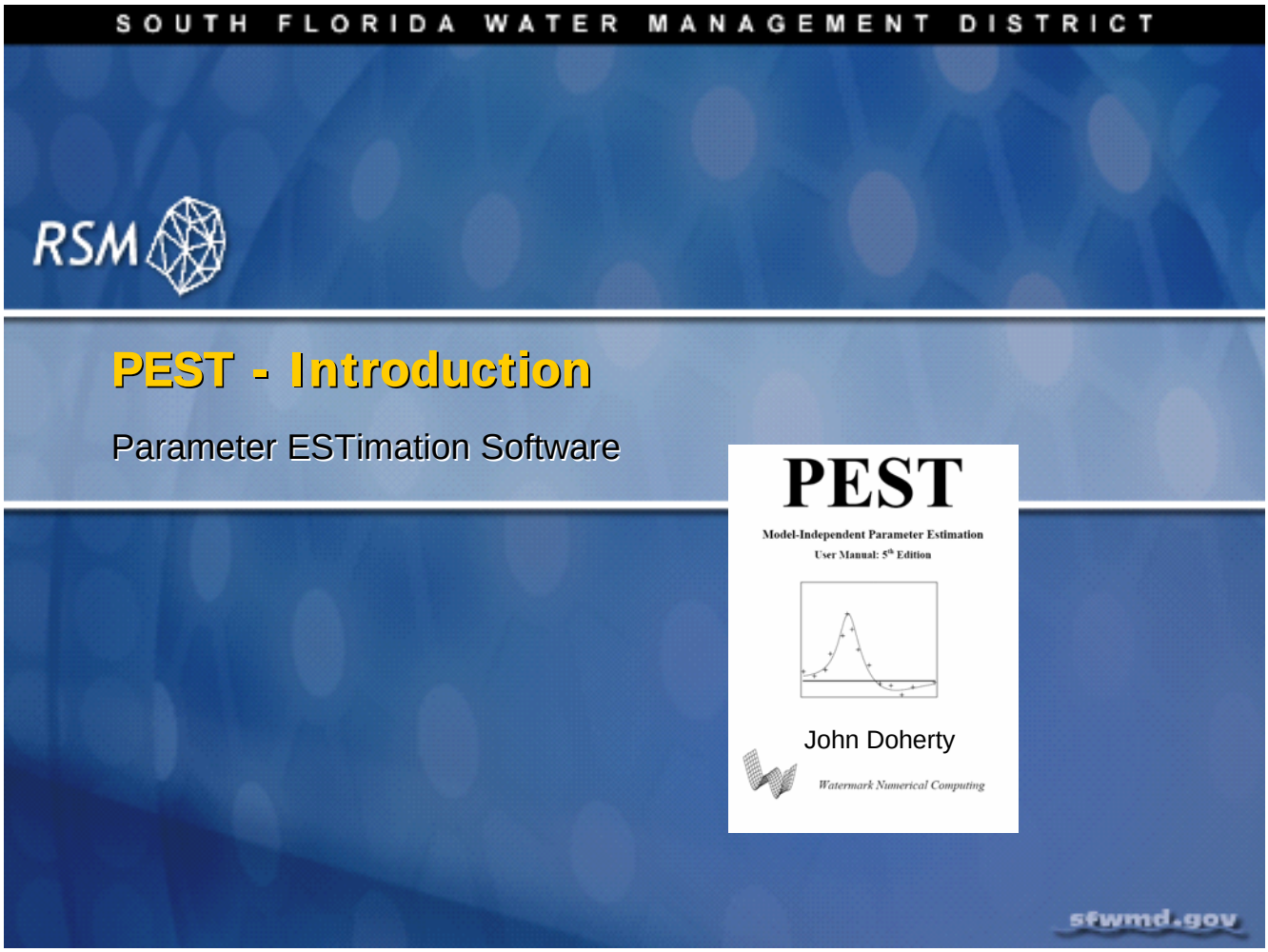


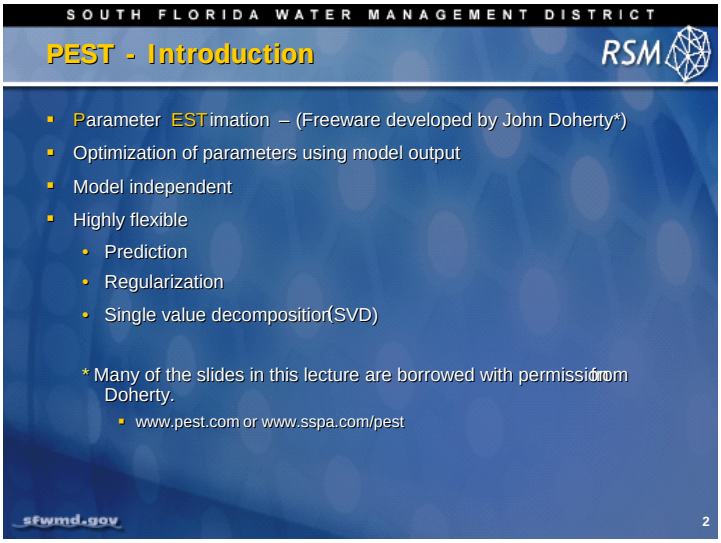
Lecture 13: PEST – Introduction. Parameter ESTimation Software

This lecture introduces the Parameter Estimation (PEST) Software.

For more detailed information, refer to the PEST User Manual, PEST: Model-Independent Parameter Estimation. User Manual 5th Edition. Doherty, John. Watermark Numerical Computing. July, 2004.

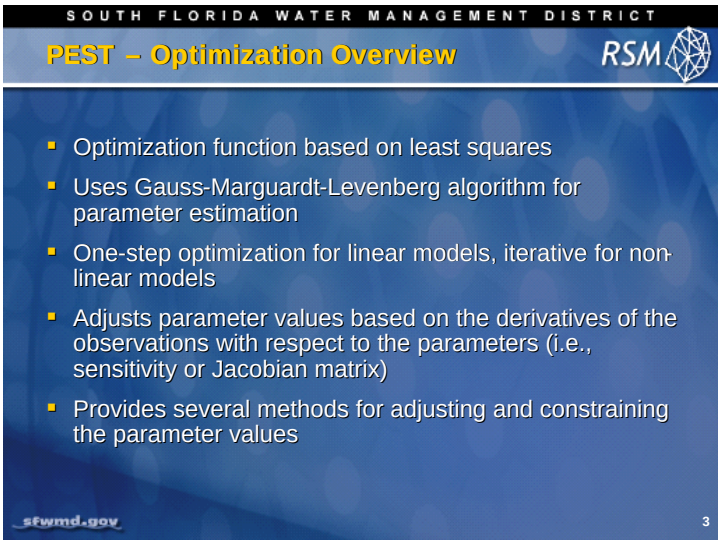
NOTE:

A copy of the PEST User Manual can be found in the labs/lab13 directory.

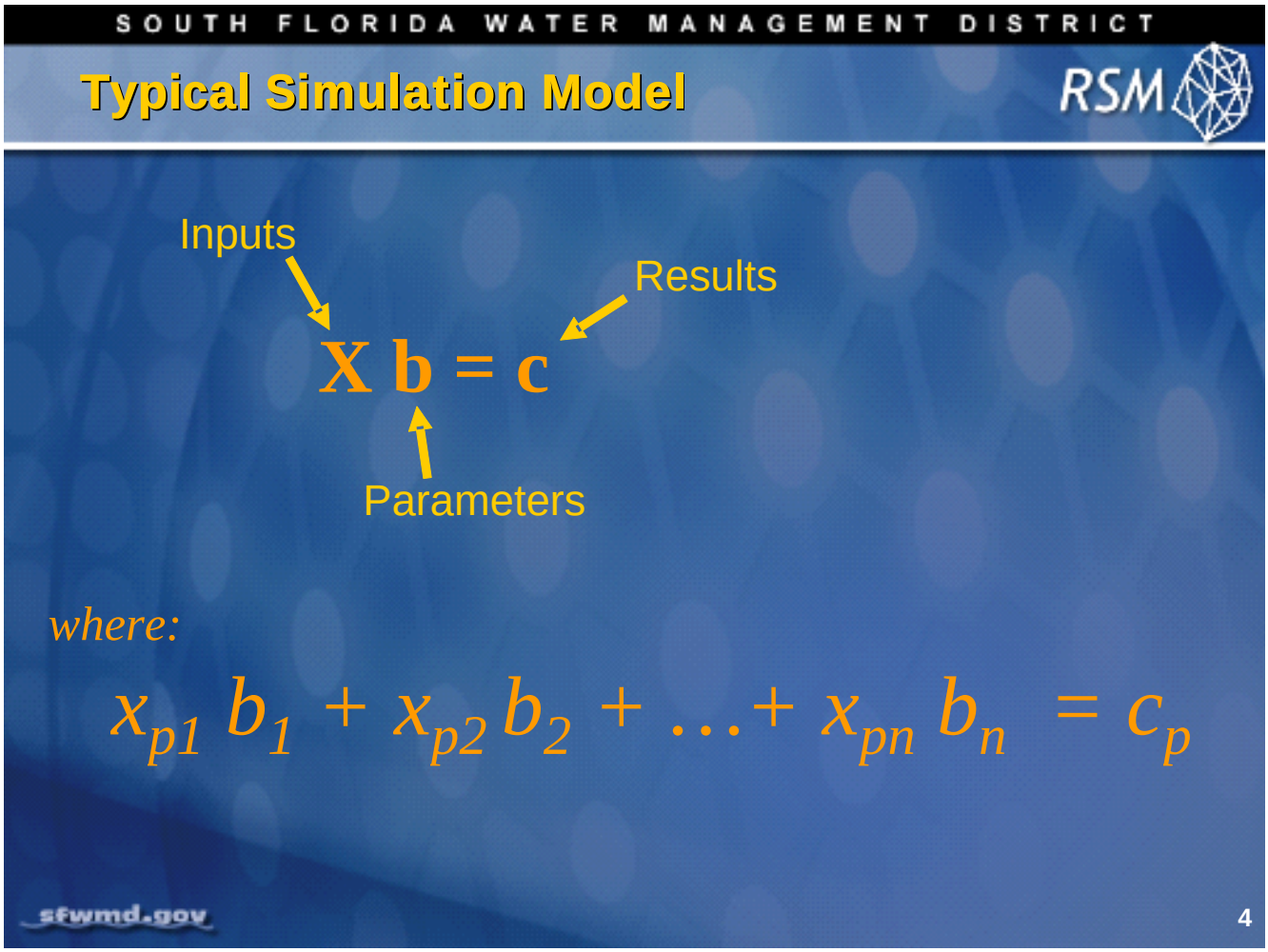


The Parameter Estimation (PEST) method is the basis of the calibration approach used for the Regional Simulation Model (RSM). The PEST software is freeware designed to run on Linux and Windows. Since the method is model independent, the RSM does not have to be modified to work with PEST.

PEST has advanced tools to improve calibration. And, it provides several methods for adjusting parameter values to minimize a user-defined objective function.

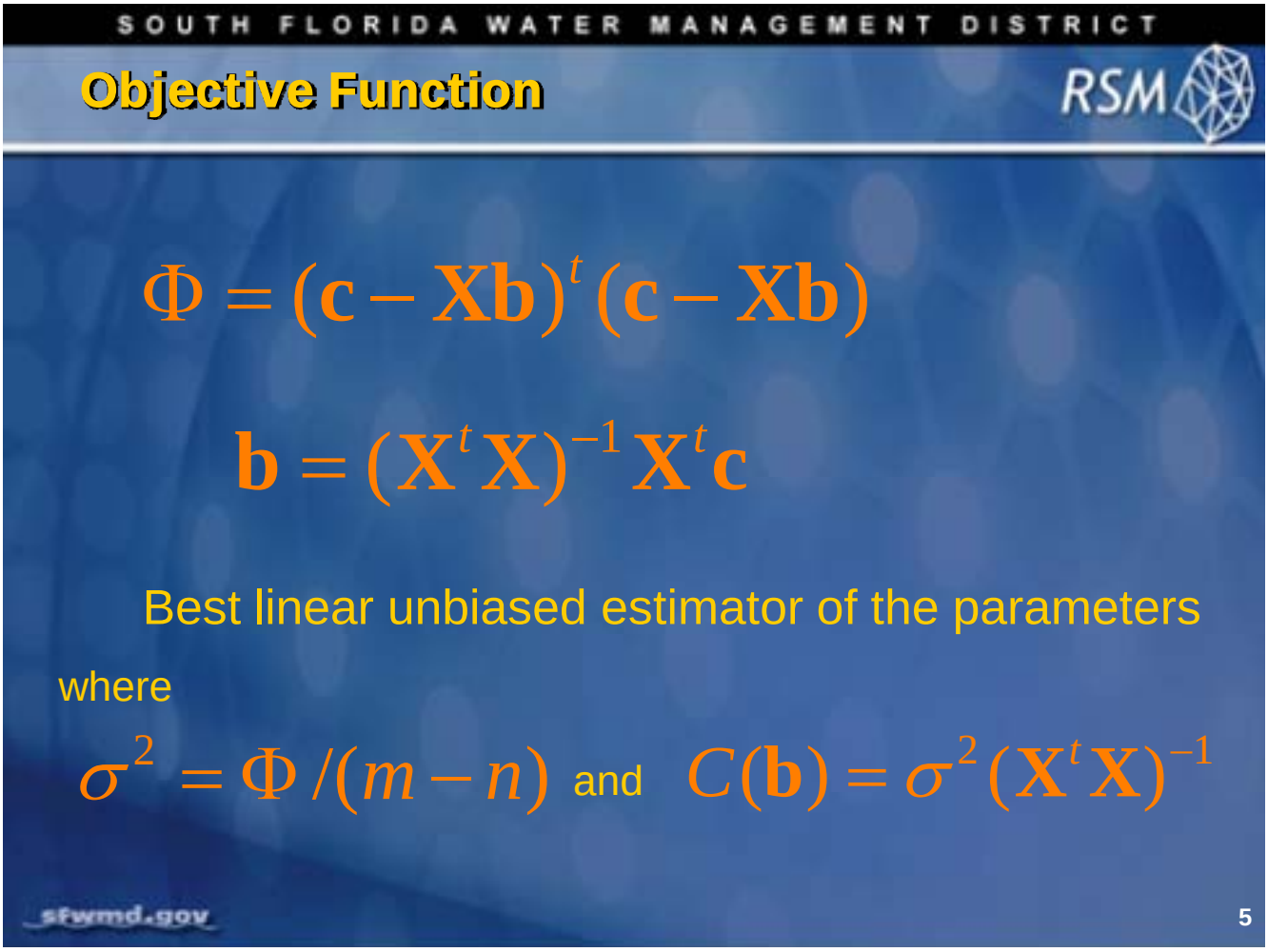


Key features of the PEST method are listed here. For a more in-depth review of each feature, the reader is encouraged to refer to the PEST User Manual.



In a typical model run, some set of input excitations (**X**), such as rainfall, potential evapotranspiration and inflow across the boundaries, and model parameters (**b**), are used to produce a set of results (**c**).

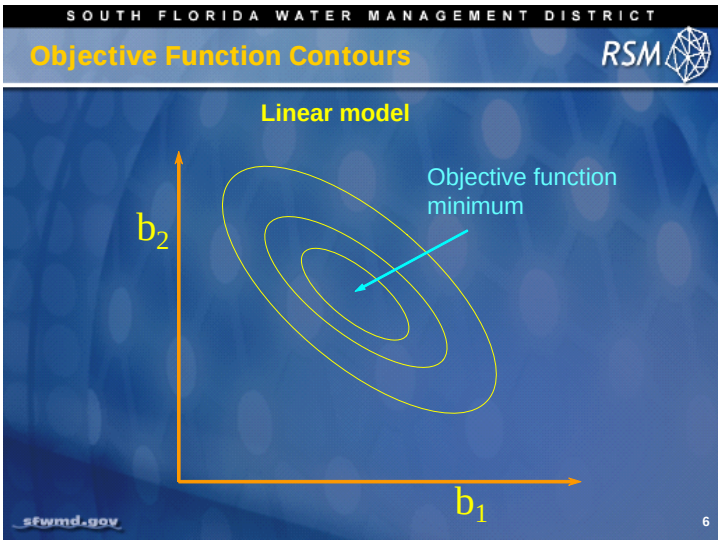
Conceptually, the inputs react with each parameter to produce the results.



The objective function is based on the best set of linear combinations of the parameters (**b**), as affected by the model inputs (**X**), compared to the observations (**c**),

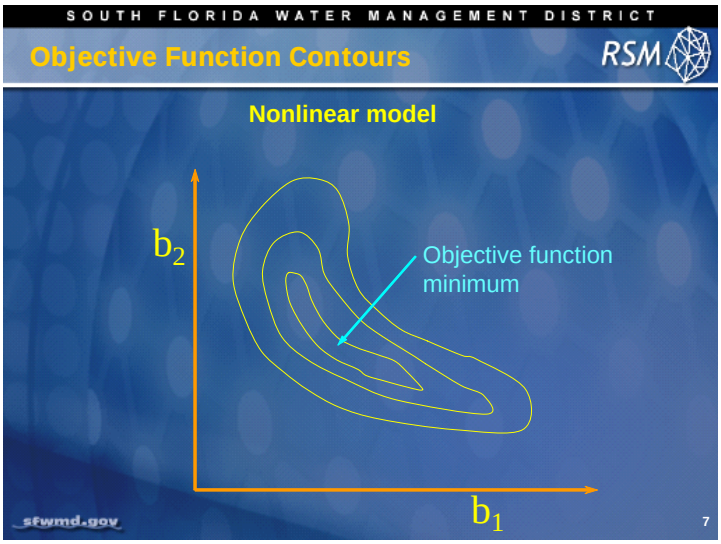
- where (m) is the number of observations
- and (n) is the number of parameters

The variance and covariance can be calculated from the objective function.



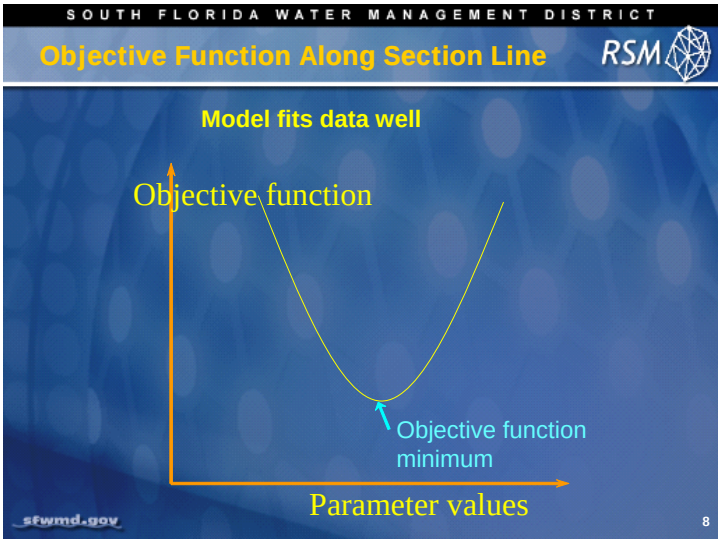
For a well-behaved linear model, the objective function has a local minimum with respect to the parameter values. The yellow ellipses are isopleths of constant value of the objective function.

In a generic sense, the objective function can be envisioned as the comparison between any two model parameters realizing that this represents **n**-dimensional parameter space.

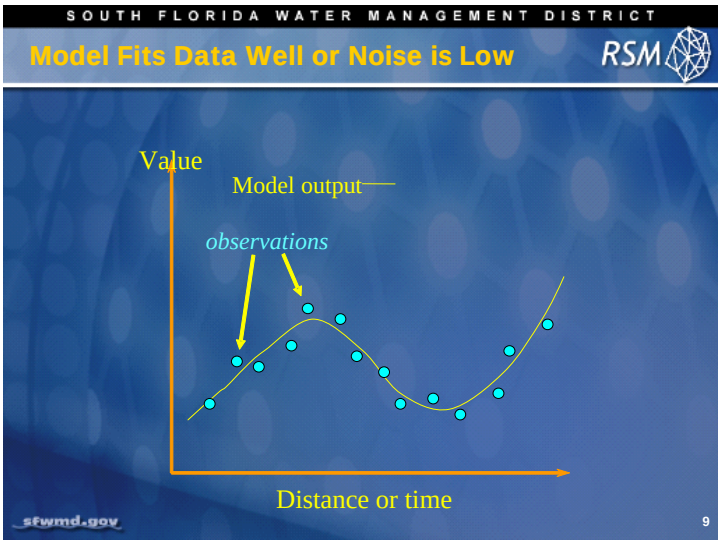


For a nonlinear model, the minimum objective function lies in a banana-shaped region (or, there may be more than one minimum).

