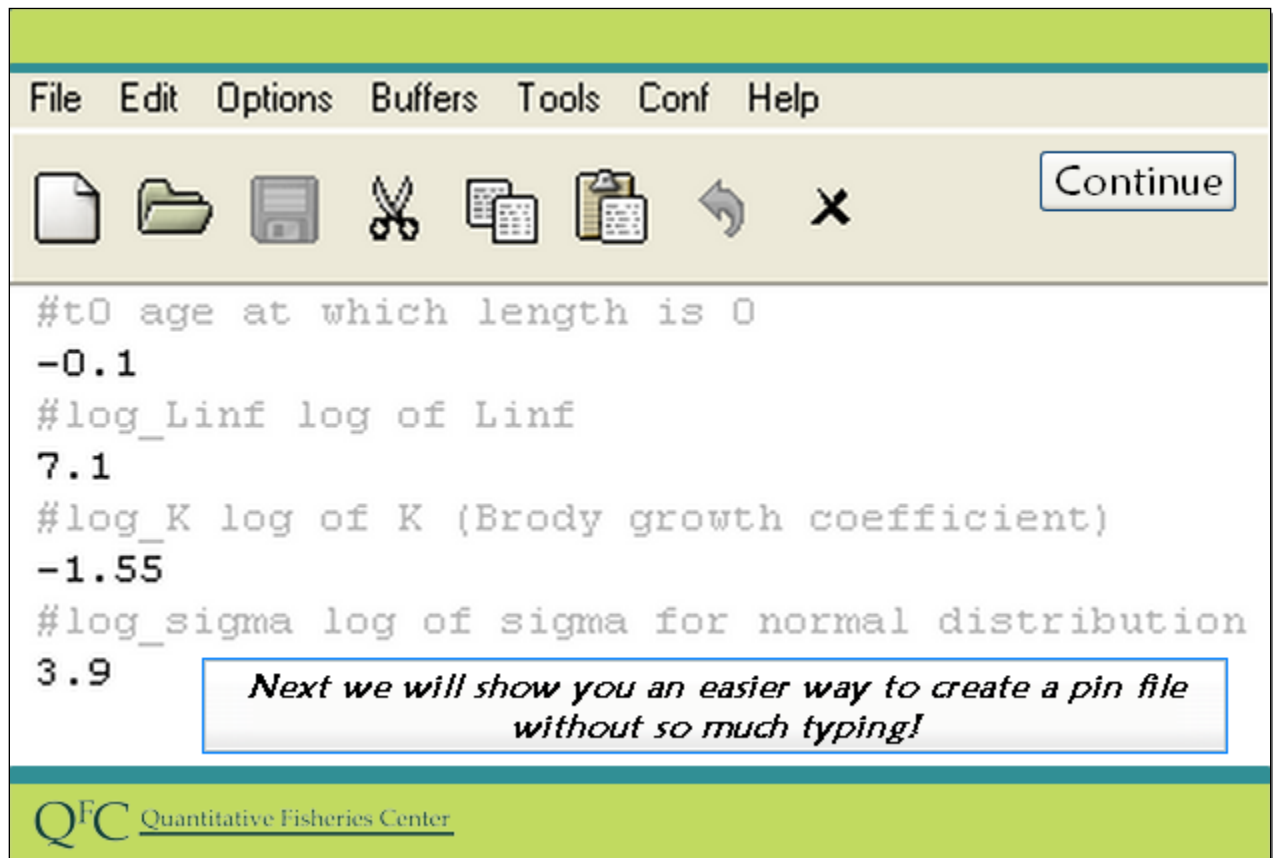
Slide Code:

```
cout<<t0<<endl;  
cout<<log_Linf<<endl;  
cout<<log_K<<endl;  
cout<<log_sigma<<endl;  
exit(44);
```

**Slide Action:**

1. Again build
2. and run the revised program.

We now see that we are using the starting values from the pin file. Notice I actually chose slightly different starting numbers from what we have used before. This is so we can check to make sure our new starting values are getting used.



With a model as simple as the growth model, with four parameters, it's easy to create the pin file manually. With more complex models with 10s to 100s of parameters, creating the file manually can be tedious, especially when many of the parameters you would have just left at their default starting values. Next, we will show you a quick way to create a pin file that already has the structure and parameter comments for all the parameters in the model.

The screenshot shows the ADMB software interface. The left pane displays a C++ source file named `growth_loglike.tpl`. The code includes initial values for `log_Linf`, `log_K`, `t0`, and `log_sigma`, followed by a `PROCEDURE_SECTION` containing commented-out `cout` and `exit` statements. The right pane shows the compilation output, indicating that the file was successfully compiled. The status bar at the bottom shows the file is 43% compiled and that no files need saving.

```

File Edit Options Buffers Tools ADMB Help

log_Linf 7.0
log_K -1.6
t0 0.
log_sigma 4.0

PROCEDURE_SECTION
// cout << "we are at top of proc>
// cout<<t0<<endl;
// cout<<log_Linf<<endl;
// cout<<log_K<<endl;
// cout<<log_sigma<<endl;
// exit(44);

Linf=exp(log_Linf);
K=exp(log_K);
--(Unix)--- growth_loglike.tpl 43% (-1\%*- *compilation* All (1,0)
(No files need saving)

```

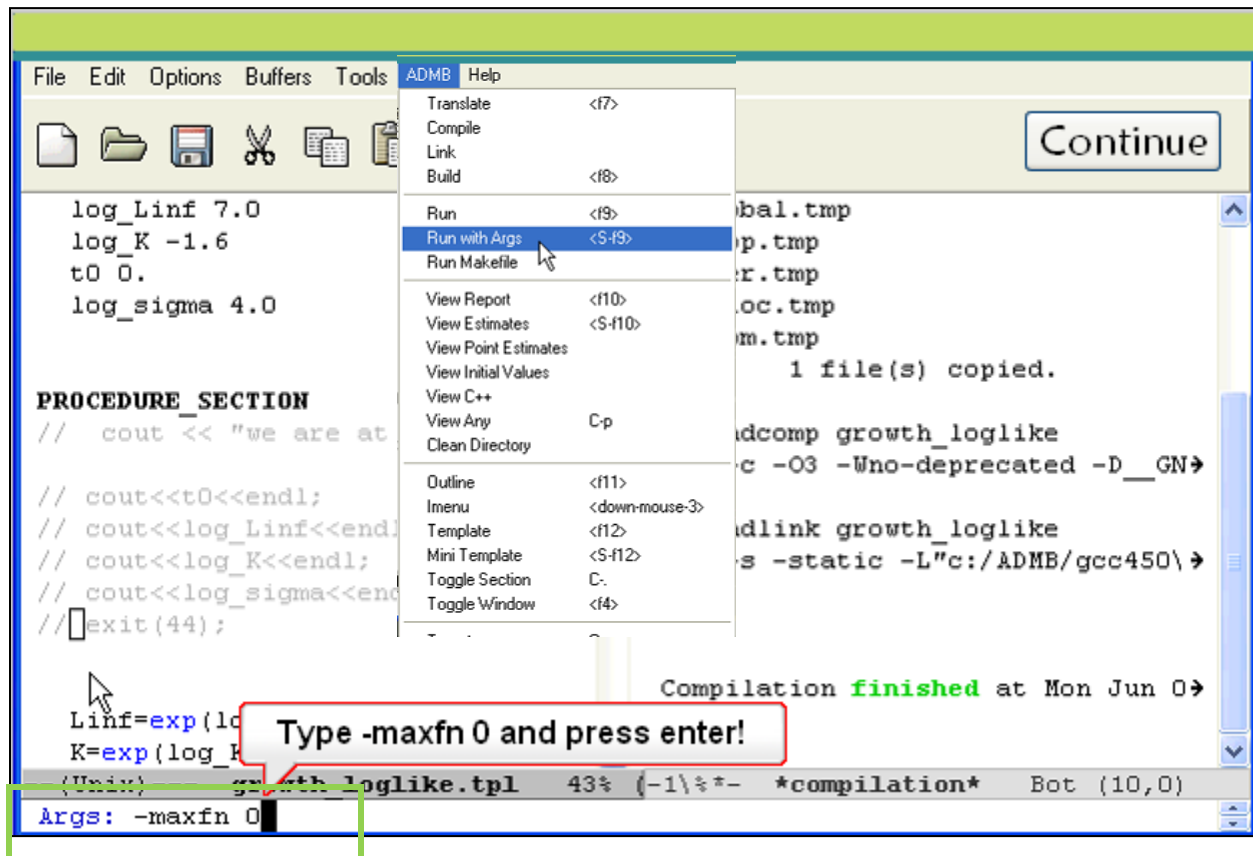
1. Before we begin, we need to **comment out** our `cout` and `exit` statements that we put in place to check which starting values we used. If we did not comment these out our program would exit before it ran.
2. Now we can **rebuild** our program by pressing the build button.

Slide Code:

```

// cout<<t0<<endl;
// cout<<log_Linf<<endl;
// cout<<log_K<<endl;
// cout<<log_sigma<<endl;
// exit(44);

```



An easy way to create an initial version for a pin file is to run the program with arguments, and use the argument “-maxfn 0”. This basically tells admb to stop before it starts.