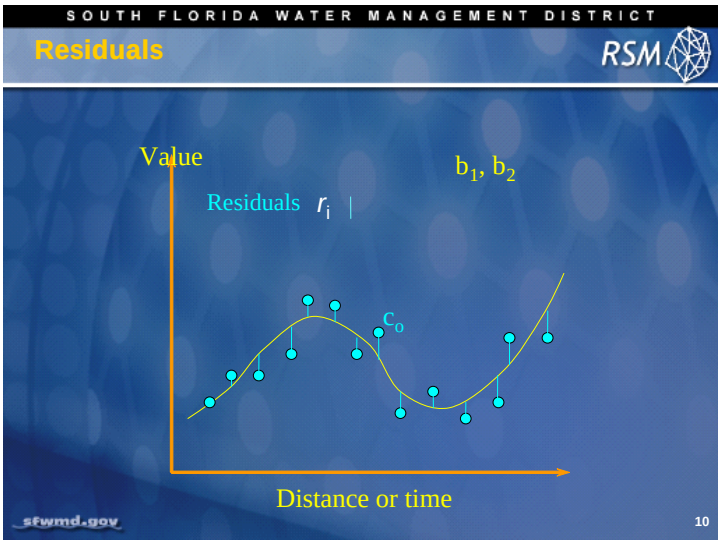
In general, it is best for the model to be more linear. Estimating the logs of certain parameters, rather than the parameters themselves, can improve linearity.



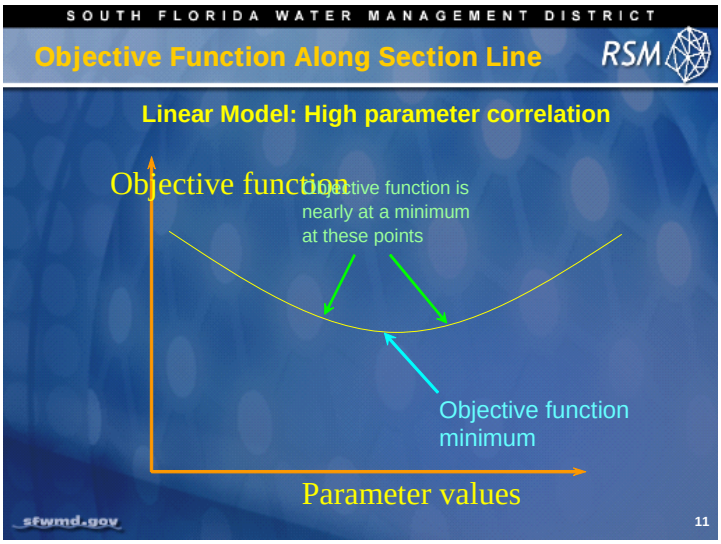
If the model fits the data well, the objective function will have a distinct minimum.



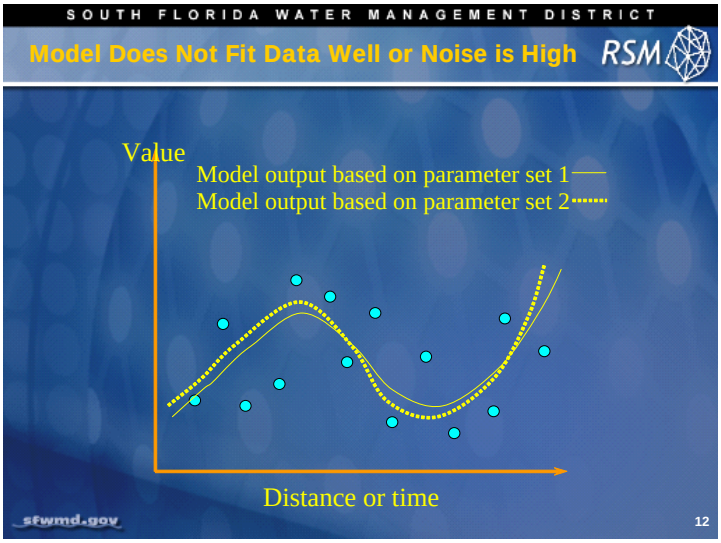
The model (shown above) fits the data well.



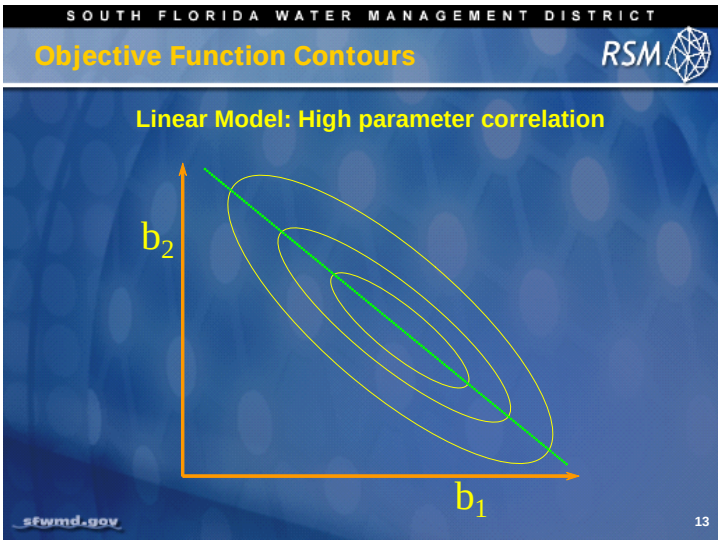
Where the model fits the data well, the residuals (the distance between the model and the observations) will be small.



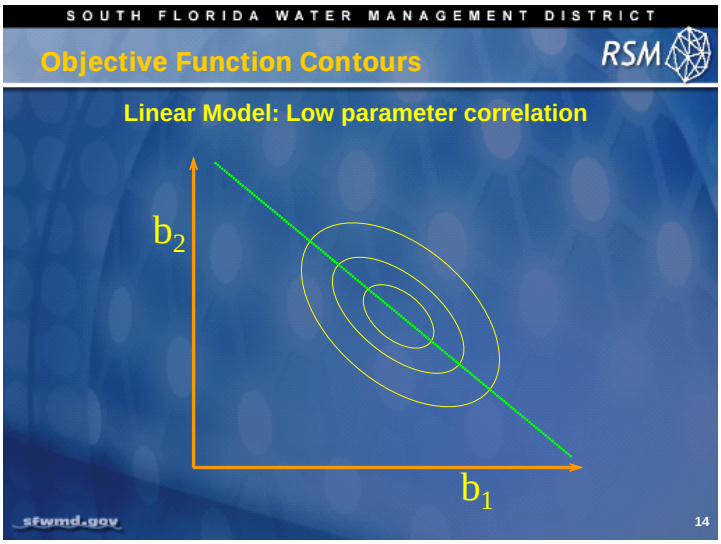
Where the model does not fit the data well, the objective function near the minimum will not be distinct and there is a strong likelihood that the model parameters are highly correlated.



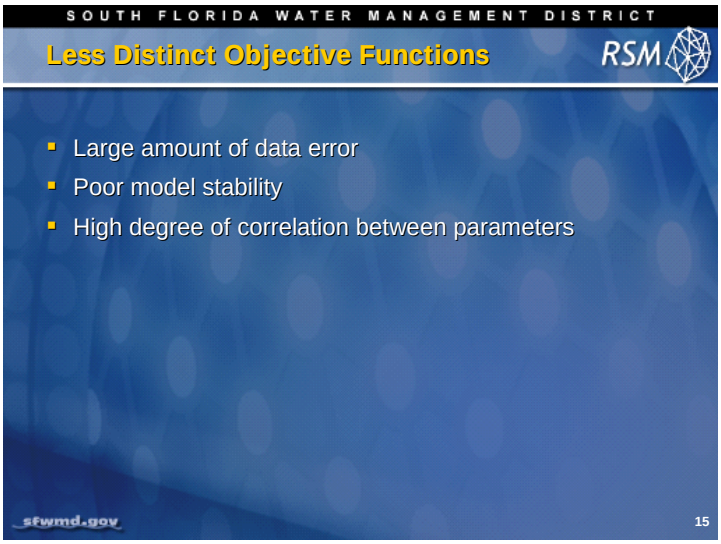
For the typical complex model, there will be more than one set of parameter values that fits the available data equally well.



If there is a high correlation between parameters, contours surrounding the minimum of the objective function will be elongated, indicating that a wide range of values of the correlated parameters will work.

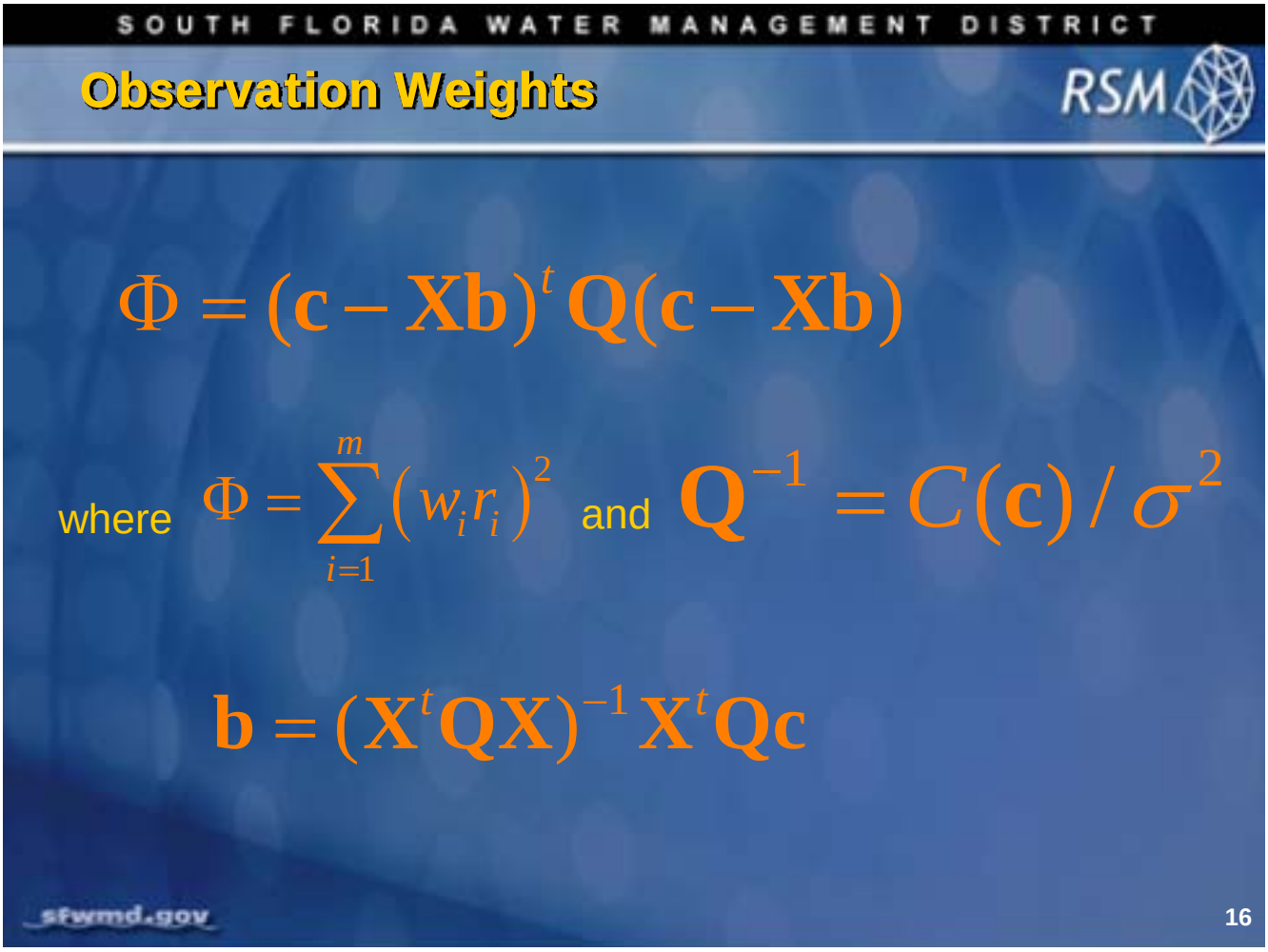


For a linear model with low parameter correlation, the minimum in the objective function will be more easily found.



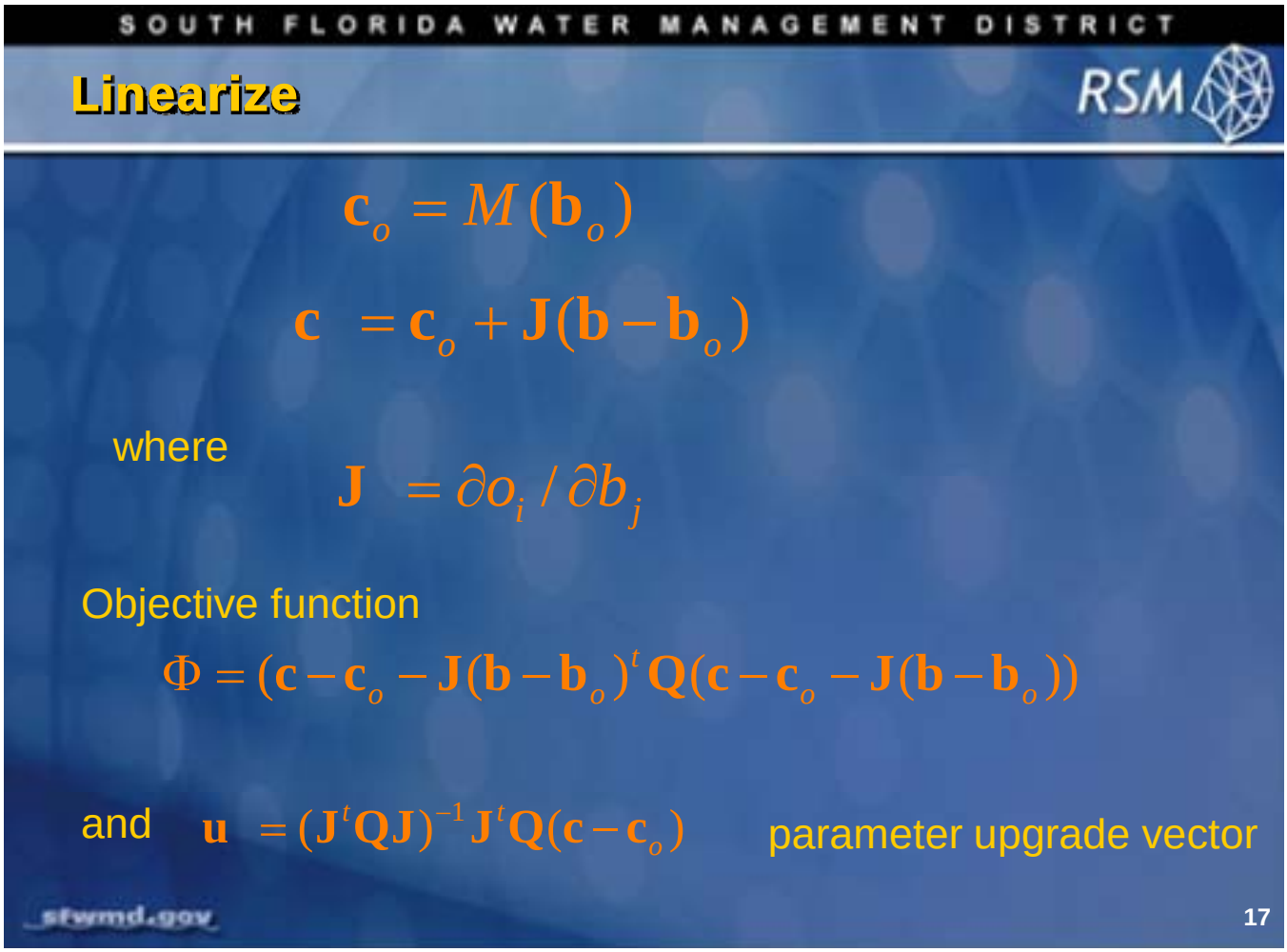
Finding the minimum of the objective function is limited if:

- There is a large amount of data error
- The model does not represent the data well
- There is a high degree of correlation among the parameters



The objective function can be better predicted if the residuals are given different weights in the analysis.

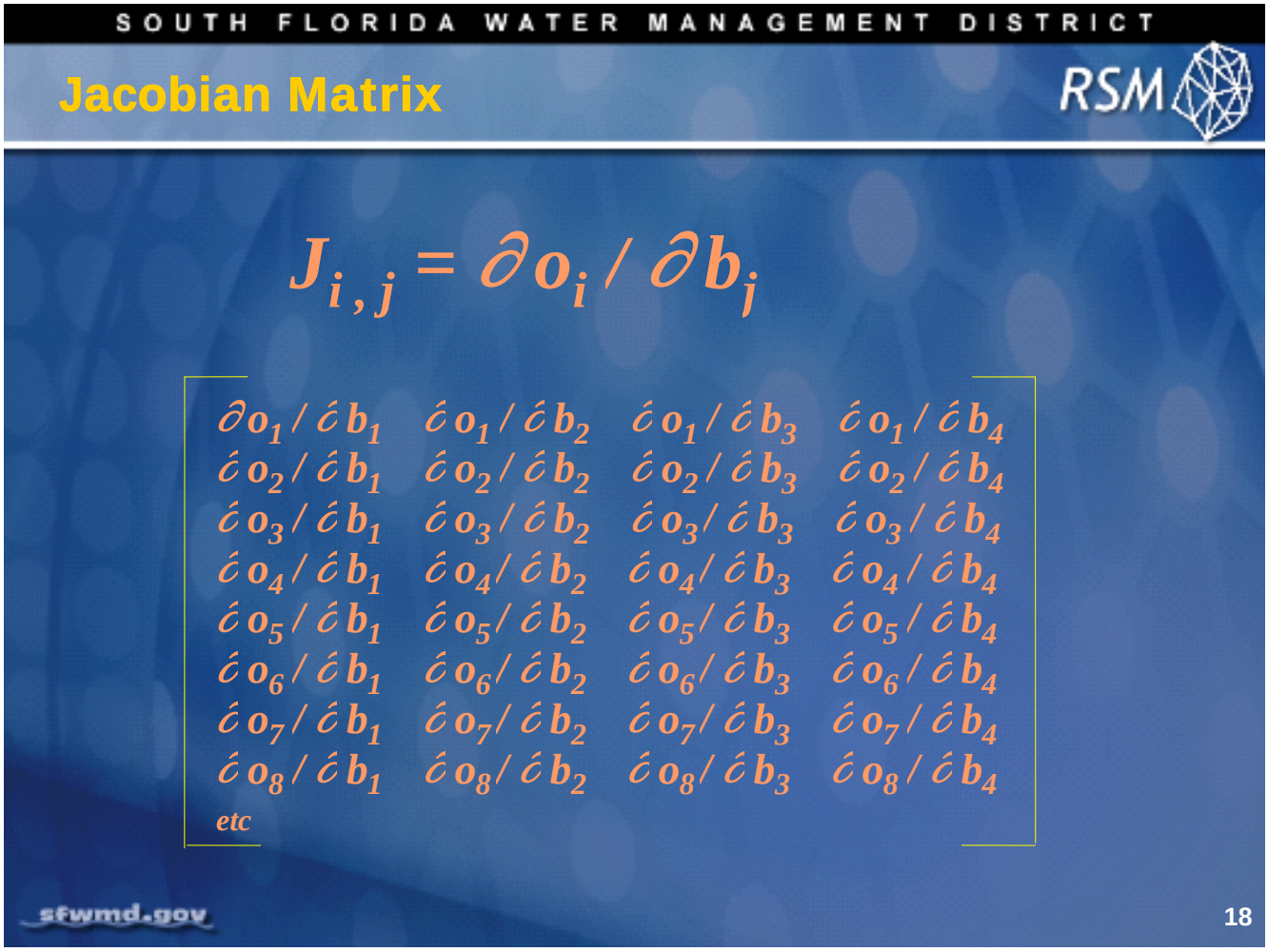
Q is a diagonal matrix of the squared observation weights. Weights are used where the observations used in the objective function have a wide range in values, such as including target flows and porosity. The weights effectively normalize the components of the objective function. It may be that some of the field data are more important to the model results than other data and thus receive greater weight.