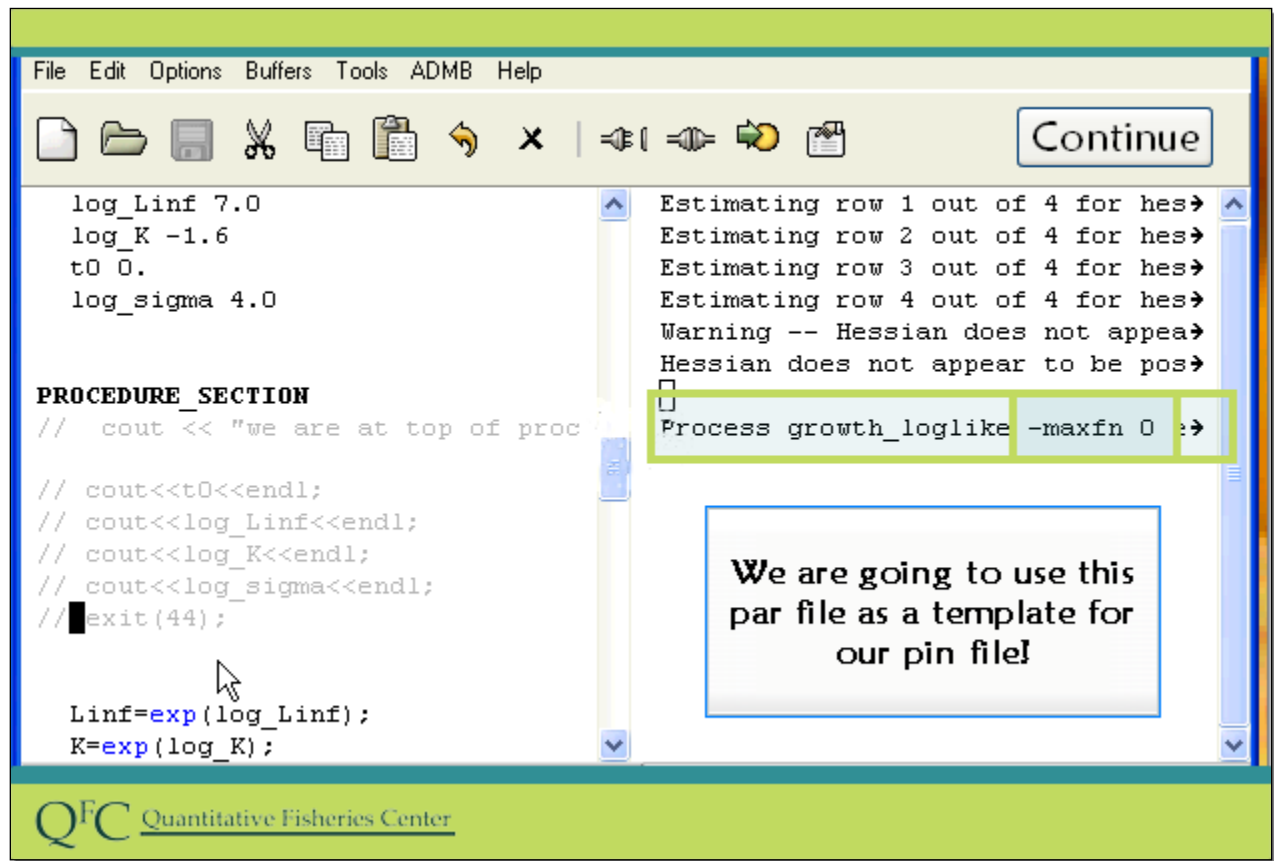
1. Go to the **ADMB** menu, down to **run with Args**.
2. Type **-maxfn 0** in the buffer
3. and press **enter**

Slide Code:

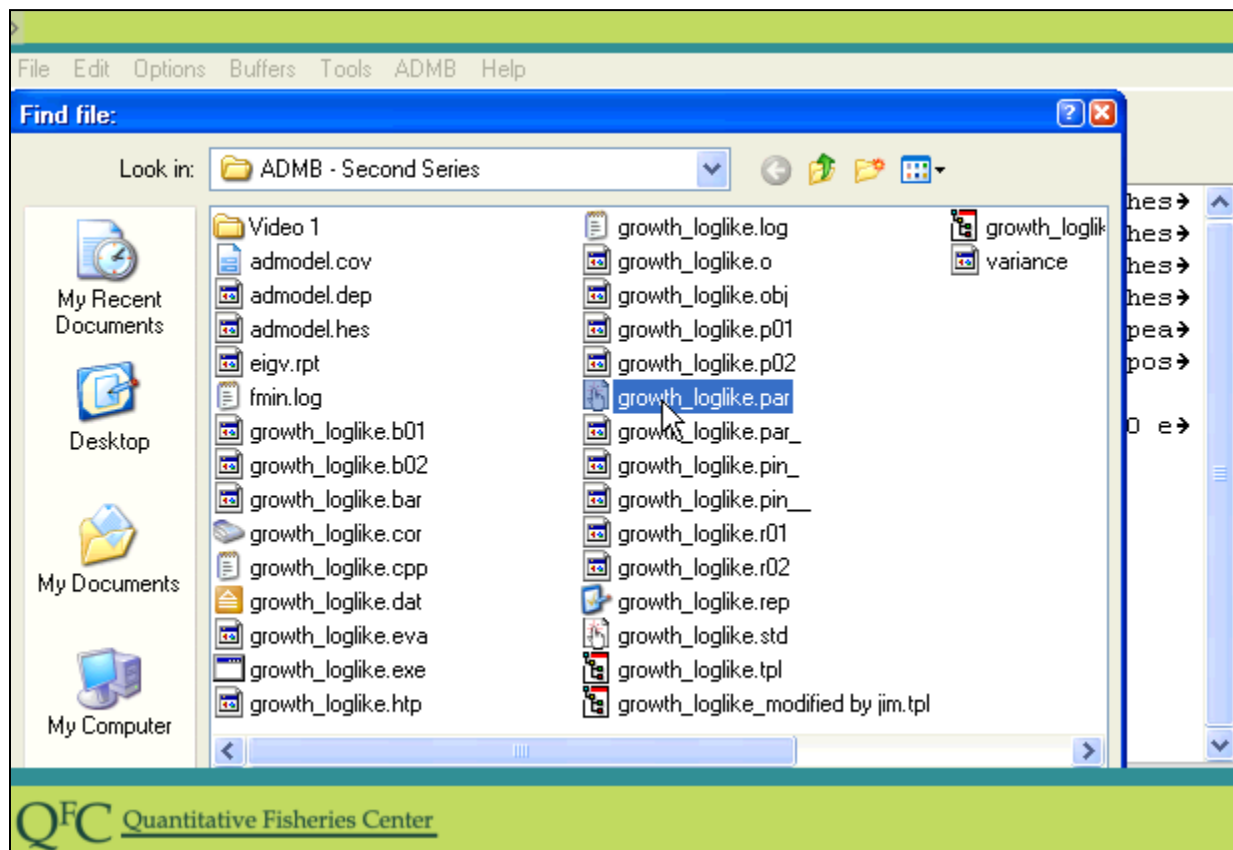
Go to **ADMB** menu and select “Run with Args”