Where the initial simulation results ($\mathbf{c}_o$) are some function (M) of an initial set of parameters ($\mathbf{b}_o$), then an improved simulation result ($\mathbf{c}$) is produced by multiplying the Jacobian ($\mathbf{J}$) with the difference between the old and new parameter values ($\mathbf{b}$-$\mathbf{b}_o$), and adding that to the old results.

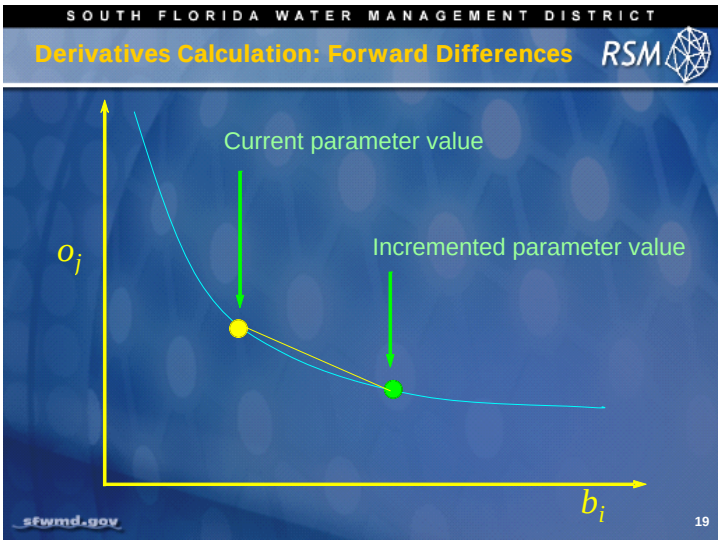
Using this relationship, the objective function can be modified to use the information available in the Jacobian.

A new variable ($\mathbf{u}$), the upgrade vector, can be calculated providing a means to estimate the next improvement in the parameters toward minimizing the objective function.

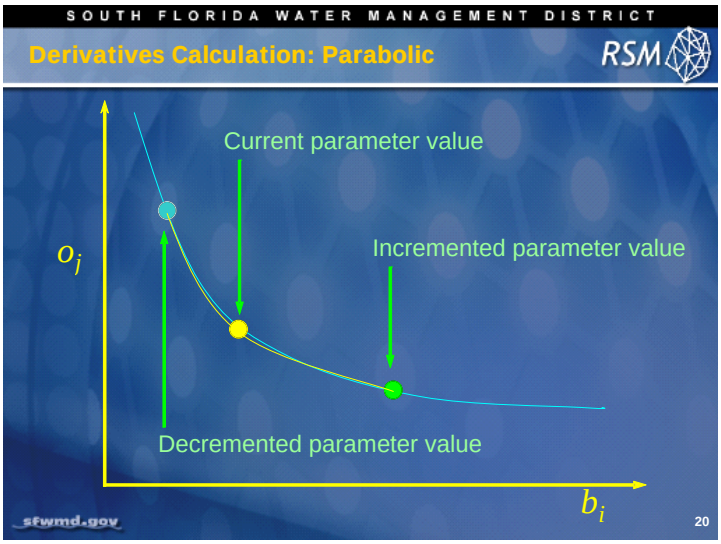


The Jacobian is the partial differential of the model results (observations) to changes in the model parameters. This is the sensitivity matrix.

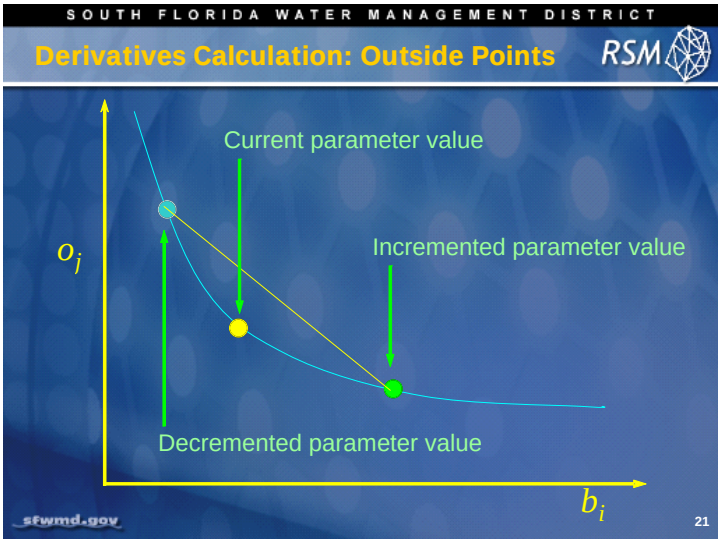
Creation of the Jacobian requires n+1 model runs.



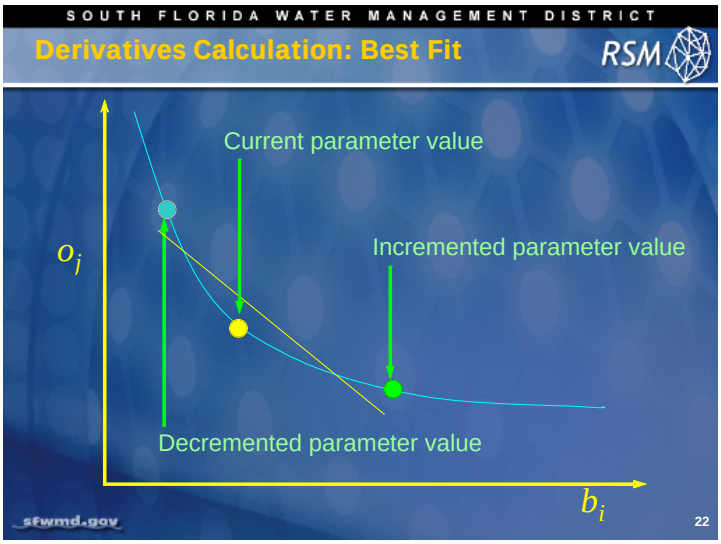
The Jacobian depends on the relationship between the model-generated observations and the parameter values. Forward differences can be used where the function is smooth.



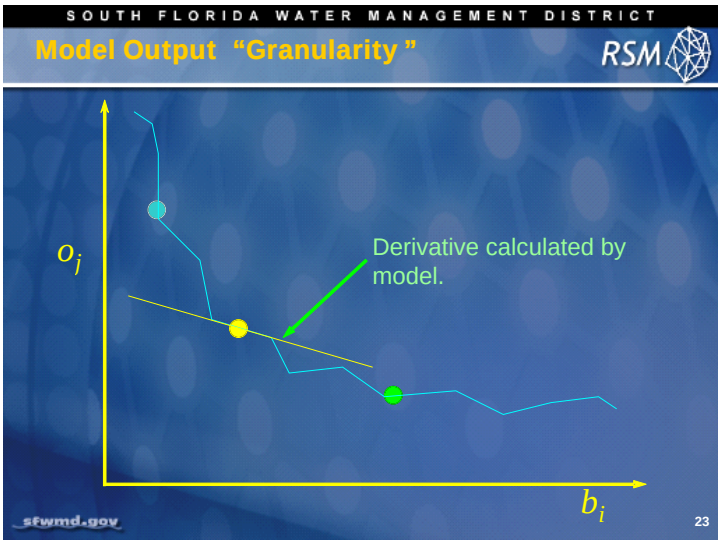
The parabolic approach may provide a better estimate but assumes a known function.



The decremented approach offers a better solution.

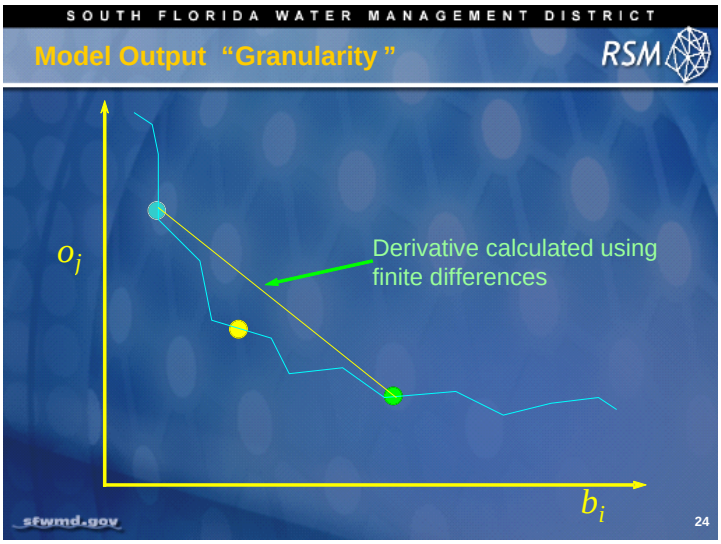


The best fit method may work better where the function is not smooth.

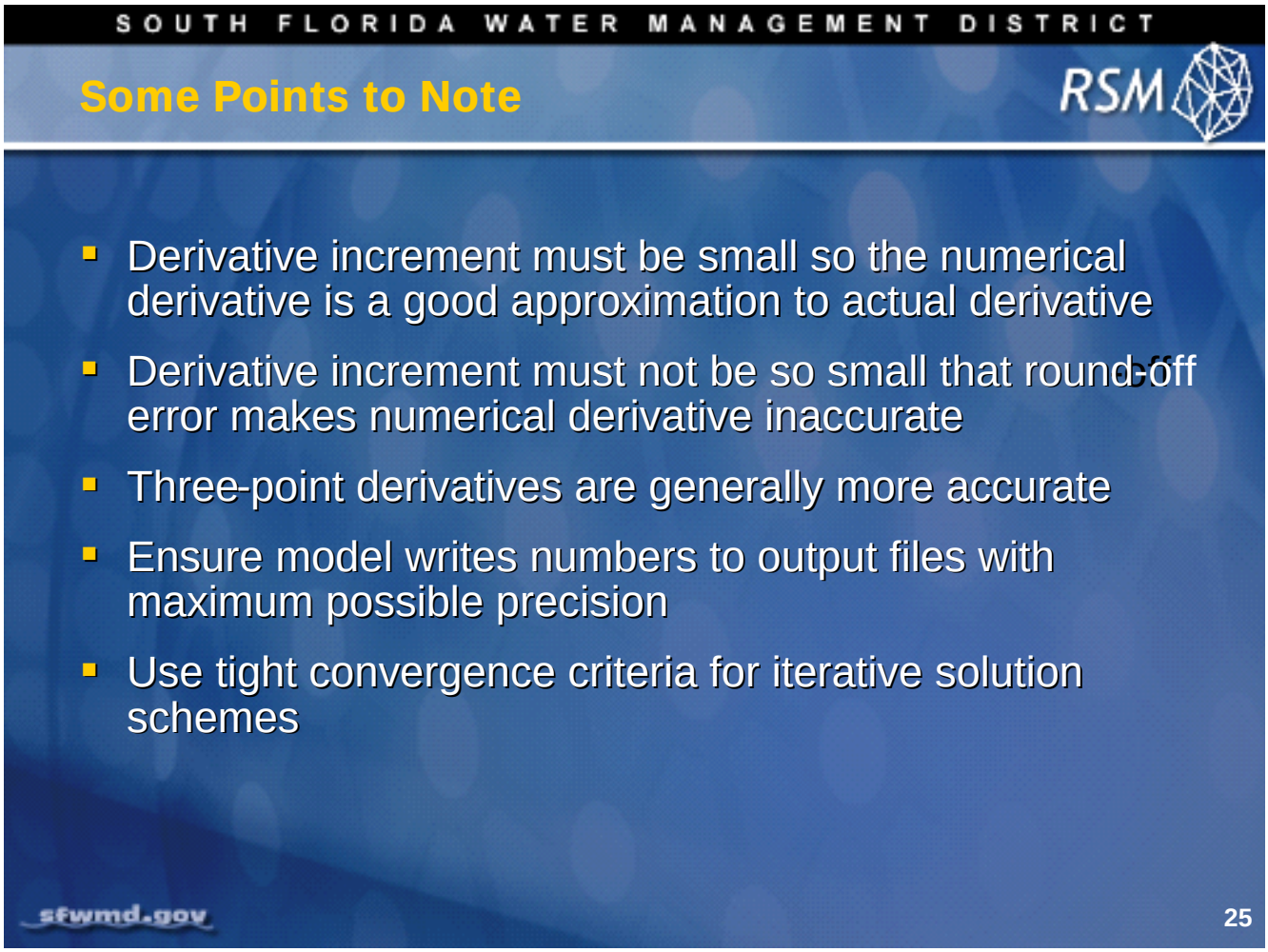


Where the model response is not smooth, local estimates of the derivatives may be misleading. Frequently the derivatives are not smooth functions. This is due to:

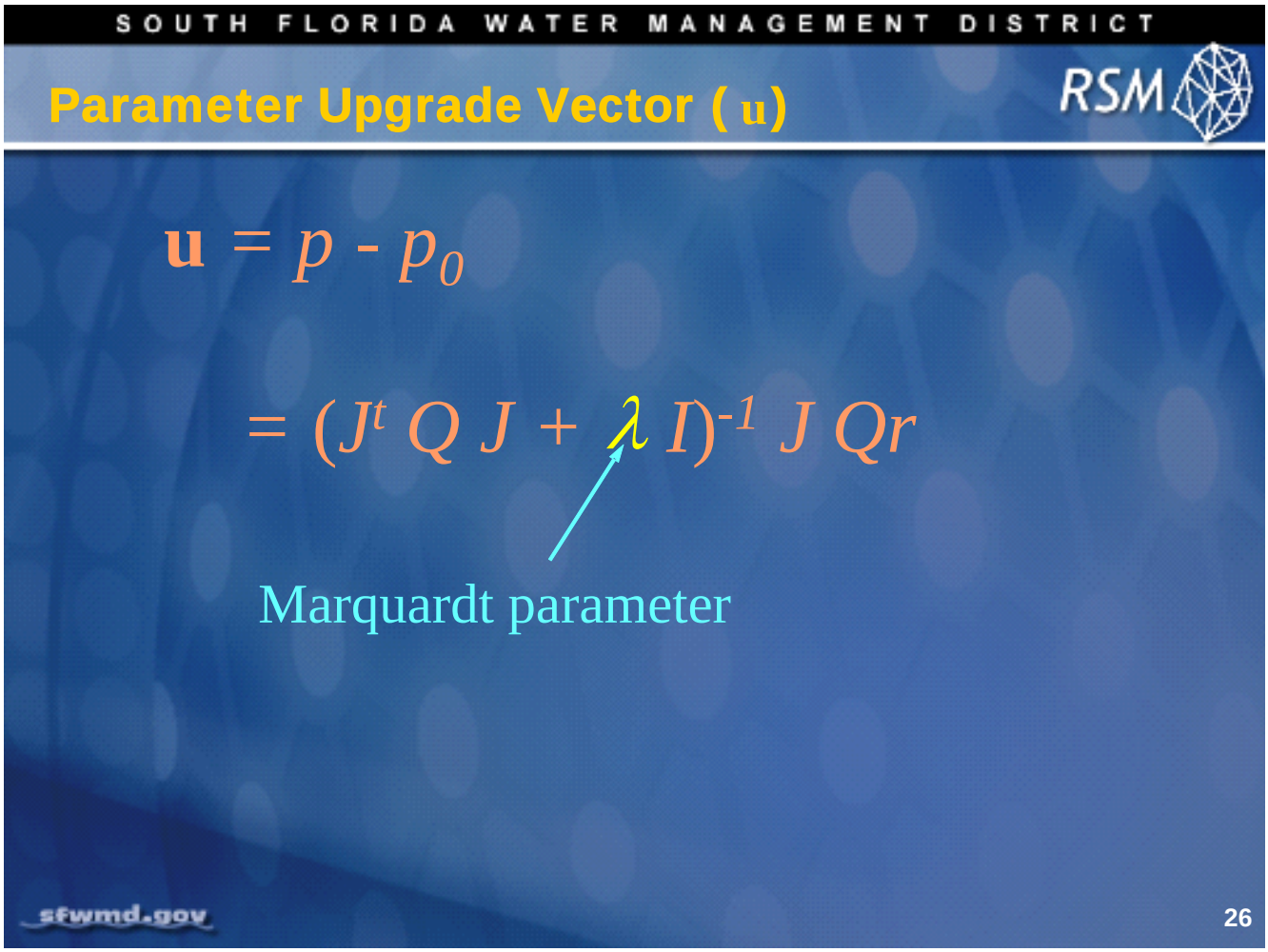
- The discretization of the domain
- Insufficiently complex model
- Non-contiguous processes in the model



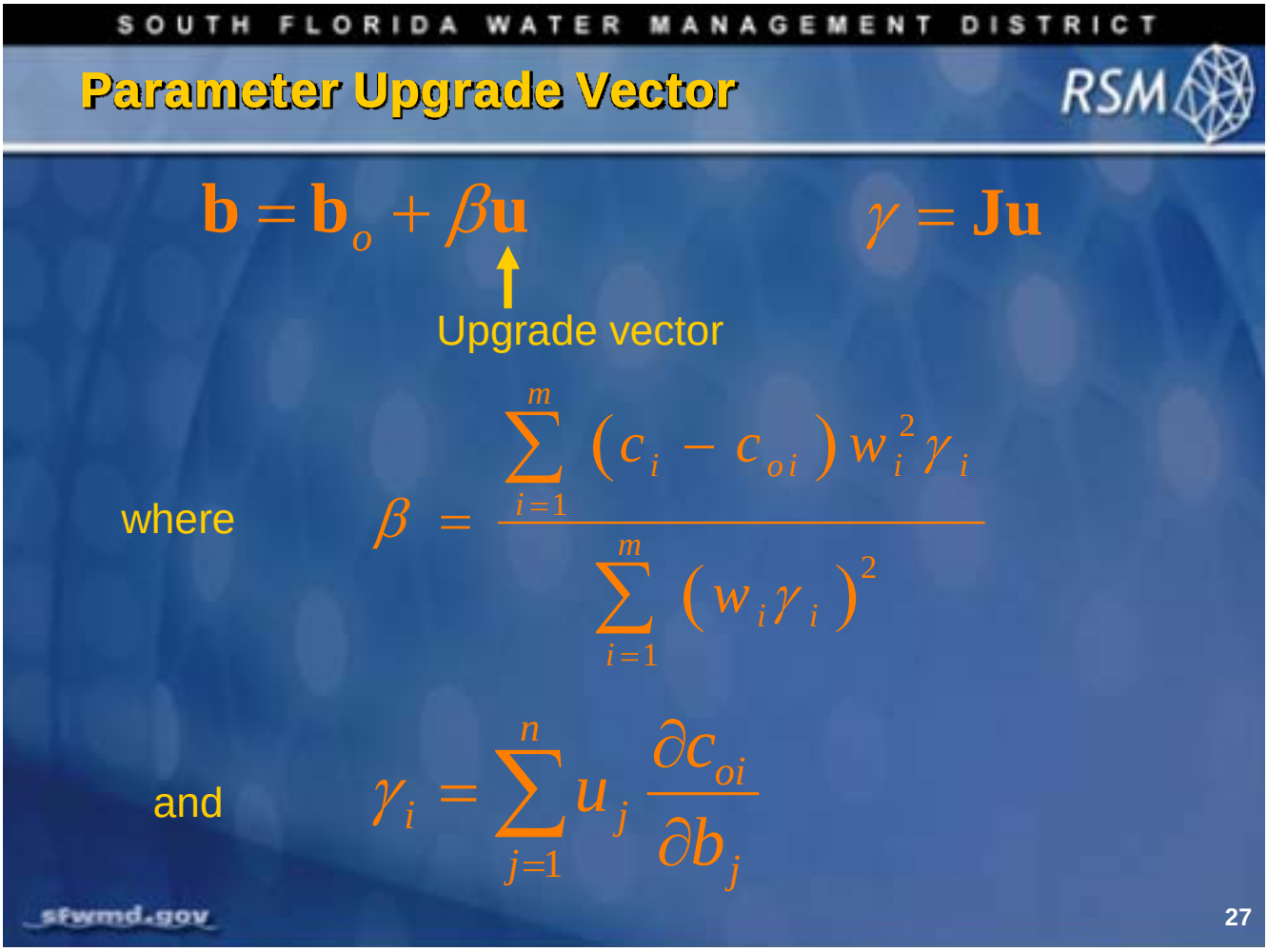
The three point method for calculating the Jacobian will improve the estimation but requires more model runs.


SOUTH FLORIDA WATER MANAGEMENT DISTRICT
Some Points to Note
RSM
Derivative increment must be small so the numerical derivative is a good approximation to actual derivative
Derivative increment must not be so small that round-off error makes numerical derivative inaccurate
Three-point derivatives are generally more accurate
Ensure model writes numbers to output files with maximum possible precision
Use tight convergence criteria for iterative solution schemes
sfwmd.gov
25

[No notes]



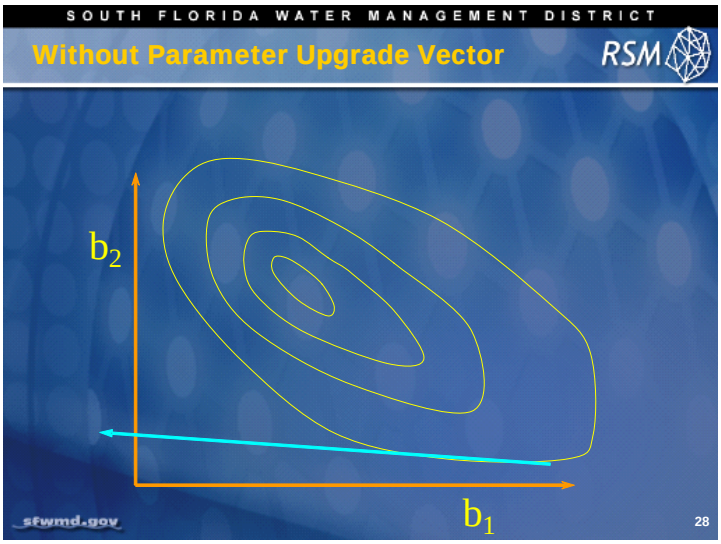
The parameter upgrade vector (**u**) can be improved by adding the Marquardt parameter, which increases the rate at which the objective function approaches the minimum. PEST starts with a user-supplied value of lambda and adjusts iteratively to improve the objective function.



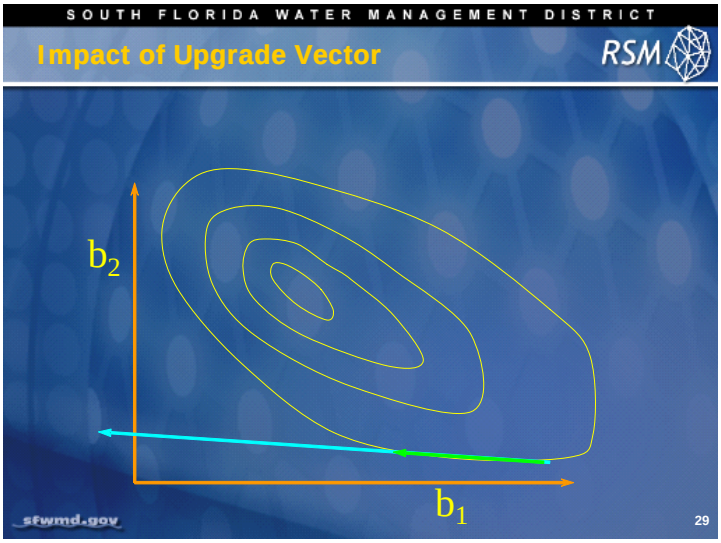
The optimal magnitude of the upgrade vector is determined by **β** which is a function of:

- The residuals ($c_i - c_{oi}$)
- The weights squared
- And γ

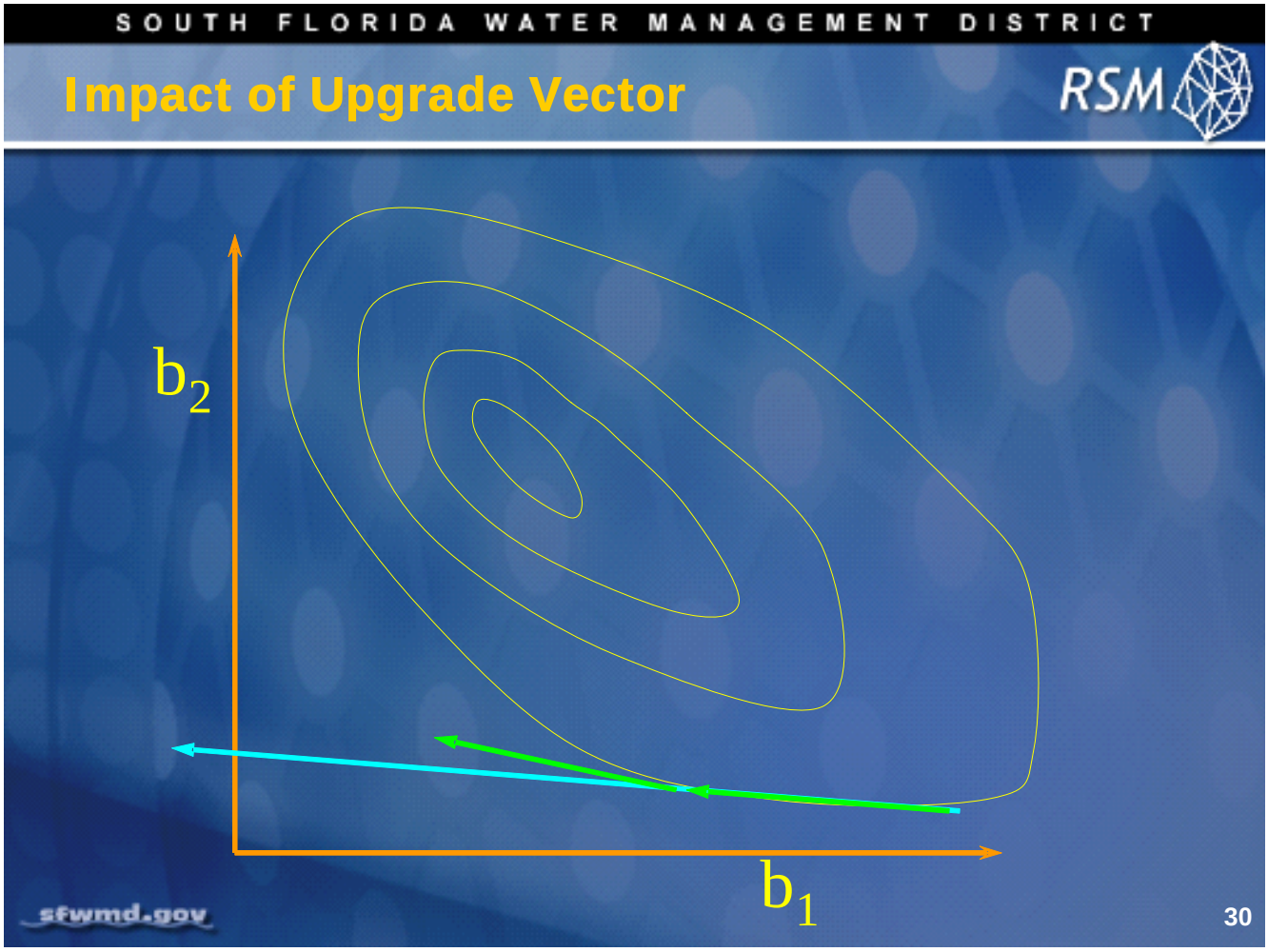
Lambda is a function of the upgrade parameter and the Jacobian.



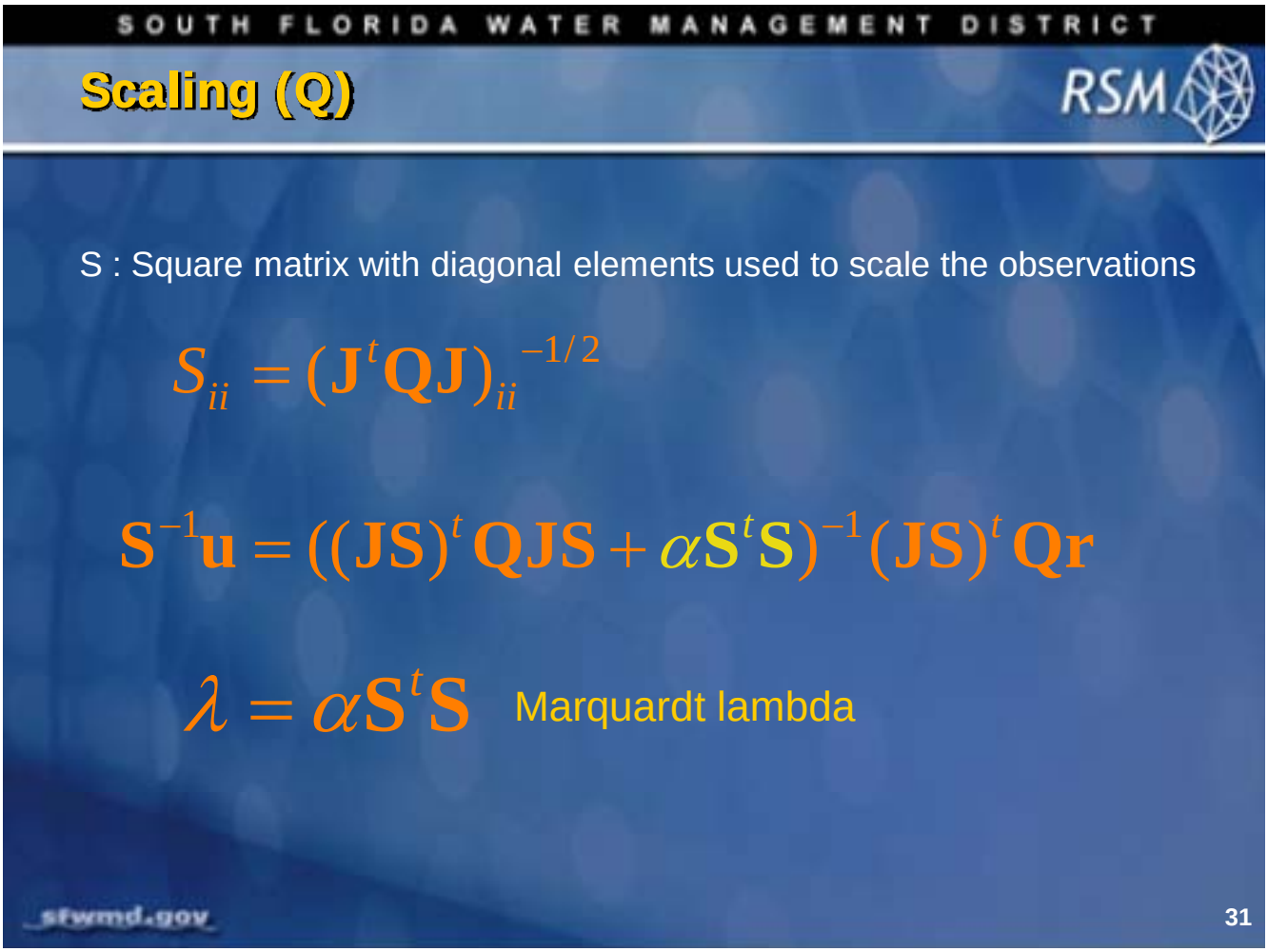
Without the parameter upgrade vector, the parameter estimates will overshoot the optimum solution leading to very slow convergence.



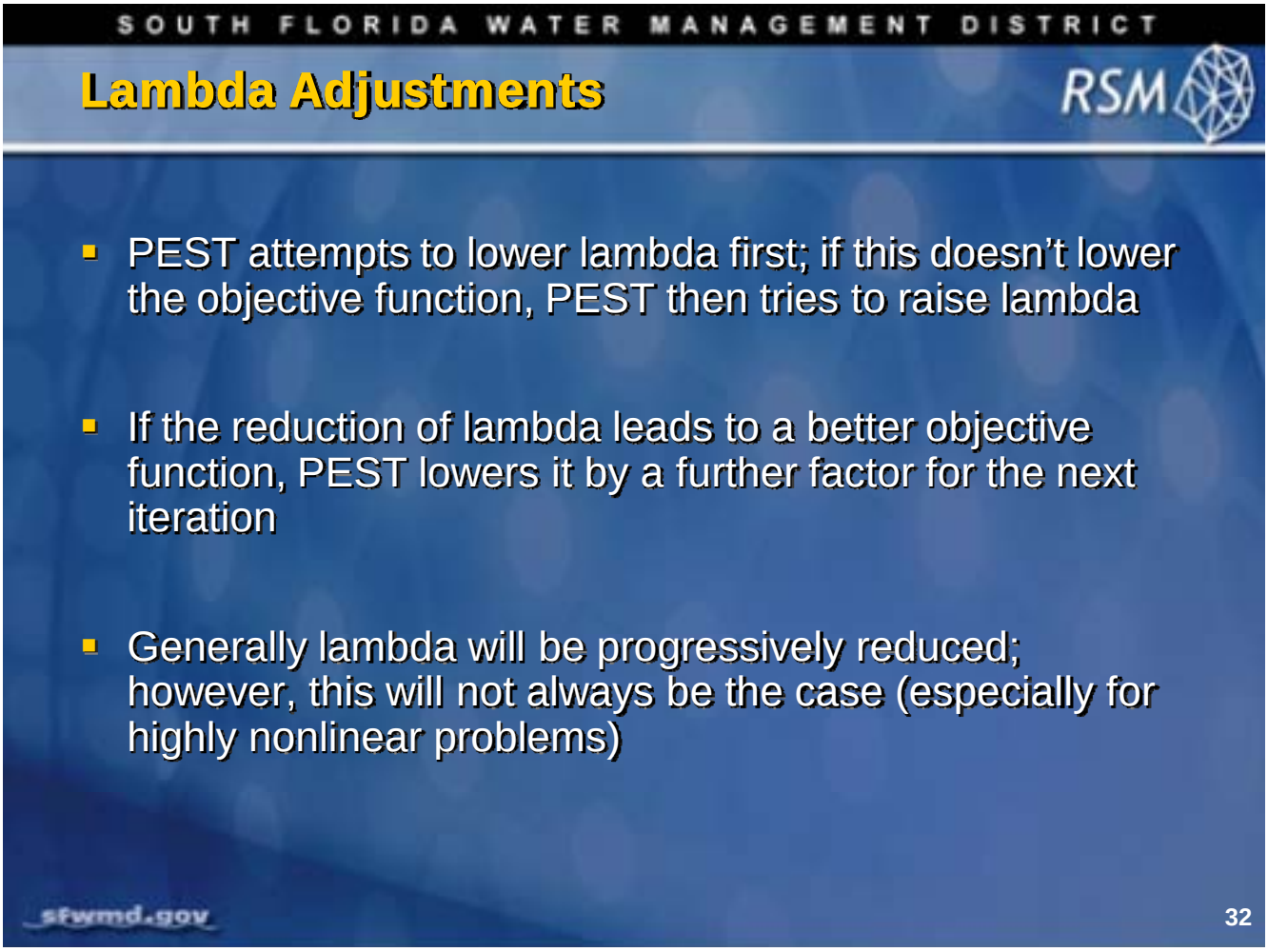
In the case where parameter b_1 is very insensitive, the parameter estimation can miss the optimum solution. The upgrade vector is used to impose a change limit.



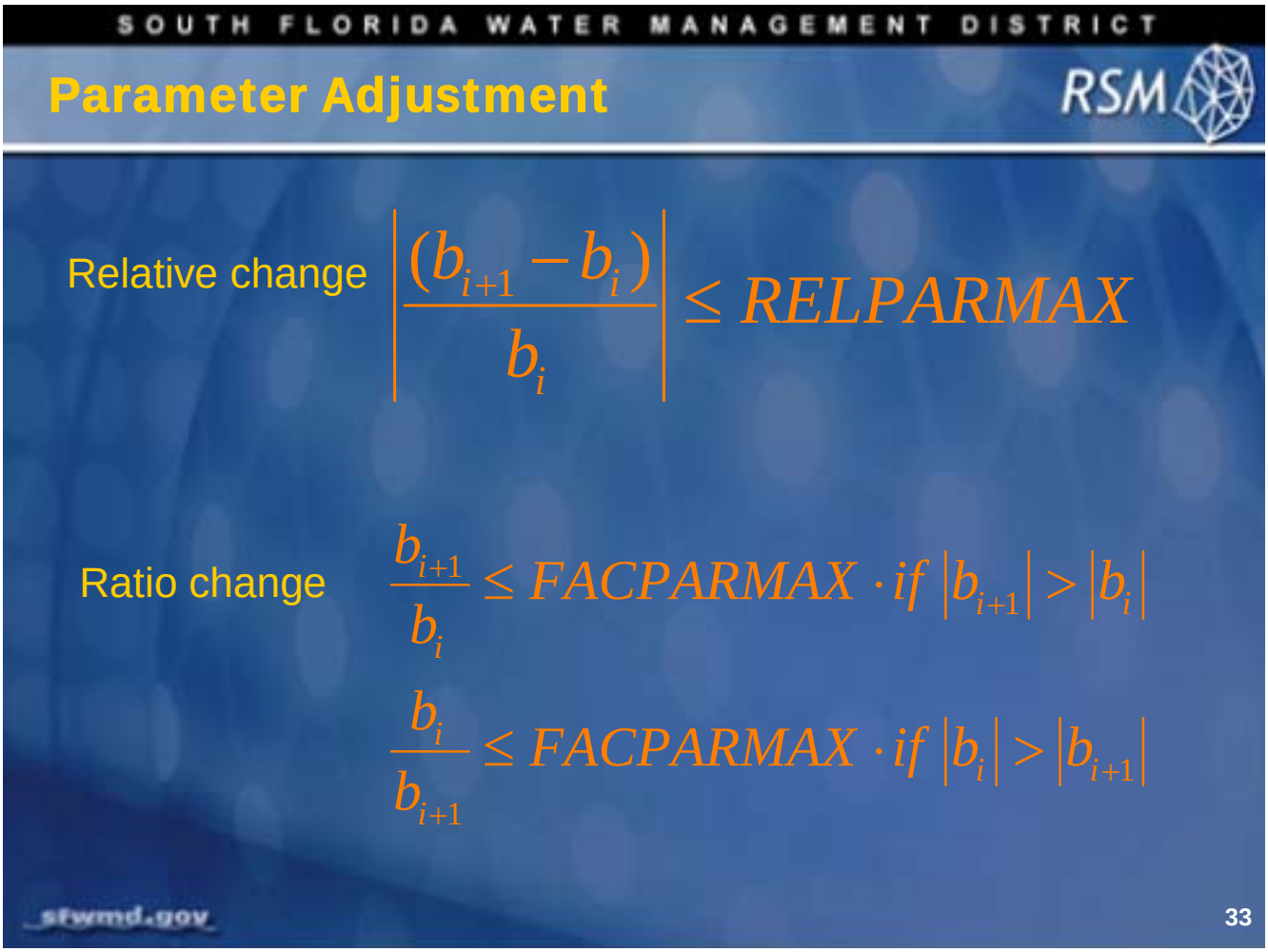
If parameter b_1 is very insensitive, the next upgrade vector with change limit imposed will more closely converge on the optimum solution.