Type **-maxfn 0** in the buffer

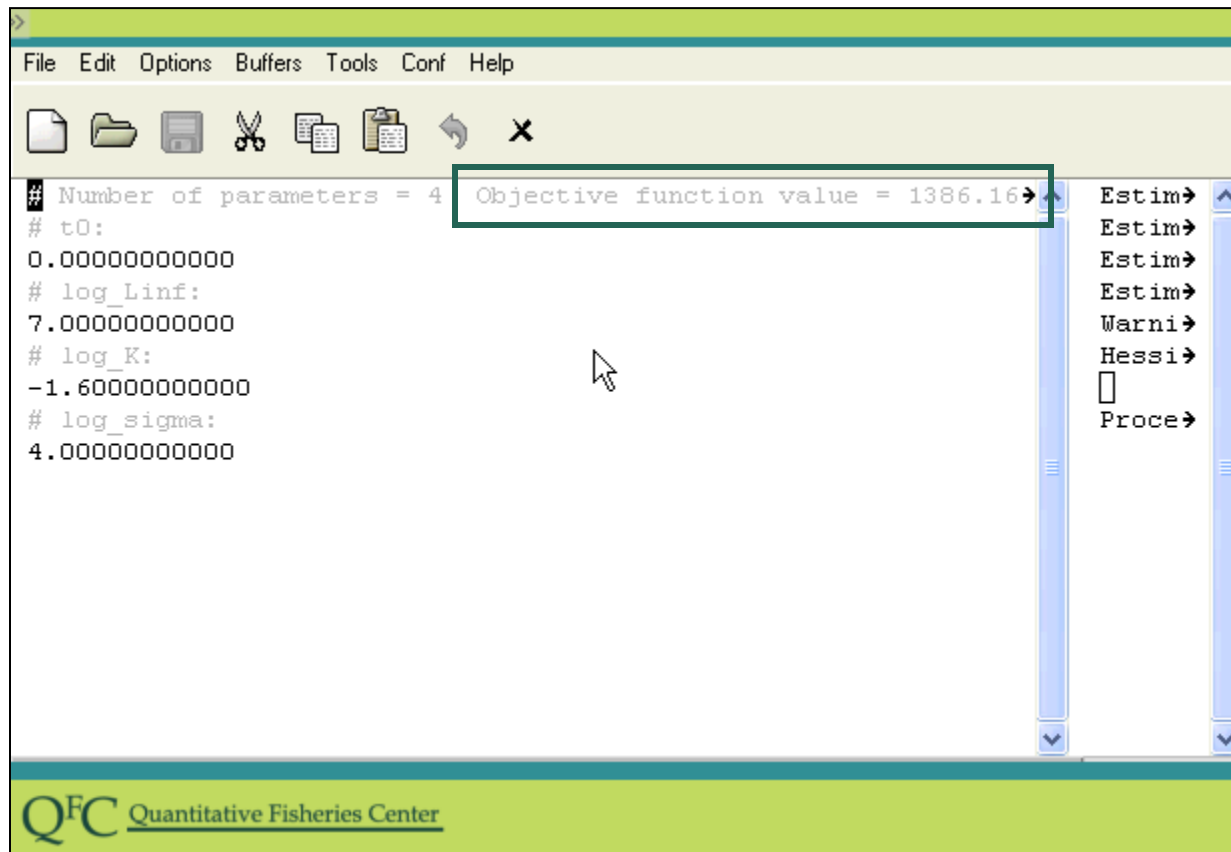
Press enter



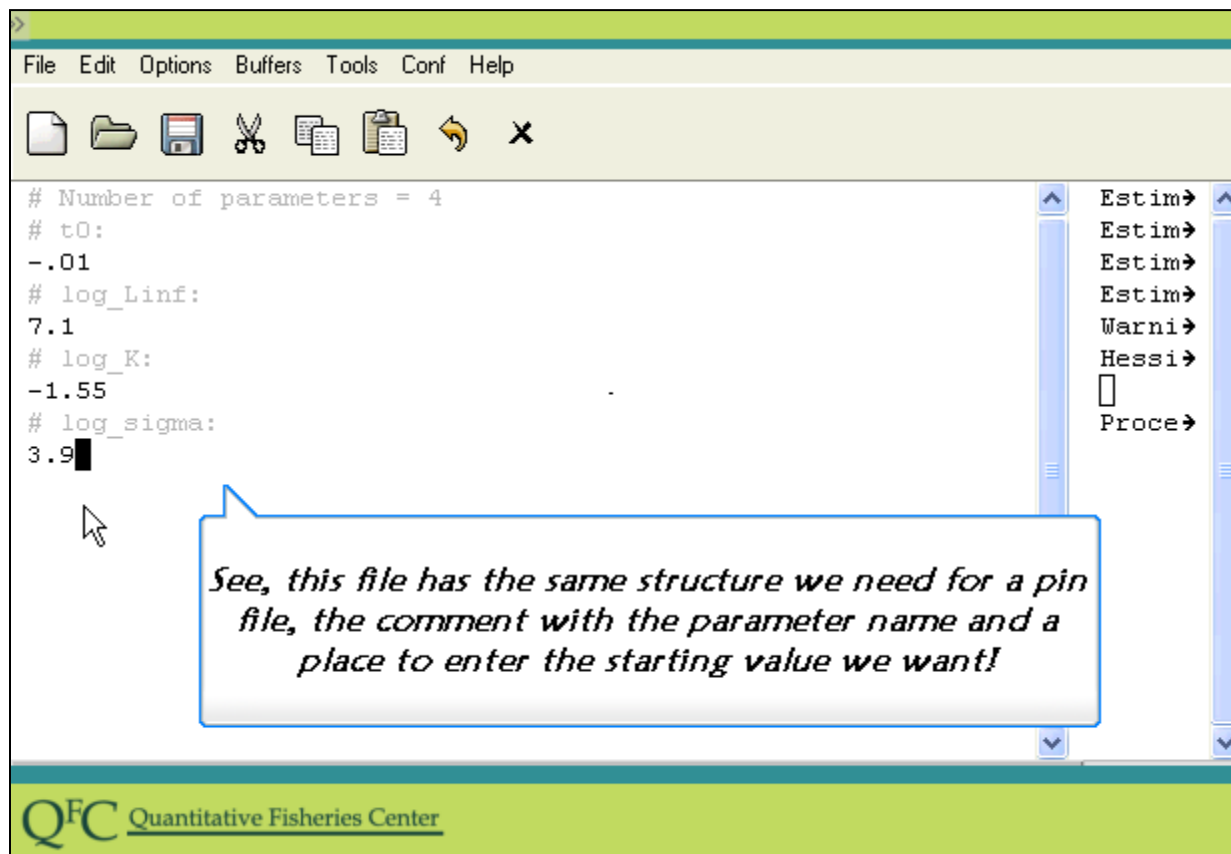
Don't worry about the error messages you get they are just telling you what you already know, the model did not converge. The program just stopped because the maximum number of times the objective function could be evaluated was exceeded because we set this to zero. When it ended, however it saved a .par file. This file has the same structure as the .pin file.



Now we can open up the .par file we just created. (growth_loglike.par)



There is some extraneous stuff about convergence status including the “Objective function value and Maximum gradient component, and you can delete this if you want.



1. We can now edit the values so the starting values are as desired. (-.01, 7.1, -1.55, 3.9)