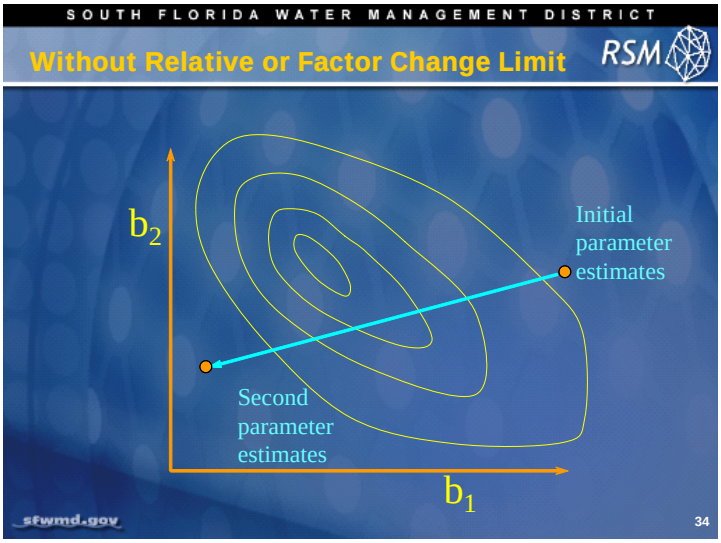
The lambda can be adjusted for different types of parameters and observations that have a wide range in magnitude. This causes elements of the Jacobian to have a wide range in magnitude leading to round-off errors. Introduction of the scaling matrix (**S**), to scale lambda, reduces the occurrences of errors.



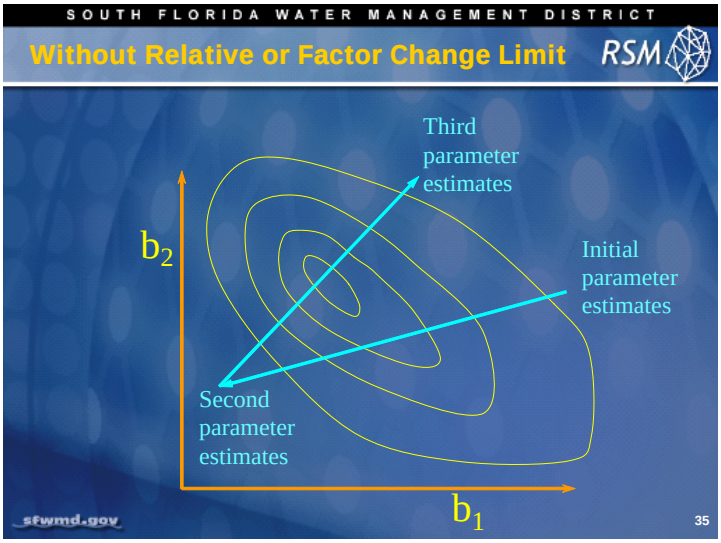
The PEST program iterates to find the optimum solution by adjusting the lambda value.



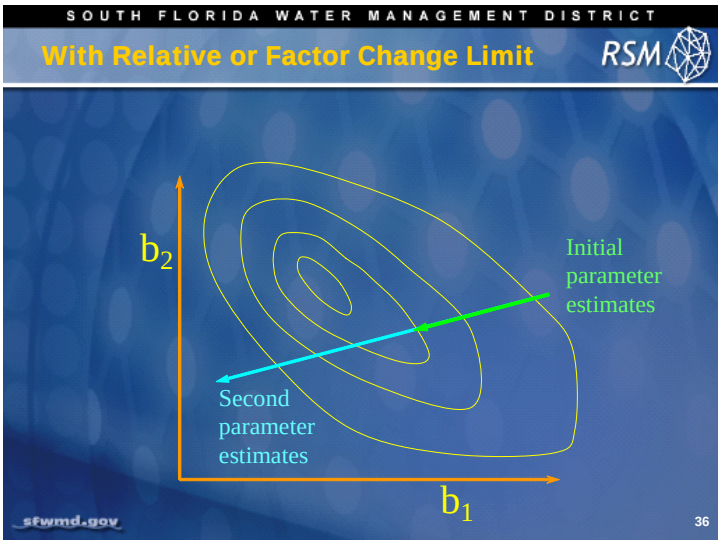
There are two methods for estimating successive parameter value changes, relative change and ratio change. These methods limit the extent to which PEST can automatically change the parameters. The Ratio change is more suitable to nonlinear parameters.



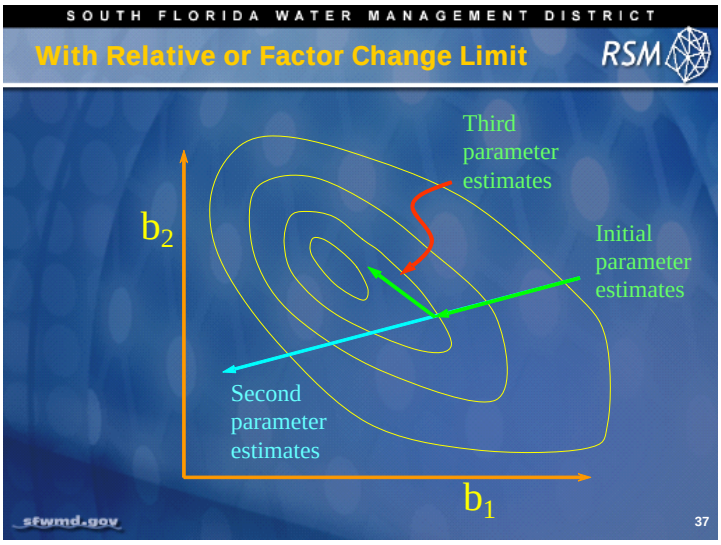
When applying PEST to the optimization problem, without the Marquardt parameter, the parameter estimates are likely to overshoot the minimum.



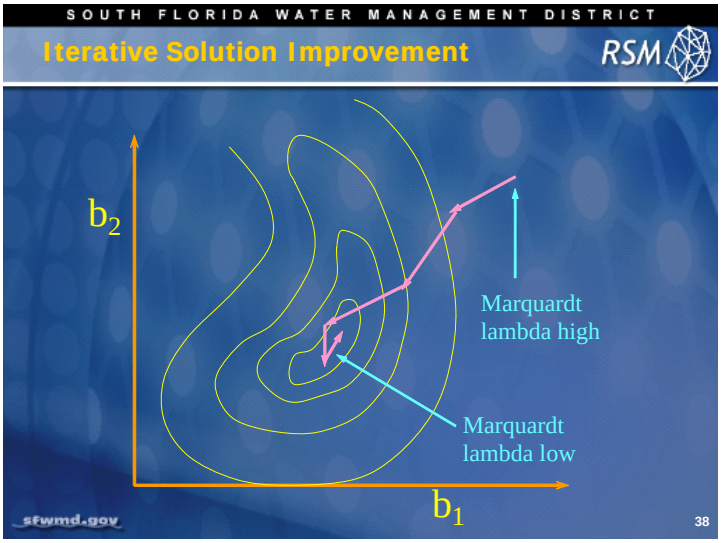
Additional parameter estimates may not improve results.



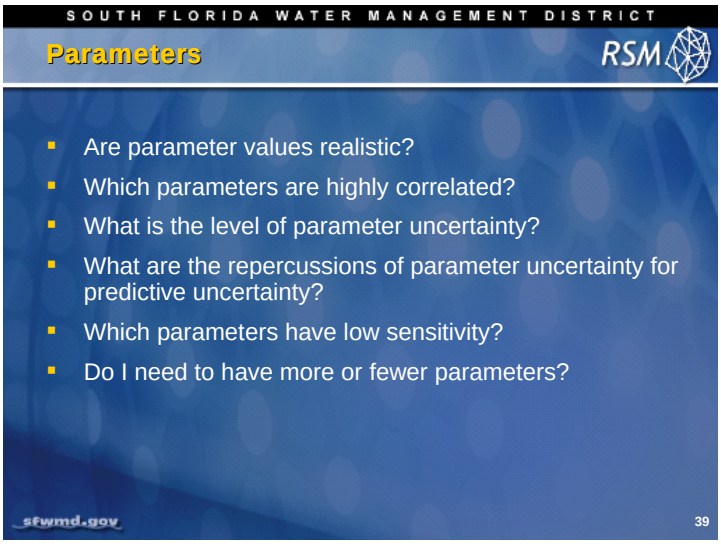
With the Marquardt parameter, the second parameter estimates more closely approach the objective function minimum.



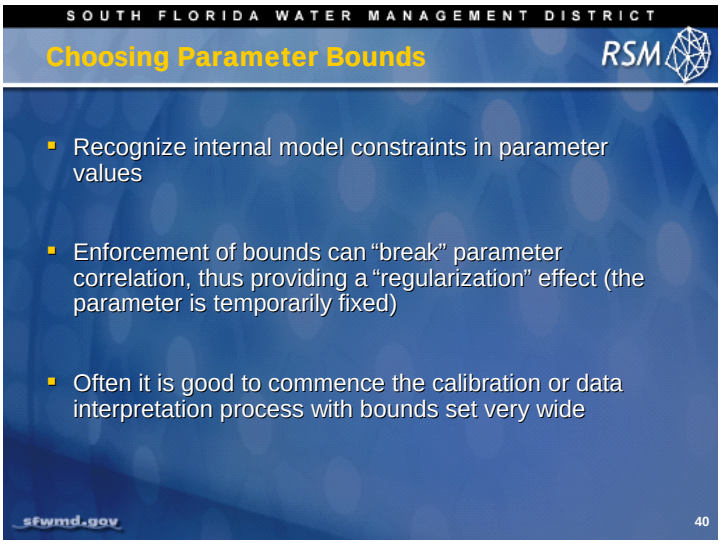
With the Marquardt parameter, the third parameter estimates and subsequent estimates more closely approach the objective function minimum.



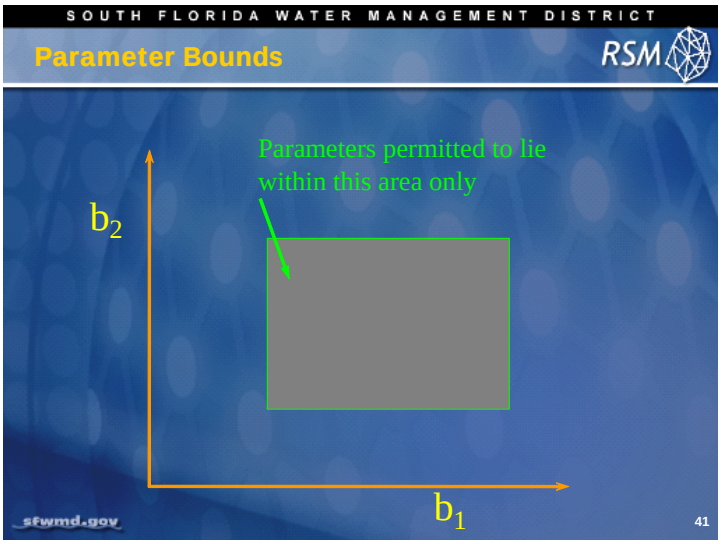
The value of lambda is modified during the optimization process, starting high and becoming smaller.



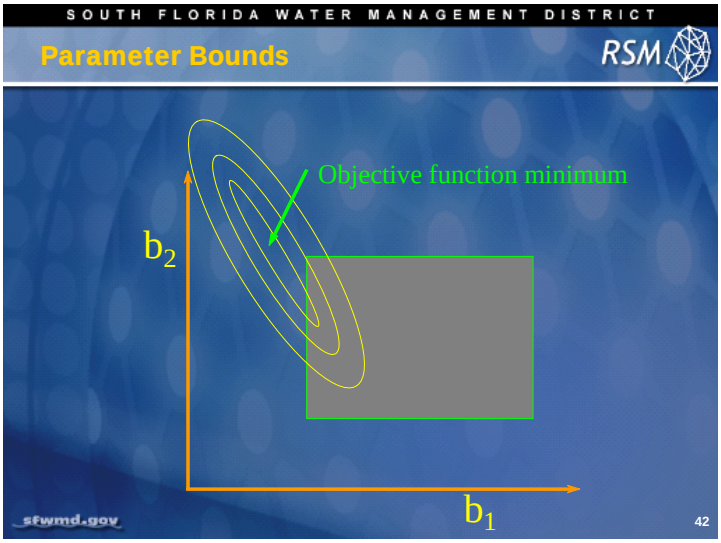
Evaluation of the model parameters and selection of the appropriate parameters is an important step when using PEST for model calibration. These are the critical issues that need to be addressed in parameter selection.



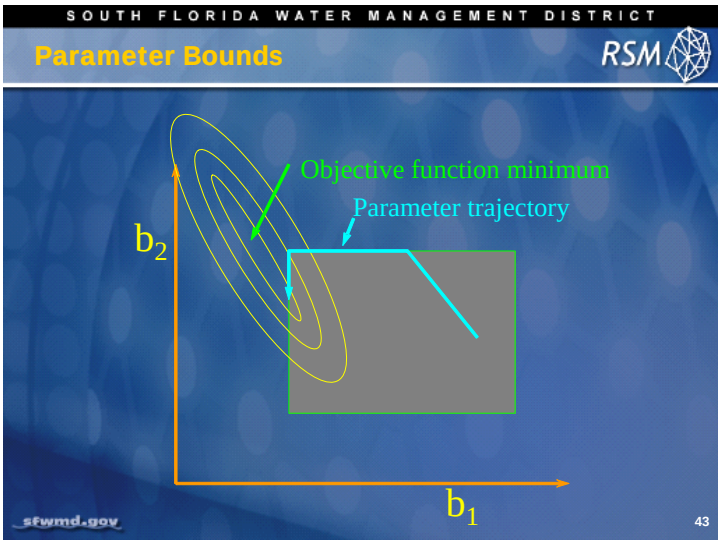
Since PEST adjusts the parameter values automatically, it is important to set the bounds of the allowable values for each parameter. In addition to recognizing physical bounds on parameter values, it is important to note that bounds may conflict with the parameter upgrading process.



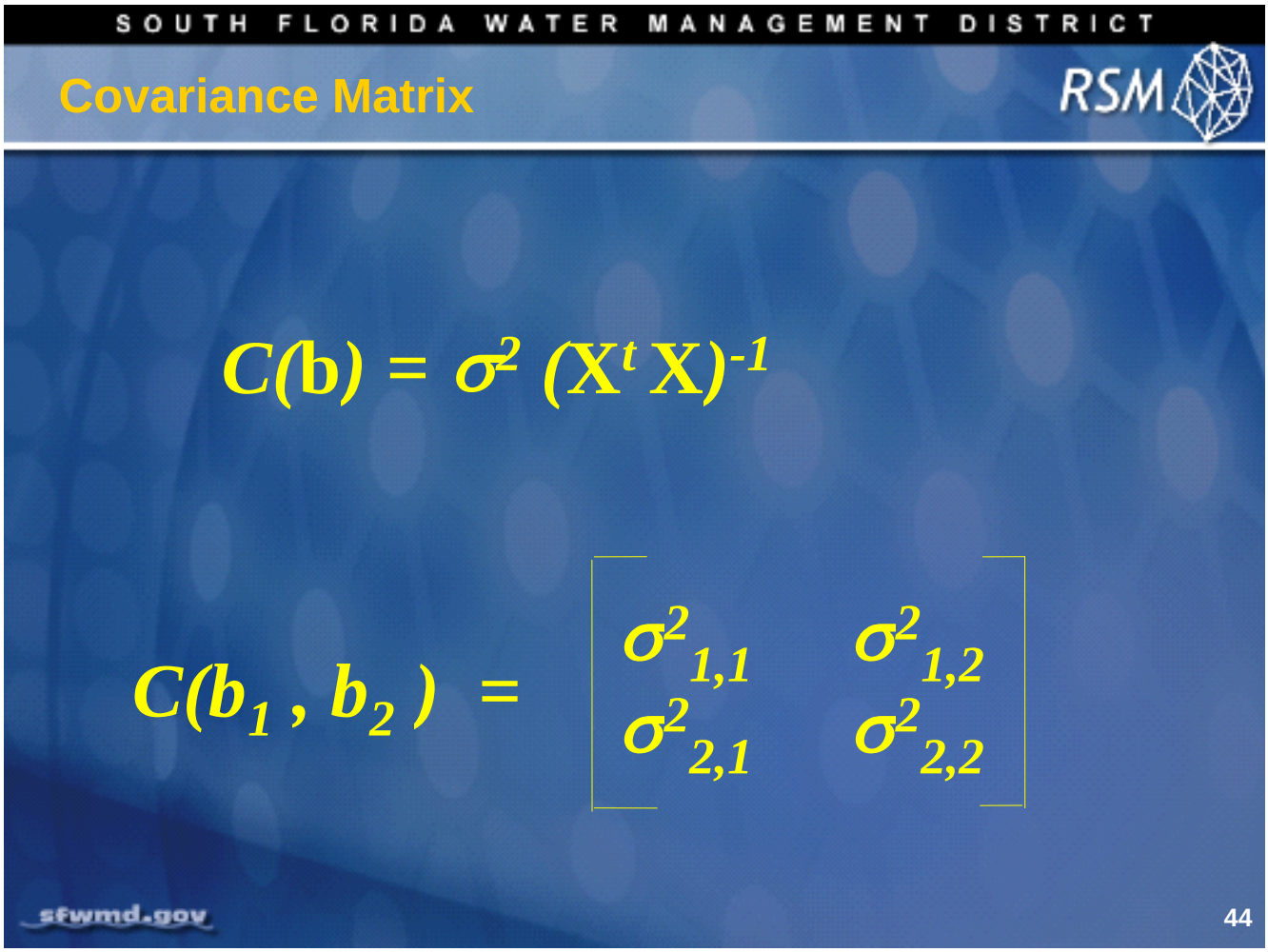
Parameter bounds can be used to limit automated PEST parameter adjustments to the range of physically meaningful values.



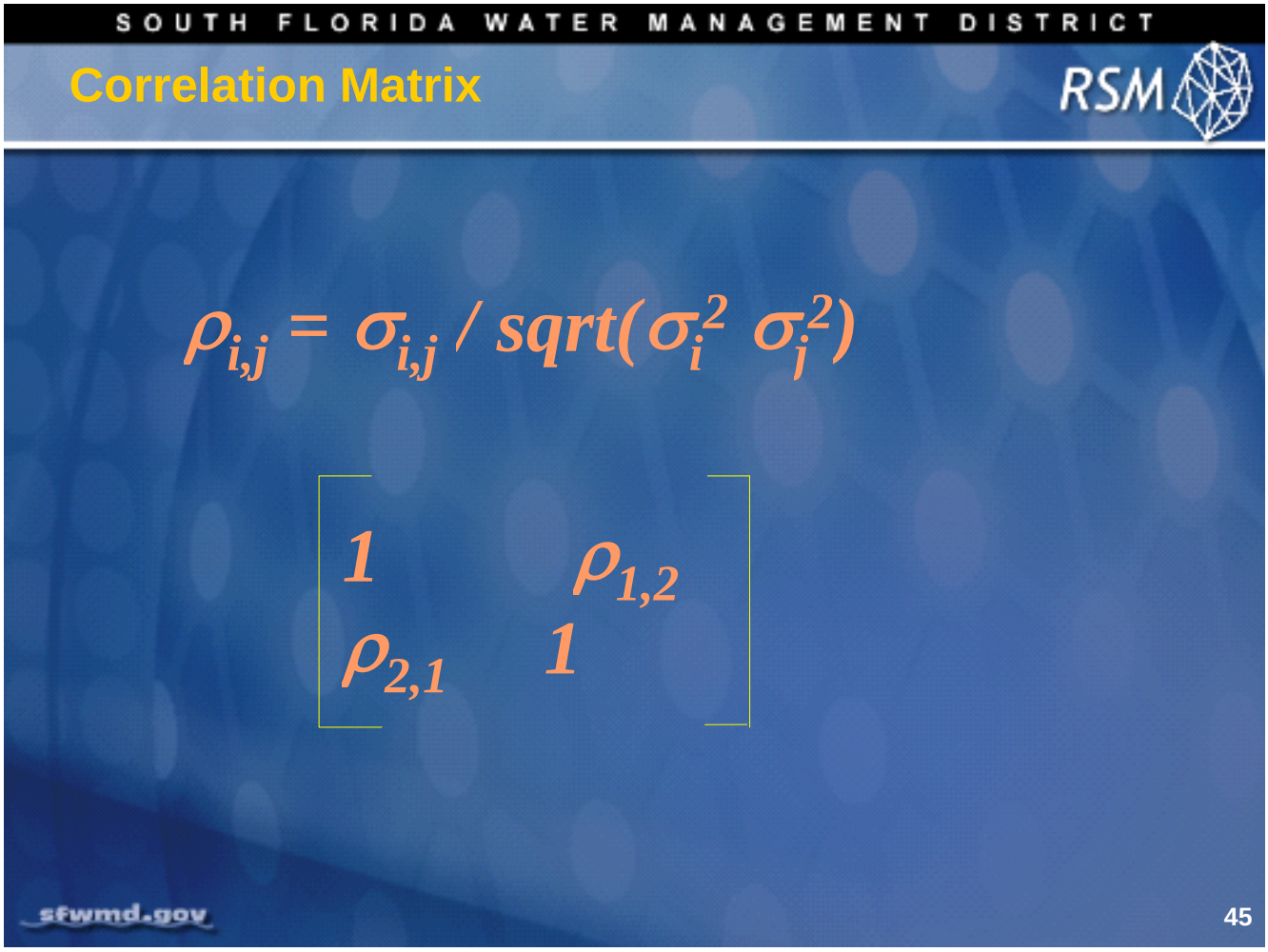
When the parameter bounds are set, they may not include the objective function minimum.