Slide Code:

#t0 age at which length is 0

-0.01

#log_Linf log of Linf

7.1

#log_K log of K (Brody growth coefficient)

-1.55

#log_sigma log of sigma for normal distribution

3.9

2. Now we can **save this as a .pin file** by going to **file** to **save as**
3. We need to **change the par extension to a pin file** so the file is named **growth_loglike.pin**

This approach to creating a pin file saves you the trouble of writing out your own comments indicating which parameter is which and also helps make sure your pin file has the right number of parameters and in the right order.

[Continue](#)

Third Way to Set Starting Values

Use the PRELIMINARY_CALCS_SECTION

Advantages

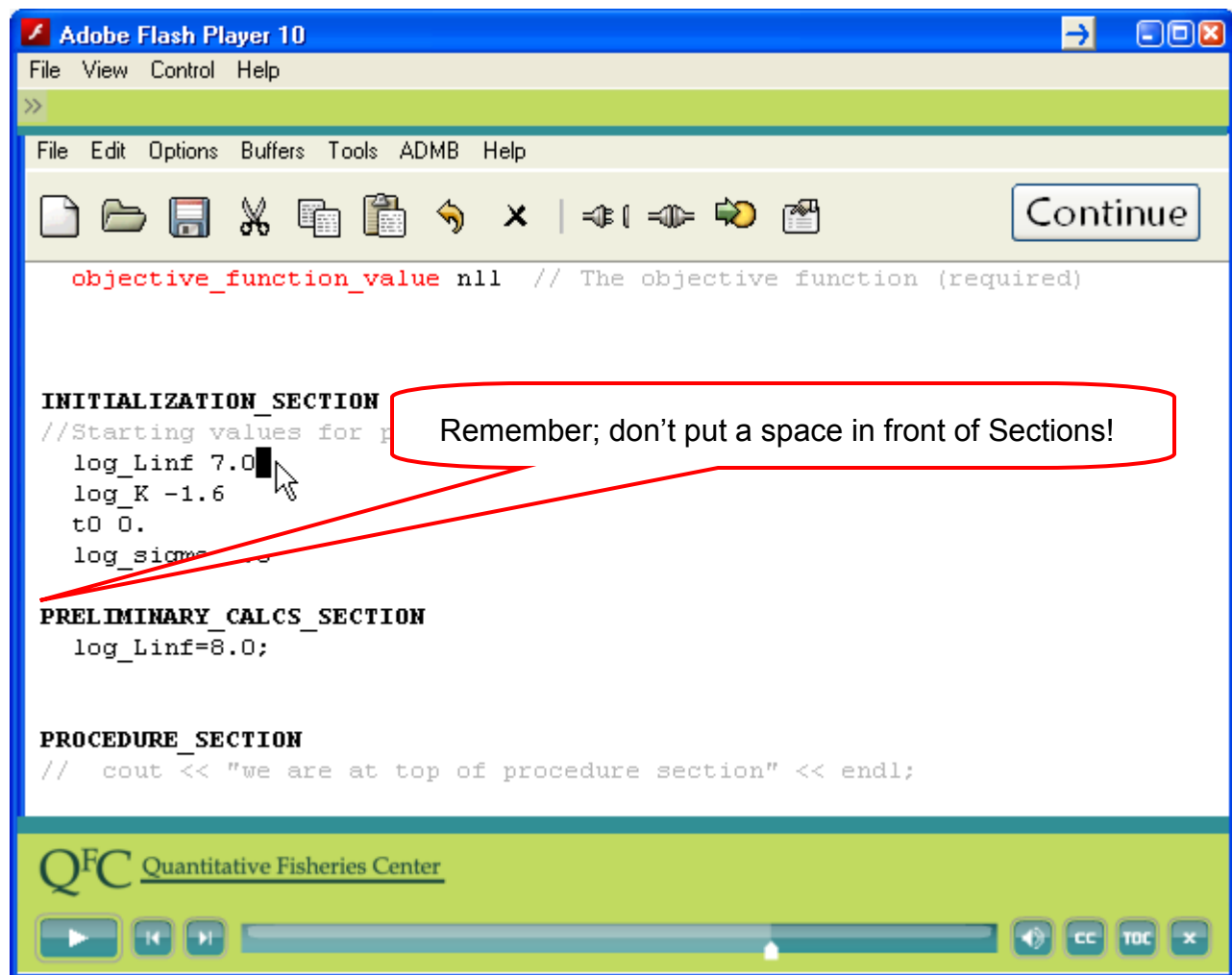
You can specify for just one element of a vector of parameters.

Calculate values rather than having to specify numerically.

Disadvantage

Must rebuild each time you change something.

A third way to set starting values for parameters is in the PRELIMINARY_CALCS_SECTION. Here you include code just like you would in the procedure section. Advantages of this way of setting starting values includes: the ability that you can specify for just one element of a vector of parameters rather than all which you have to do with the other ways of setting starting values, and you can calculate starting values rather than having to specify them numerically. A disadvantage to this way is you must rebuild the program each time you change the values.

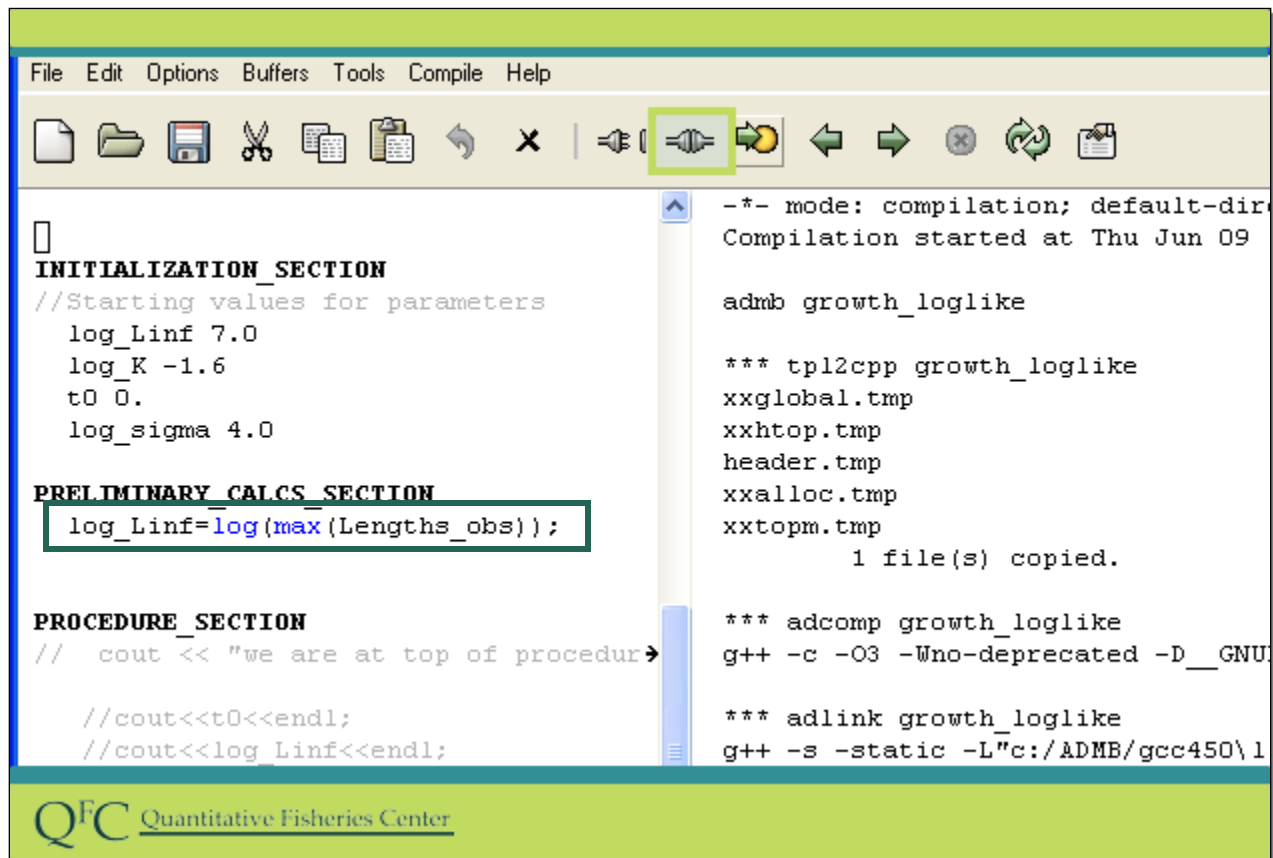


Here is a simple example. We will add the **PRELIMINARY_CALCS_SECTION** with `log_Linf = 8.0`. This example is so simple you probably would never actually do this since it would be just as easy to set the starting value in the initialization section.