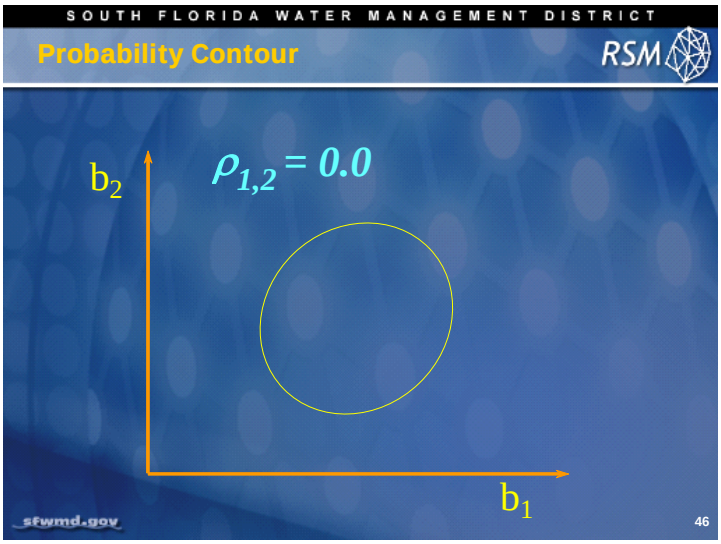
PEST will allow the parameter adjustments to proceed toward a objective function minimum subject to the parameter constraints. The constraints may result in a less than optimum solution.



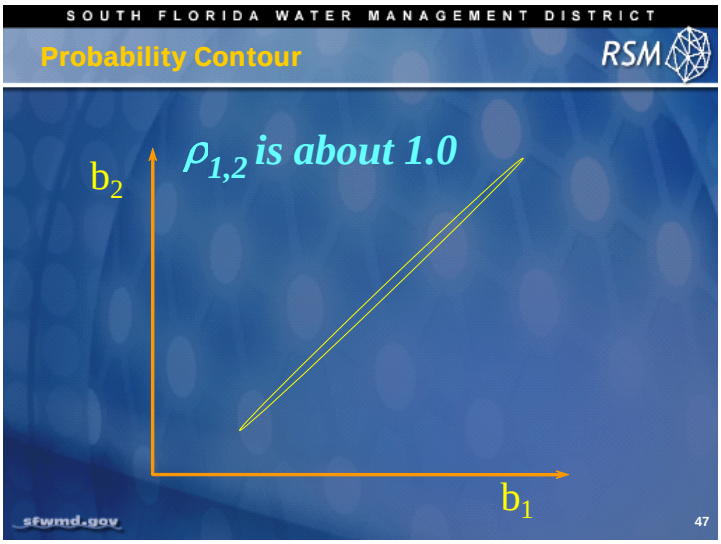
Covariance of the parameters is important as it determines how well PEST will converge to the optimal set of parameter values.



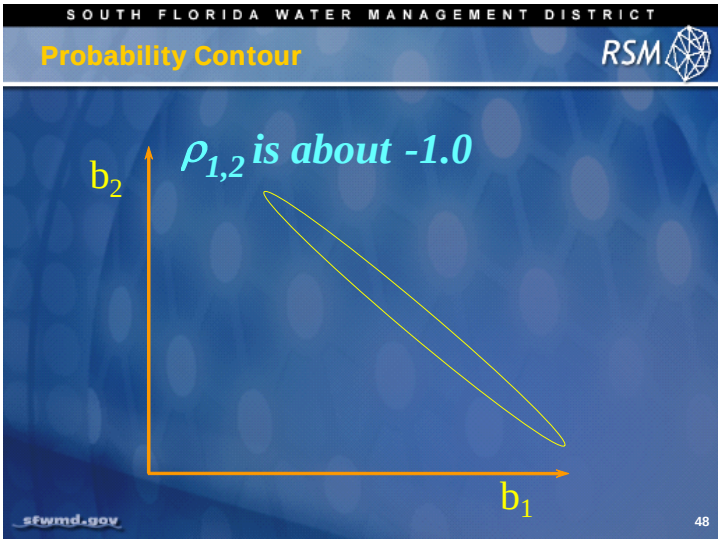
The correlation matrix determines how well the objective function will converge. High values of the correlation coefficients will limit the ability of PEST to work well.



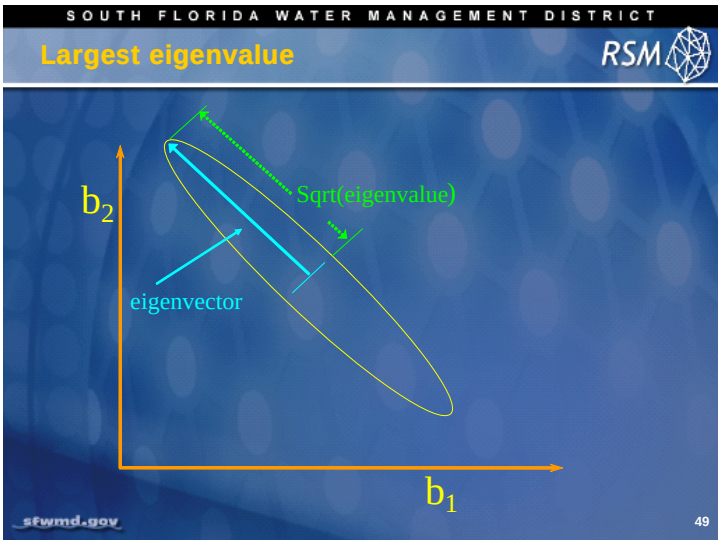
The best situation is where there is no parameter correlation.



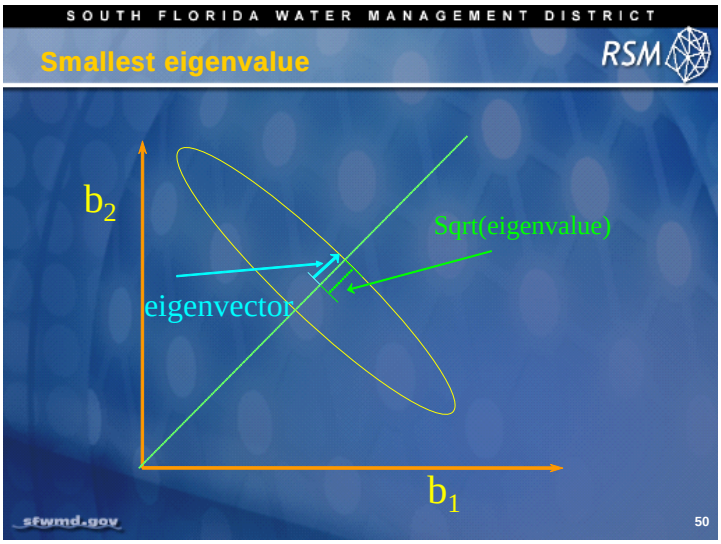
Where there is high parameter correlation, the objective function minimum will be difficult to obtain.



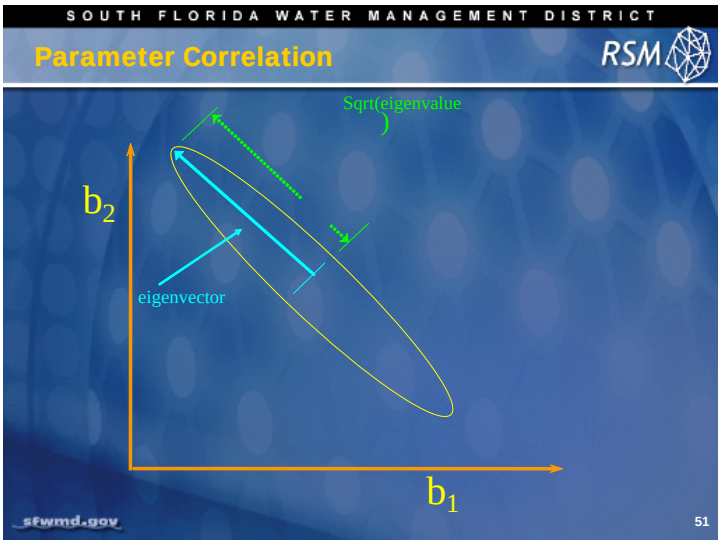
Or…



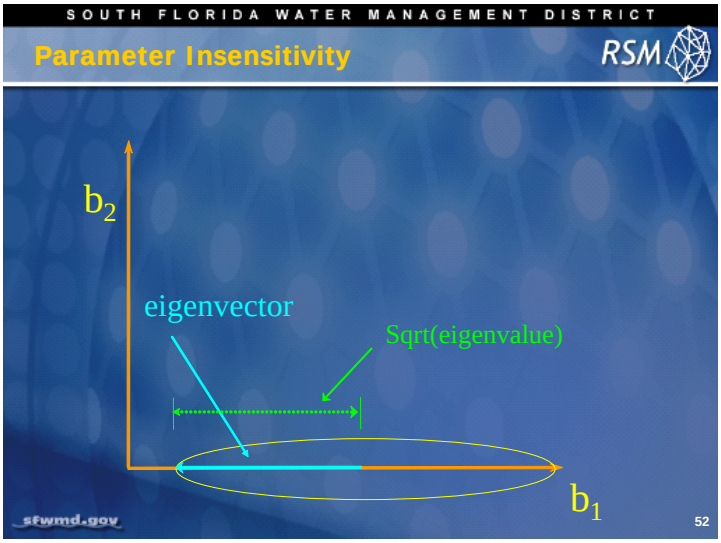
The orthogonal eigenvectors provide information about parameter sensitivity. The largest eigenvalue indicates where the insensitive parameters are likely to occur. The eigenvectors contain the contributions of all of the parameters in orthogonal vectors.



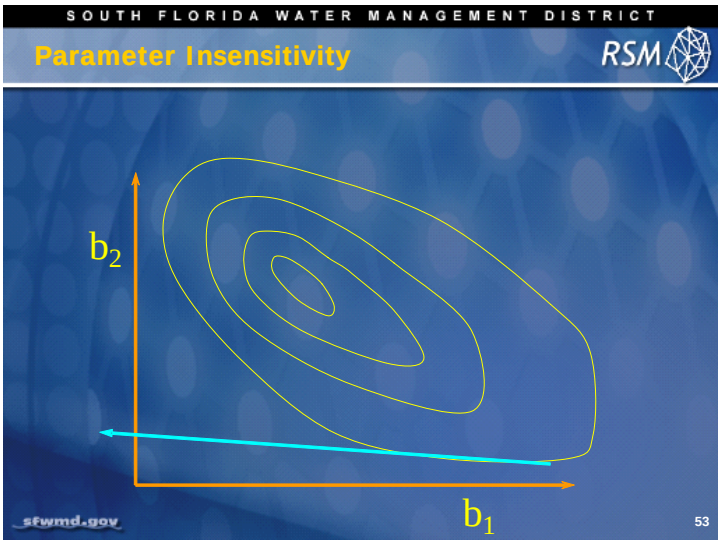
The ratio of the largest eigenvalue to the smallest eigenvalue indicates how well the calibration process is working. If the ratio is greater than 106, then there are insensitive parameters in the model that should be removed.



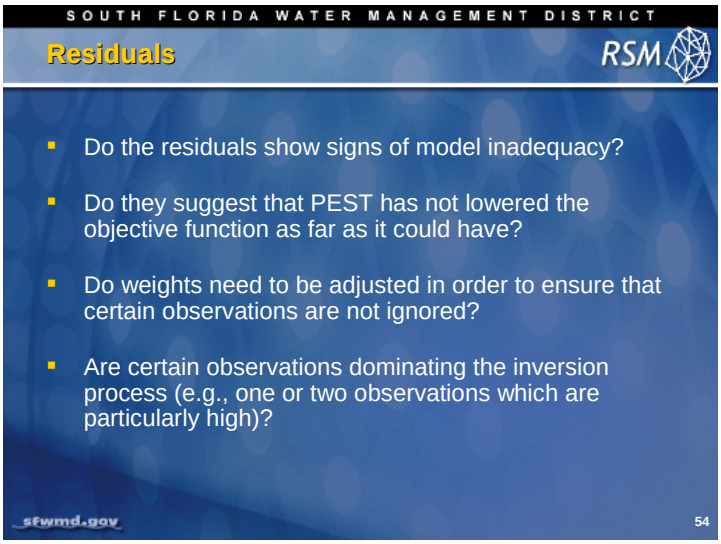
The eigenvector with the largest eigenvalue has many components.



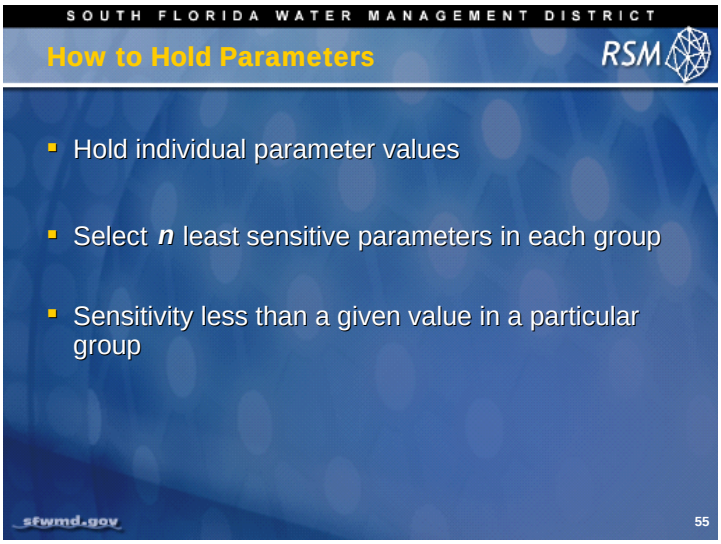
The eigenvector with the largest eigenvalue is typically dominated by one component. In this case, that component is the single parameter, b_1, to which the model is insensitive (i.e., large changes in b_1 have no effect on the objective function).



Parameter b_1 is very insensitive.



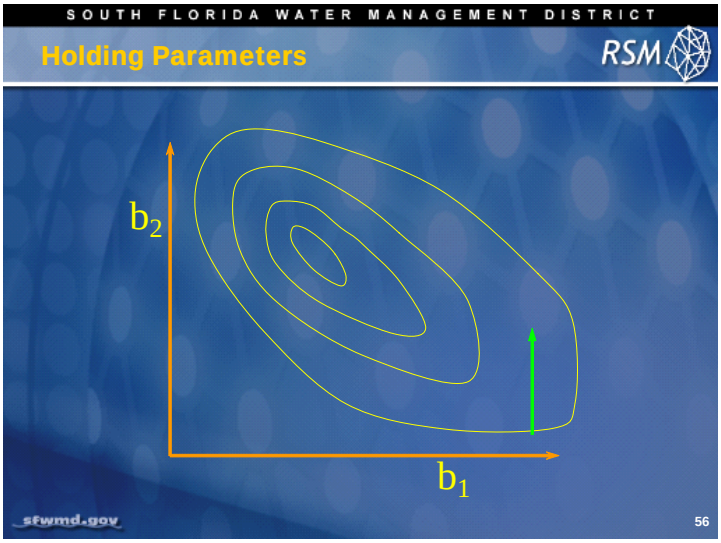
Evaluation of the residuals is an important step in PEST. The residuals provide useful information about the PEST solution and how to refine the process.



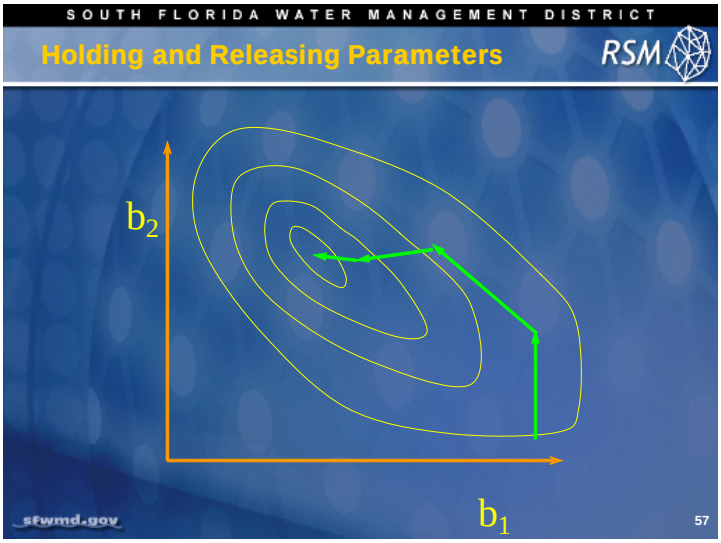
One of the methods for evaluating the impact of different model parameters is to hold a specific parameter constant, rather than letting PEST adjust the parameter value.

PEST allows the user to hold parameters constant for part of the calibration process, and then to adjust the parameter values.

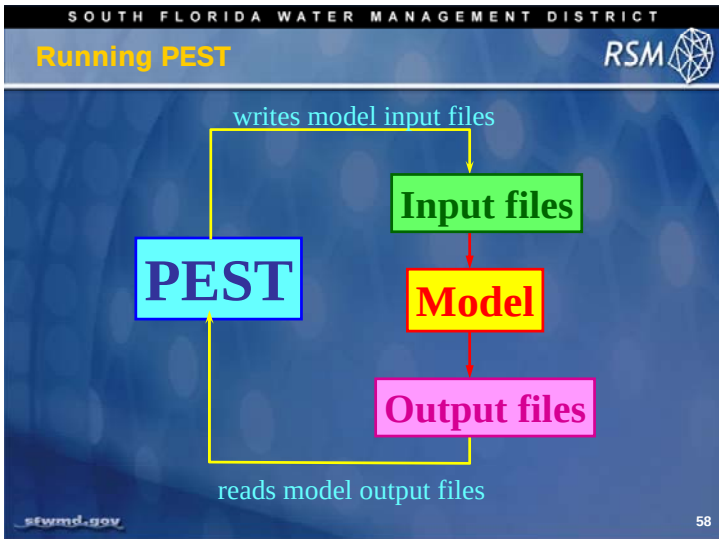
For complex models, this capability enables the user to adjust the parameters in a specific order (e.g. resolving large scale features first and then releasing the calibration of small-scale features).



If parameter b1 is held constant, estimation begins like this:



Then b1 is released. The inversion process converges to optimal b1 and b2.