Slide Code:

PRELIMINARY_CALCS_SECTION

`log_Linf=8.0;`



The screenshot shows a C++ IDE with a menu bar (File, Edit, Options, Buffers, Tools, Compile, Help) and a toolbar. The main editor window displays C++ code with three sections: **INITIALIZATION_SECTION**, **PRELIMINARY_CALCS_SECTION**, and **PROCEDURE_SECTION**. In the **PRELIMINARY_CALCS_SECTION**, the line `log_Linf=log(max(Lenghts_obs));` is highlighted with a green box. The **PROCEDURE_SECTION** contains a commented-out line `// cout << "we are at top of procedur`. The right-hand pane shows the compilation output, including the mode (compilation), default directory, and the start time (Thu Jun 09). It lists the files being compiled: `admb growth_loglike`, `*** tpl2cpp growth_loglike`, `xxglobal.tmp`, `xxhtop.tmp`, `header.tmp`, `xxalloc.tmp`, and `xxtopm.tmp`. It also shows the command `g++ -c -O3 -Wno-deprecated -D__GNU` and the output `1 file(s) copied.`

```
File Edit Options Buffers Tools Compile Help

[Icons]

[ ]
INITIALIZATION_SECTION
//Starting values for parameters
log_Linf 7.0
log_K -1.6
t0 0.
log_sigma 4.0

PRELIMINARY_CALCS_SECTION
log_Linf=log(max(Lenghts_obs));

PROCEDURE_SECTION
// cout << "we are at top of procedur

//cout<<t0<<endl;
//cout<<log_Linf<<endl;
```

QFC Quantitative Fisheries Center

Where this method is most useful is when you want to set a starting value based on the data. For example you could set the starting asymptotic length to the maximum of the observed lengths so we will take out the 8 and replace it with the log of the max observed lengths from our data.

Slide Code:

PRELIMINARY_CALCS_SECTION

```
log_Linf=log(max(Lenghts_obs));
```

The screenshot shows a C++ IDE with a code editor on the left and a console output on the right. The code editor contains three sections: **INITIALIZATION_SECTION**, **PRELIMINARY_CALCS_SECTION**, and **PROCEDURE_SECTION**. In the **INITIALIZATION_SECTION**, the line `log_Linf 7.0` is highlighted with a yellow box. The **PRELIMINARY_CALCS_SECTION** contains `log_Linf=log(max(Lenghts_ob`. The **PROCEDURE_SECTION** contains several `cout` statements. The console output on the right shows the program's execution, including function values, gradients, and intermediate statistics. A "Continue" button is visible in the top right corner of the IDE window.

```

INITIALIZATION_SECTION
//Starting values for parameter
log_Linf 7.0
log_K -1.6
t0 0.
log_sigma 4.0

PRELIMINARY_CALCS_SECTION
log_Linf=log(max(Lenghts_ob

PROCEDURE_SECTION
// cout << "we are at top of
// exit(44);

// cout<<t0<<endl;
// cout<<log_Linf<<endl;
// cout<<log K<<endl;
  
```

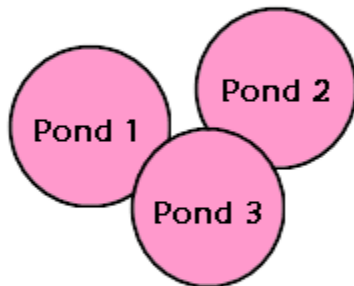
Function value 2.6670718e+002; maximum gra
 Var Value Gradient |Var Value G
 1 -0.01000 1.10396e+001 | 2 7.07675 -2.
 Intermediate statistics: 3 variables; itera
 Function value 2.6326743e+002; maximum gra
 Var Value Gradient |Var Value G
 1 -0.77108 -3.34156e-003 | 2 7.14887 -3
 - final statistics:
 3 variables; iteration 13; function evaluat
 Function value 2.6327e+002; maximum gradie
 Exit code = 1; converg criter 1.0000e-004
 Var Value Gradient |Var Value G
 1 -0.77102 -1.44828e-006 | 2 7.14887 1.
 Initial statistics: 4 variables; iteration
 Function value 2.6326743e+002; maximum gra
 Var Value Gradient |Var Value G

1. We then rebuild
2. and run the program.

Note that this approach to setting starting values writes over any starting values you set in the pin file (which we assigned at 7.1) as well as any starting values set in the **INITIALIZATION_SECTION** (which was assigned 7.0).