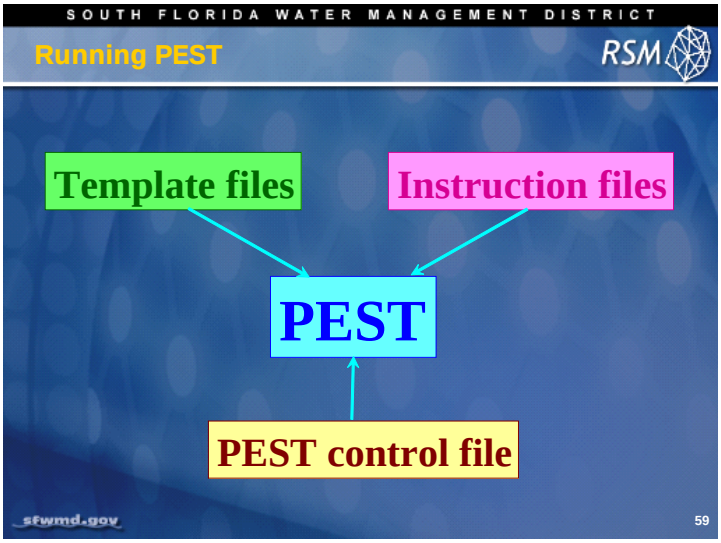


In this segment of the lecture, PEST implementation is discussed.

PEST sits "outside" of the Regional Simulation Model (RSM) and does not interact with the model source code. PEST creates the appropriate model input files and adjusts the user identified parameter values.

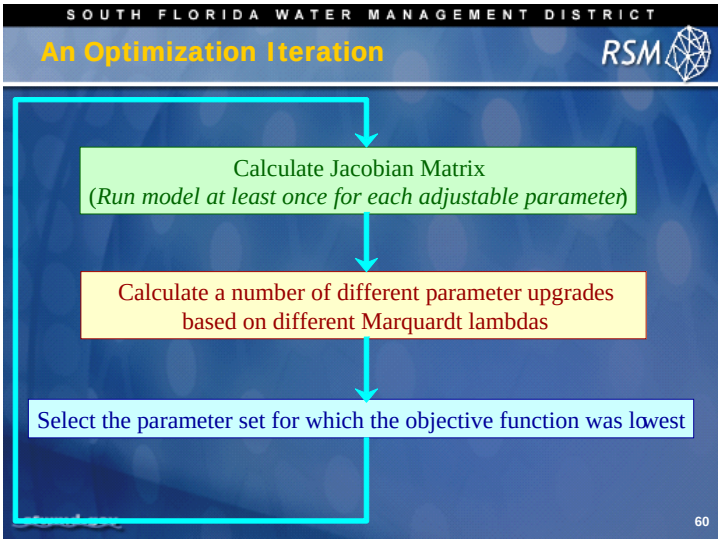
The results of the model are processed to calculate the components of the objective function which PEST uses to adjust the next iteration of the input parameters. In the case of the RSM, the simulated stages and flows are compared to historical observations. The bias and root mean squared error (RMSE) are calculated from that comparison.

The bias and RMSE are used by PEST in the objective function. PEST does not use the RSM output directly, but it uses the processed results from the model.

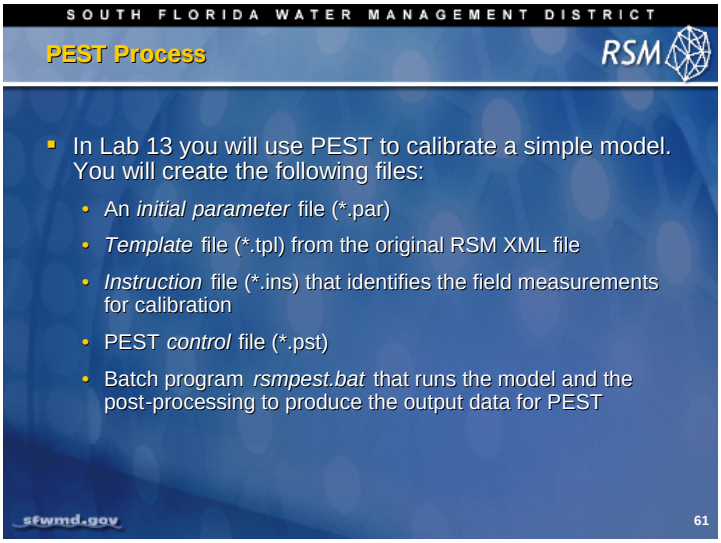


Running PEST requires three files:

- The Instruction file (*.ins) describes where to find the output values that are compared to measured data for model optimization
- The Template file (*.tpl) is similar to the model input file that contains the parameters to be optimized. The template file substitutes the variable names for the parameter values in the template file. This tells PEST which parameters to adjust.
- The PEST Control file (*.pst) contains all of the parameter values to be adjusted, and variables to control the operation of the PEST model run.



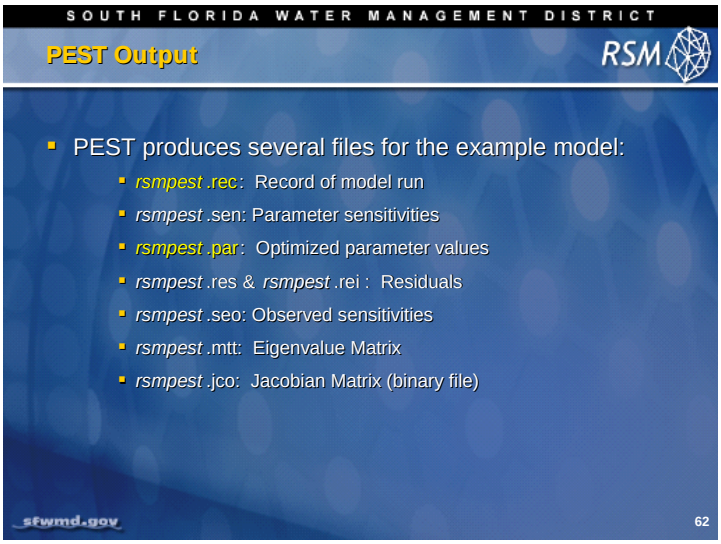
The PEST program first creates the Jacobian Matrix by making **n**+1 model runs. The model then iterates through several parameter upgrade sets based on a modified lambda value.



There are several files that the user must create to run PEST.

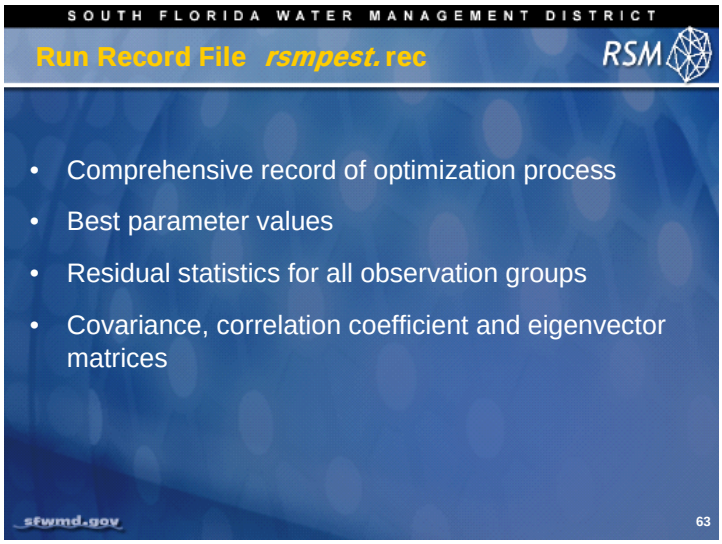
In Lab 13 and Lab 14, you will create the files needed to run a simple RSM model and a simplified RSM subregional implementation.

The PEST User Manual provides examples and detailed instructions for creating the necessary files. Additionally, there are several utility programs provided with PEST that will help create the necessary files, or check the content of the files for syntax and agreement with other necessary files.

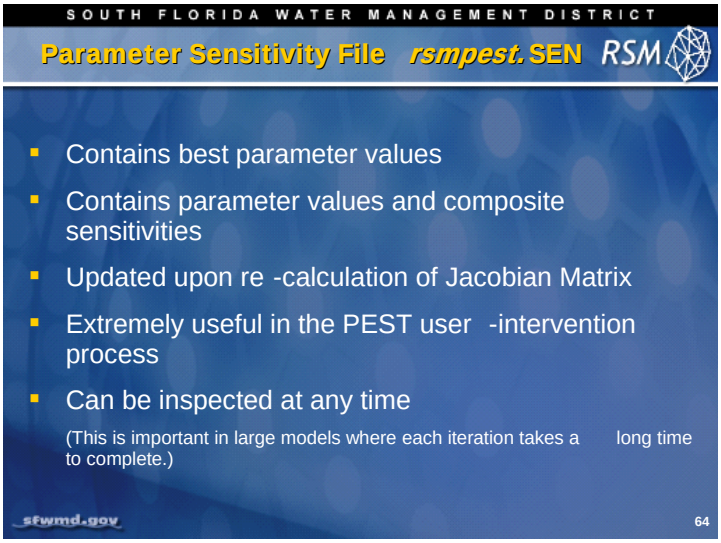


PEST produces several output files that contain the optimized parameter values (case.par) and the record of how they were obtained. This information is useful for interpreting the selection process (case.rec).

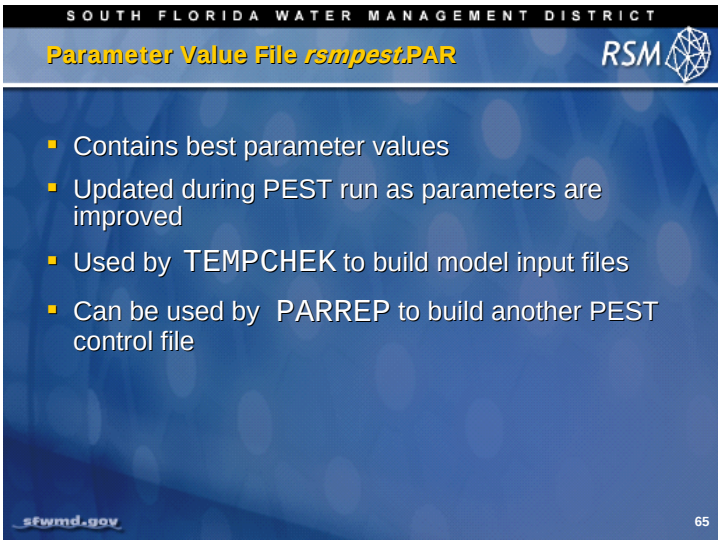
The parameter sensitivities and Jacobian Matrix can be reviewed to determine how well the PEST performed, and what the important parameters and relationships are between parameters.



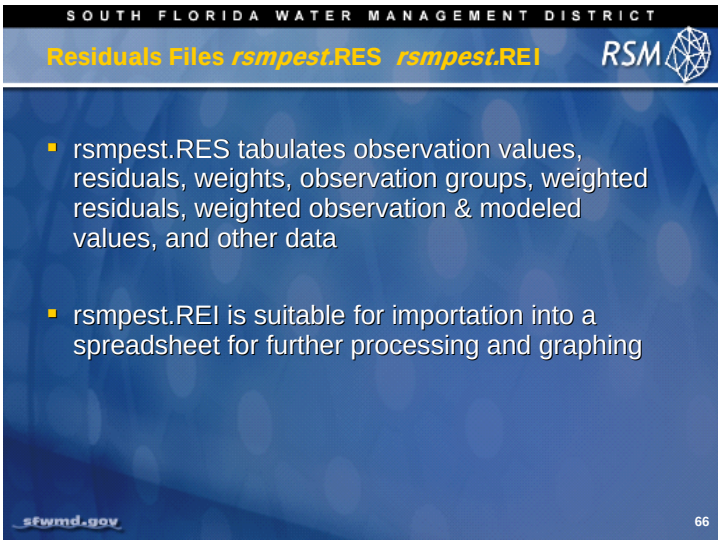
The .rec file provides the summary for the PEST process.



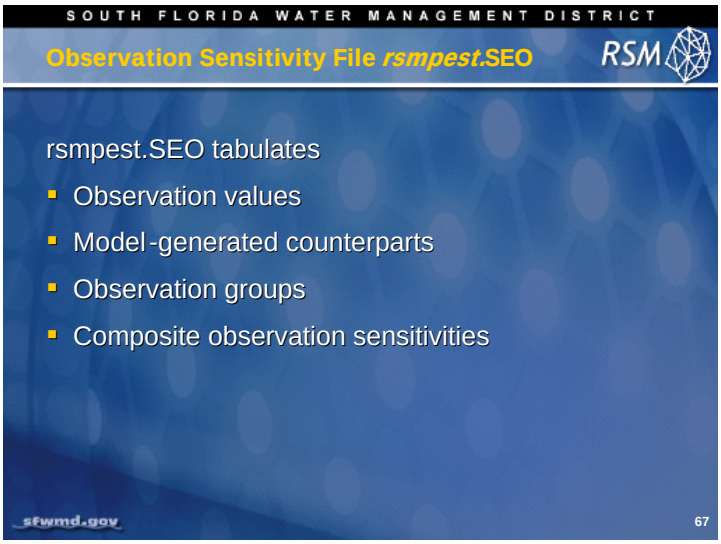
The PEST output file with the .SEN extension provides the information on parameter sensitivities.



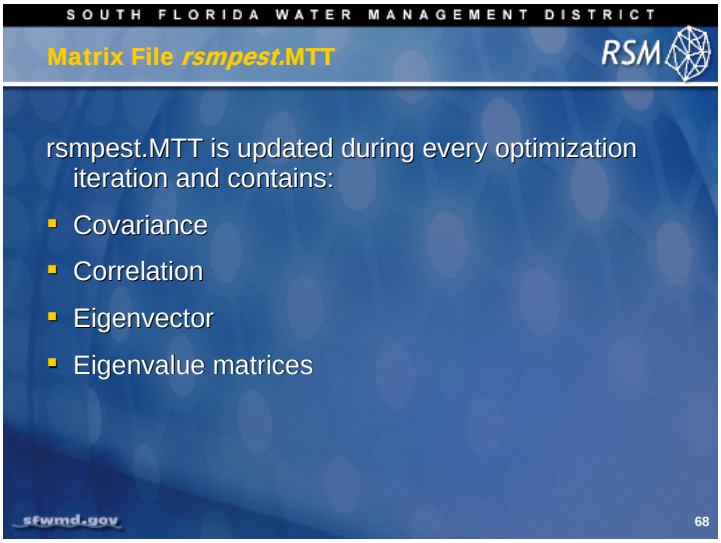
The PEST output file with .PAR extension contains the parameter values.



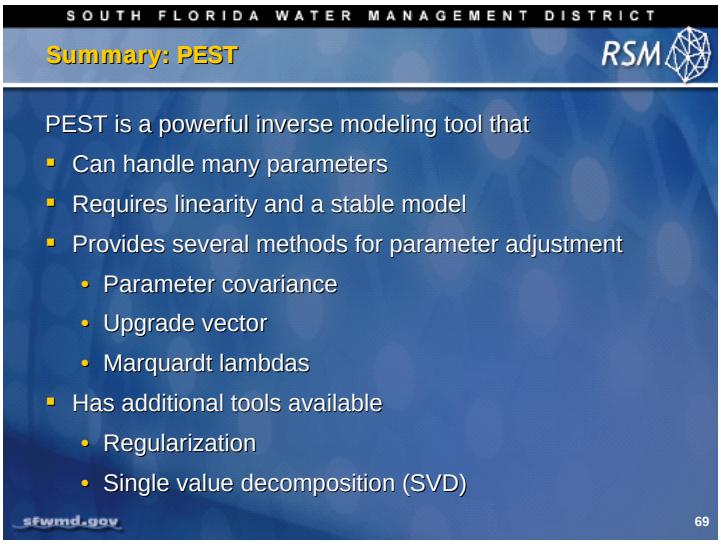
The PEST output file with the .RES extension contains the residual values that are necessary to understand PEST performance. This is a binary file. The PEST .REI file is used to view the residuals.



The PEST file with .SEO extension contains the observed and simulated values with the sensitivities.



The PEST output file with the .MTT extension contains the values from the last iteration.



Additional methods:

Regularization and Single value decomposition (SVD) provide valuable means for obtaining the appropriate parameter values for complex real world models.

Refer to the PEST User Manual for additional details.

KNOWLEDGE ASSESSMENT

(pre- and post-lecture quiz to assess efficacy of training materials)

1. What is PEST used for?
2. How does PEST integrate with RSM to estimate the parameters?
3. How well does PEST work?
4. How can the objective function be formulated?
5. What causes problems with the objective function?
6. Why are weights added to the objective function?
7. How is the upgrade parameter used?
8. What is the Jacobian matrix and how is it used?
9. What is the difficulty in obtaining a good Jacobian matrix?
10. What does the Marquardt lambda do?
11. What information is obtained from evaluation of the model residuals?
12. What are eigenvectors and what is their importance?

Answers

1. PEST is a program that produces an optimal set of user-selected parameters based on satisfying a user-selected optimization function.

2. PEST does not integrate with RSM. PEST post-processes the model output to calculate the objective function and modifies the input XML files appropriately.

3. PEST provides the best unbiased estimator (linear least-squares sense) for parameter values. For a linear model the best parameters are produced in one run. For a non-linear model like RSM, PEST iterates to a solution.

4. The objective function can be formulated in any form the user chooses. Typically, the model results are post-processed to create useful metrics.

5. The objective function can be reduced by poor model stability, large data error and high degree of correlation between parameters.

6. Weights are added to improve the importance of various model residuals.

7. The upgrade vector is used to make the objective function more sensitive to the field observations.

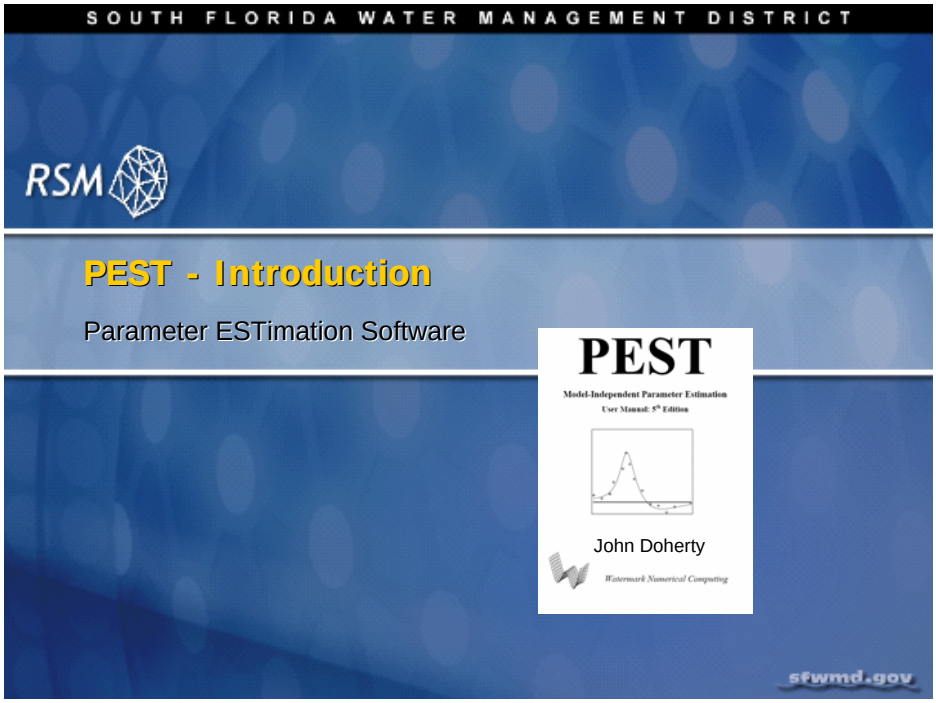
8. The Jacobian matrix is the matrix of partial differences of the observations with respect to the parameters; the parameter sensitivities. It is used to construct the upgrade vector for improving the search process for minimizing the objective function.

9. The Jacobian can be limited by a too-small or too-large increment in parameter adjustments or too high a granularity in the model performance or non-continuousness in the model.

10. The Marquardt lambda is a function of the parameter sensitivities and the upgrade vector. Lambda helps the objective function converge quicker.

11. The model residuals tell us which observations are important, whether the model is adequate and if the objective function could be further reduced (i.e, was the process stopped to soon.)

12. The eigenvectors are normalized orthogonal combinations of the covariances. The associated eigenvalues are useful for determining which parameters are insensitive or correlated and should be removed from the calibration.



Lab 13: Simple RSM PEST application

Time Estimate: 4.0 hours

Training Objective: Learn RSM calibration statistics and simple PEST

A simple inflow-outflow model will be constructed as a Regional Simulation Model to illustrate the behavior of Parameter Estimation (PEST) for model calibration. The user will learn how to create a simple model calibration using PEST and the use of the PEST utilities.