[Continue](#)

Use With a Vector of Parameters



`init_number log_Linf`

1. Added bounded
2. Changed from number to vector
3. Changed parameter name
4. Added vector index and bounds

`init_bounded_vector log_Linfs[1,3,5.5,7.5]`

a vector of parameters, all of which can be estimated.

This method can be used as a quick and dirty way to set different starting values for different elements of a parameter vector or matrix. For example maybe the data come from three ponds and each pond has its own asymptotic length. We could change our asymptotic length parameter `Linf` to a vector of parameters `Linfs`. Note the changes. Creating this vector of parameters allows us to set different starting values for each of the ponds, rather than just using one starting value for all three.

File Edit Options Buffers Tools ADMB Help

Continue

END_CALCS

PARAMETER_SECTION

```
init_number t0(2) //age at which length is 0
init_bounded_vector log_Linfs(1,3,5.5,7.5) // log of Linf
init_number log_K(1) // log of K (Brody growth coefficient)
init_bounded_number log_sigma(2.3,4.6,3) //log of sigma

number Linf
number K
number sigma

vector Lengths_pred(1,nobs)
vector resids(1,nobs)

sdreport_vector predlst(1,11)

objective_function_value null // The objective function
```

Initial status
Function value
Var Value
1 7.14677 -

- final status
2 variables; 1
Function value
Exit code = 1;
Var Value
1 7.08532 -

Initial status
Function value
Var Value
1 -0.10000 9

1. Add bounded
2. Change number to vector
3. Add an "s" to the parameter name
4. Add indices and bounds

QFC Quantitative Fisheries Center

Make the changes to our template. We added bounded, changed number to vector, added an s to the parameter name and added the indices and bounds.

Of course this means our code would need to know how to use a vector of asymptotic lengths, thus this would require lots of other changes.

Slide Code:

```
init_bounded_vector log_Linfs(1,3,5.5,7.5)
```


The screenshot shows the ADMB software interface. The main window displays Fortran code for a von Bertalanffy growth model. The code includes a `PARAMETER_SECTION` where parameters are defined. A callout box highlights the change from `number Linf` to `vector Linfs`.

```

Lengths_obs=column(vonBdata,2);
age1st.fill_seqadd(1,1);

END_CALC

PARAMETER_SECTION
  init_number t0(2) //age at which length is 0
  init_bounded_vector log_Linfs(1,3,5.5,7.5) // log of Linf
  init_number log_K(1) // log of K (Brody growth coefficient)
  init_bounded_number log_sigma(2.3,4.6,3) //log of sigma

  vector Linfs(1,3)
  number K
  number sigma

  vector Lengths_pred(1,nobs)
  vector resids(1,nobs)
  
```

The console window on the right shows the output of the model, including the initial status, function value, and exit code.

Initial status
Function value
Var Value
1 7.14677 -

- final status
2 variables; 1
Function value
Exit code = 1;
Var Value
1 7.08532 -

al status
ion value
Value
0.10000 9

1. Change number to vector
2. Add an "s" to the parameter name
3. Add indices

QFC Quantitative Fisheries Center

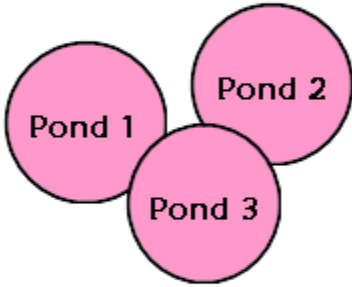
For example, we need to change the object holding the backtransformed asymptotic lengths to a vector also.

Slide Code:

vector Linfs (1,3) //(instead of number Linf)

Continue

PRELIMINARY_CALCS_SECTION



You don't need to type this code...it is just an example of what you could do.

PRELIMINARY_CALCS_SECTION

```
log_Linfs(1)=5.7; //first pond  
log_Linfs(2)=6.0; //second pond  
log_Linfs(3)=6.5; //third pond
```

Q^FC Quantitative Fisheries Center

MICHIGAN STATE UNIVERSITY

You would also set the starting values to different values for each of the three ponds within the PRELIMINARY_CALCS_SECTION with this code. This enables us to set a different starting value for each of the ponds rather than being constrained by using one starting value for all the ponds. We will not rebuild and rerun the program with these changes because the rest of the model is not set up to use the changes. We just wanted to demonstrate that you could use a parameter of different starting values when using the PRELIMINARY_CALCS_SECTION.

[Continue](#)

Review of Three Methods

1. INITIALIZATION_SECTION

Simple

Need to rebuild each time you change

2. pin file

Takes precedence over INITIALIZATION_SECTION values

Don't need to rebuild each time you change

Need to set values for every parameter

3. PRELIMINARY_CALCS_SECTION

Takes precedence over INITIALIZATION_SECTION and .pin values

Can use calculated values for starting values

Need to rebuild each time you change

As a note of closing regarding specifying starting values, we have described three basic approaches. You can specify them in the initialization section, through a pin file, or via code in the preliminary_calcs section. Using the initialization section is simple but to change them you need to rebuild your application. Specifications in a pin file take precedence over values specified in the initialization section, and they can be changed without rebuilding. You can set parameters in the preliminary calcs section. This is the most flexible approach and values set this way take precedence over values set in both pin files and initialization section. You can also use calculated values for the starting values here rather than just numeric numbers, however, you do need to rebuild. If you specify parameters in a pin file you need to specify values for every parameter, whereas the other approaches allow you to set a subset of parameters.