This example is adapted from a simple storage model developed by Dr. John Doherty of Watermark Numerical Computing (2003).

Files for this lab will be created from the basic **`rsmpest.xml`** file in the **`lab13_PEST`** directory. The complete set of resulting files can be found in the **`lab13_PEST/answers`** directory (for comparison).

NOTE:

For ease of navigation, you may wish to set an environment variable to the directory where you install the RSM code using the syntax

```
setenv RSM <path>
```

For SFWMD modelers, the path you should use for the NAS is:

```
/nw/oomdata_ws/nw/oom/sfrsm/workdirs/<username>/trunk
```

```
setenv RSM /nw/oomdata_ws/nw/oom/sfrsm/workdirs/<username>/trunk
```

Once you have set the RSM environment variable to your trunk path, you can use $RSM in any path statement, such as:

```
cd $RSM/benchmarks
```

Training files are currently located in the following directories:

```
INTERNAL_TRAINING
|
|____data
|        |__geographic
|        |__C111
|        |__rain+et
|        |__glades_lecsa
|        |__losa_eaa
|        |__BBCW
|
|____trunk
|        |___benchmarks
|        |___hpmbud
|
|____labs
```

Files for this lab are located in the labs/lab13_PEST directory. Additional materials in the directory include:

PEST User Manual (**`pestman.pdf`**)

Activity 13.1 Calibrating a Simple RSM Model

Overview

Activity 13.1 includes five exercises:

- **Exercise 13.1.1.** Implement a simple RSM model.
- **Exercise 13.1.2.** Construct a PEST template file.
- **Exercise 13.1.3.** Construct a PEST instruction file.
- **Exercise 13.1.4.** Construct a PEST control file.
- **Exercise 13.1.5.** Run PEST

To learn Parameter Estimation (PEST**)** methods you will create a simple Regional Simulation Model (RSM) for calibration. The model will be a simple inflow-outflow model using the standard RSM benchmark mesh (**Fig. 13.1**).

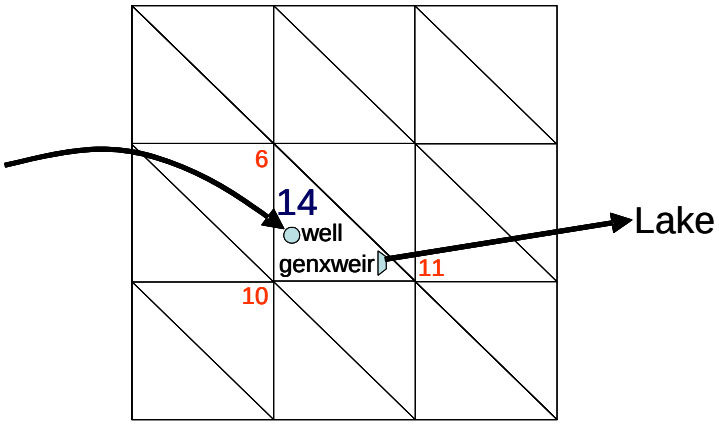


Figure 13.1 Layout for simple inflow-outflow model with recharge well and very large lake

The model consists of a simple 3x3 mesh with no-flow walls around Cell 14. Cell 14 has a well with a constant inflow (R) and a **<genxweir>** that freely drains to a very large lake.

The flow equation for the **<genxweir>** from the RSM User Manual is as follows:

$$q = C_d L (H_u - H_d)^b \quad (1)$$

where
q = discharge in $m^3 s^{-1}$
k = $C_d L$,
$H_d = 0$, H_d = downstream head
b=1, H_u = upstream head

which simplifies to

$$q = k\,h \quad (2)$$

The resulting description of the head in Cell 14 is:

$$s\frac{dh}{dt} = R - kh \tag{3}$$

where
S = Storage (m^3/m^2)

R = Recharge (m^2/s)

for which the solution is:

$$h = h_1 + \left(\frac{R}{k} - h_1\right)\left(1 - e^{\frac{-kt}{s}}\right) \tag{4}$$

The simple model (**Equation 3**) will be implemented in the RSM using a **<well>** for recharge and a **<genxweir>** with discharge to a large off-mesh lake for discharge.

Exercise 13.1.1 Implement a Simple RSM Model

Construct a simple RSM that contains the standard benchmark mesh

1. Copy the **run3x3.xml** from one of the benchmarks and rename it **rsmpest.xml**
2. Set the simulation period for **01jan65** to **01jan70**
3. Create **<noflow>** boundary conditions for each of the walls surrounding Cell 14. These walls must be created individually
4. Set the **bottom = -20**, **surface = 110** and **initial depth = 10.0**. The **aquifer specific yield** is a constant **0.20**
5. Create a well as a **<meshbc>** element following the syntax in Benchmark 7. Set the **recharge rate** at a constant **0.1 m/d, (Note: The areal recharge rate must be converted to m^3/s)**
6. Create a **lake waterbody** following the syntax in Benchmark 70 for the lake specification for the ocean. This creates a lake that will not cause a backwater effect
7. Create a **<genxweir>** as a watermover element following the syntax in Benchmark 68
8. Set:
   - **Forward Flow Coefficient** as **1.0**
   - **Backward Flow Coefficient** as **0.0**
   - **Depth Exponent** as **0**
   - **Slope Exponent** as **1.0**
   - **Crest Elevation** for **5.0**
   - **Crestlength** as **0.15**

The coefficients will produce an outflow of **0.1 m/d.**

The output for this model should include a **<cellmonitor>** for Cell 14 that is directed to a DSS file and a separate **<cellmonitor>** that is directed to an ASCII file following the form for ASCII output from Benchmark 32. The ASCII output will be used to calibrate the model.

The final model should look like the code in **Fig. 13.2** on the following page.

```
<?xml version="1.0" ?>
<!DOCTYPE hse SYSTEM "../../../RSM_REL_2.0.0/benchmarks/hse.dtd" [
]>
<hse>
  <control
    tslen="1" tstype="day" startdate="01jan1965" starttime="0000"
    enddate="31dec1970" endtime="2400" alpha="0.900"
    solver="PETSC" method="gmres"  precond="ilu" runDescriptor="base">
  </control>

  <mesh>
    <geometry file="mesh3x3.2dm"> </geometry>
    <mesh_bc>
      <noflow section="ol_gw"> <nodelist> 6 10  </nodelist> </noflow>
      <noflow section="ol_gw"> <nodelist> 10 11 </nodelist> </noflow>
      <noflow section="ol_gw"> <nodelist> 11 6  </nodelist> </noflow>
      <well label="well" cellid="14" wmID="101">
             <const value="14.46"> </const>
      </well>
    </mesh_bc>
    <shead><gms file="hin3x3b.dat"></gms></shead>
    <bottom>  <const value="-20.0">   </const> </bottom>
    <surface> <const value="110.0"> </const> </surface>
    <conveyance> <mannings a="1.000" detent="0.00001"></mannings>
    </conveyance>
    <transmissivity><unconfined k = "0.02"> </unconfined> </transmissivity>
    <svconverter> <constsv sc="0.20"> </constsv>  </svconverter>
  </mesh>
  <lakes>
     <lake id="20" head0="0.0"><cylinder toparea="2.788e+011" bot="-20.0" />
     </lake>
  </lakes>
  <watermovers>
     <genxweir wmID="102" id1="14" id2="20" fcoeff="1.0" bcoeff="1.0"
               crestelev="5.0" crestlen="0.150" dpower="0.0" spower="1.0">
     </genxweir>
  </watermovers>
  <output>
    <budget file="budget.dat"></budget>
    <budgetpackage file="budget.nc"></budgetpackage>
    <cellmonitor id="14" attr="head" label="c14"><dss file="heads" />
    </cellmonitor>
    <cellmonitor id="14" attr="head" label="c14">
       <asciiform file="heads.dat" format= "%15.12f"/></cellmonitor>
    <lakemonitor id="20"  attr="head" label="l20"><dss  file="heads" />
    </lakemonitor>
    <wmmonitor wmID="101"  attr="flow" label="well"><dss  file="flows" />
    </wmmonitor>
    <wmmonitor wmID="102"  attr="flow" label="weir"><dss  file="flows" />
    </wmmonitor>
  </output>
</hse>
```

Recharge rate

Specific yield

weir length

Figure 13.2 Complete **`rsmpest.xml`** run file

9. Run the model using the RSM Graphical User Interface (RSM GUI).

10. Observe the results in **HecDssVue** from the RSM GUI. The head time series should look like **Fig. 13.3**.

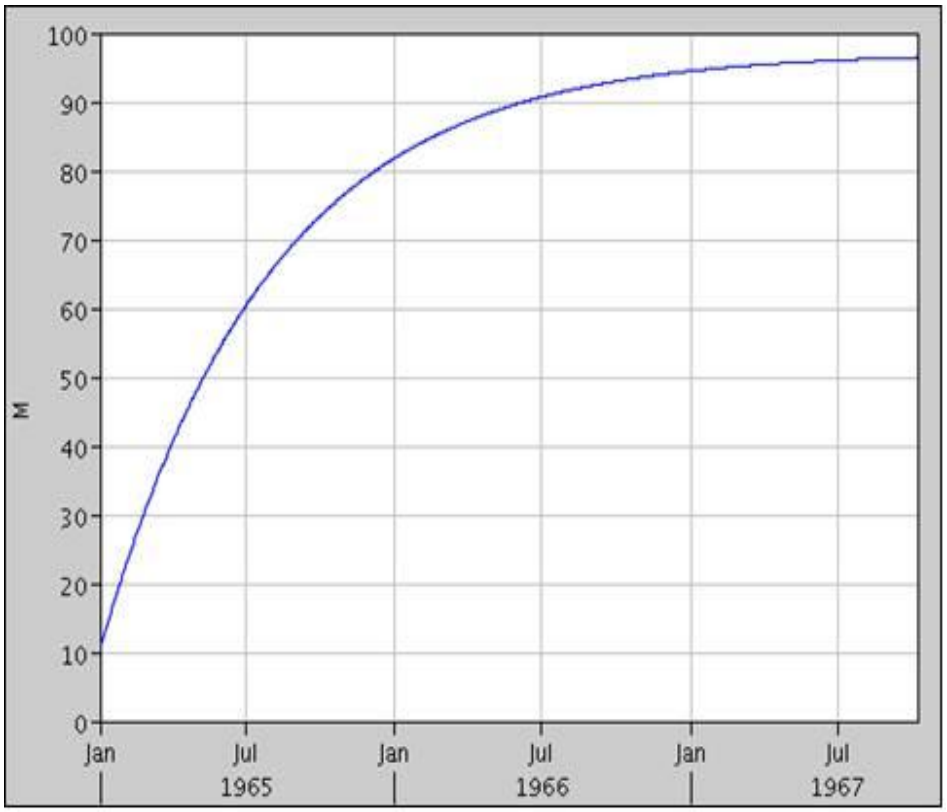


Figure 13.3 Simulated heads in Cell 14 for the simple **rsmpest.xml** RSM

Create the calibration dataset

There are several metrics that can be used for model calibration including water budgets, time series and point observations. In this example, point observations are used for calibration. These point observations are called "field measurements".

A simple program (**logoutput**) has been written and compiled that samples the **heads.dat** file and creates a file, **rsmpest.dat,** containing the time (days) and head values for the days:

- 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000 and 2000.

11. Create the **rsmpest.dat** file using **./logoutput**.

> **NOTE:**
>
> **If there is already a file called rsmpest.dat, delete it so you can create a new one.**

PEST Input Files

To run **PEST** it is necessary to construct three files:

- a Template file,
- an Instruction file
- and a control file.

The Template file indicates the location of the model inputs. The Instruction file reads the values from the model output files for which there are field measurements. The Control file contains the parameters for controlling the PEST run, as well as the names of the model run, template and instruction files.

Activity 13.1.2 Construct a PEST Template File

The first line of a template file must contain the string: "**ptf #**". The "**#**" is used as the parameter delimiter and marks the beginning and end of each "parameter space" as described below. The parameter space must be the size of the expected significant figures required for the parameter estimates.

In this exercise three parameters will be estimated: well recharge rate, aquifer specific yield, and the weir length for the **<genxweir>.**

12. Edit **rsmpest.xml** file.

    - Replace the values for **recharge rate**, **aquifer specific yield** and **weir length coefficient** for the **<genxweir>** with "**well**", "**porosity**" and "**kweir**", with the parameter space of 12 spaces delineated by the '**#**,' included in the 12 spaces:

```
<well label="well" cellid="14" wmID="101">
<const value="#well      #" /const>
<svconverter> <constsv sc="#porosity  #"> </constsv>  </svconverter>
<genxweir wmID="102" id1="14" id2="20" fcoeff="1.0" bcoeff="1.0"
    crestelev="5.0" crestlen="#kweir      #" dpower="0.0" spower="1.0">
```

    - Save template file as **rsmpest.xml.tpl**.

13. Run the TEMPCHEK utility on the template file to make sure it is correct:

    - **tempchek rsmpest.xml.tpl**

    The result should show no errors.

Activity 13.1.3 Construct a PEST Instruction File

The Instruction file is used by PEST to read the field measurements used to calibrate the model. The format of the instruction file is as follows:

- **`1n [parameter name]n1:n2`**

Where:

- **1n** = number of lines processed to the next parameter
- **n1** = column number at the beginning of the parameter
- **n2** = column number at the end of the parameter space

Observing the values in the **`rsmpest.dat`** file, the 11 field measurements "**`head`**"or Water_level are found in column 9 through column 26. The first line of the Instruction file is: "**`pif#`**".

14. Enter an instruction for each head value.

    Your Instruction file **`rsmpest.ins`** should look like the file in **Fig 13.4**

```
pif #
l2 [head1]9:26
l1 [head2]9:26
l1 [head3]9:26
l1 [head4]9:26
l1 [head5]9:26
l1 [head6]9:26
l1 [head7]9:26
l1 [head8]9:26
l1 [head9]9:26
l1 [head10]9:26
l1 [head11]9:26
```

Figure 13.4 The PEST instruction file rsmpest.ins for rsmpest.xml RSM

15. Run the **`INSCHEK`** utility to check the syntax of the instruction file:

    - **`inschek rsmpest.ins`**

16. Run the **`INSCHEK`** utility to check that it matches the field measurements file created in Activity 13.1:

    - **`inschek rsmpest.ins rsmpest.dat`**

    The result should show "no errors"

Activity 13.1.4 Construct a PEST Control File

The **pestgen** utility is used to create the PEST control file.

17. Create a Parameter Value file containing the calibration **parameter names**, **scale** and **offset**. (The file also contains the values for PEST variables **PRECIS** and **DPOINT**: **single** and **point**.)

18. Construct an **initial.par** parameter file with the values listed in **Fig. 13.5**. (These values will be used as a starting point of the calibration process.)

```
single point
well     90.0  1  0
kweir     1.0  1  0
porosity  0.05 1  0
```

Figure 13.5 The PEST parameter value file **initial.par** for **rsmpest.xml**

19. Verify that the observation file **rsmpest.obf** (**Fig. 13.6**), containing the names of the measured values for calibration, was created by the **inschek** utility.

```
head1                    10.44616
head2                    10.89002
head3                    12.20790
head4                    14.35943
head5                    18.49907
head6                    29.70729
head7                    44.92296
head8                    65.74100
head9                    89.95899
head10                   95.97963
head11                   96.47122
```

Figure 13.6 The PEST observation file **rsmpest.obf** for **rsmpest.xml**

20. Verify that the output values for the model are in the **rsmpest.dat** file.

21. Run the **PESTGEN** utility to construct the preliminary control file **rsmpest.pst**. This file will contain reasonable default values.

    - **pestgen rsmpest initial.par rsmpest.obf**

    Note: The **PESTGEN** utility provides a syntax error check on the necessary files.

22. Adjust the *** parameter data** section of the **rsmpest.pst** so that the parameter values are log-transformed to make the inversion problem more linear. Set the upper and lower bounds to the parameter movement to **1.0E-10** and **1.0E+10**, as shown in **Fig 13.7** (this is a 'free format' Fortran input). Change "none relative" to "log factor":

```
* parameter data
well      log factor  90.0      1.0E-10 1.0E+10 well     1.0 0.0  1
kweir     log factor  1.00      1.0E-10 1.0E+10 kweir    1.0 0.0  1
porosity  log factor  5.00E-02  1.0E-10 1.0E+10 porosity 1.0 0.0  1
```

Figure 13.7 The ***parameter data** section of the **rsmpest.pst** file

23. Edit the *** model input/output** section of **rsmpest.pst** and add the file names for running the model (**Fig. 13.8**).

    In the *** model command line** section, the batch script to run the two programs used to produce the model results are added. The **rsmpest.xml** runs the RSM simulation and logout processes the results to produce the observed (simulated) values to match the measured (field) values.

    In the *** model input/output** section the following four files are added:

    - Template file
    - Model Run file
    - Instruction file
    - Field Data file

    PEST takes the template file and creates an input file: **rsmpest.inp.**

```
* model command line
./rsmpestbat
* model input/output
rsmpest.xml.tpl     rsmpest.xml
rsmpest.ins         rsmpest.dat
```

Figure 13.8 Modified *** model command line** and *** model input/output** sections of the **pest2.pst** file

24. Create a batch program **rsmpestbat** (see example in **Fig. 13.9**) to run the **rsmpest.xml** model and produce output that PEST uses to calibrate the model. The batch program will run the RSM using **./hse** directing the results to a junk file. To implement PEST it is necessary to run the model and then post-process the results to get the appropriate values that PEST will compare to the observations.

```
../../trunk/src/hse rsmpest.xml > junk
./logoutput
cat rsmpest.dat
```

Figure 13.9 The content of the **rsmpestbat** script

25. Run the **logoutput** file to resample the model results to produce the values used for calibration and list the results. The results should look like **Fig. 13.10**:

```
  Time          Water_Level
     1      10.446161270142
     2      10.890021324158
     5      12.207904815674
    10      14.359431266785
    20      18.499071121216
    50      29.707294464111
   100      44.922962188721
   200      65.741004943848
   500      89.958992004395
  1000      95.979629516602
  2000      96.471221923828
```

Figure 13.10 The PEST measured values file **rsmpest.dat** for **rsmpest.xml** RSM

26. Run the **PESTCHEK** utility to check for any errors in the Instruction, Template and Control files. The case name "**pest2**" is used without an extension. The complete Control file is shown in **Fig. 13.11**.

- **pestchek rsmpest**

```
Pcf
* control data
restart  estimation
    3    11     3     0     1
    1     1 single point   1   0   0
  5.0   2.0   0.3  0.03    10
  3.0   3.0 0.001  0
  0.1
   30  0.01     3     3  0.01     3
    1     1     1
* parameter groups
well         relative 0.01  0.0  switch  2.0 parabolic
kweir        relative 0.01  0.0  switch  2.0 parabolic
porosity     relative 0.01  0.0  switch  2.0 parabolic
* parameter data
well      none relative  90.0      -1.0E+10 1.0E+10 well     1.0
0.0  1
kweir     none relative  1.00      -1.0E+10 1.0E+10 kweir    1.0
0.0  1
porosity  none relative  5.00E-02  -1.0E+10 1.0E+10 porosity 1.0
0.0  1
* observation groups
obsgroup
* observation data
head1                   10.4462      1.0  obsgroup
head2                   10.8900      1.0  obsgroup
head3                   12.2079      1.0  obsgroup
head4                   14.3594      1.0  obsgroup
head5                   18.4991      1.0  obsgroup
head6                   29.7073      1.0  obsgroup
head7                   44.9230      1.0  obsgroup
head8                   65.7410      1.0  obsgroup
head9                   89.9590      1.0  obsgroup
head10                  95.9796      1.0  obsgroup
head11                  96.4712      1.0  obsgroup
* model command line
./rsmpestbat
* model input/output
rsmpest.xml.tpl  rsmpest.xml
rsmpest.ins      rsmpest.dat
* prior information
```

Figure 13.11 The preliminary PEST control file **`rsmpest.pst`**

Exercise 13.1.5 Run PEST

27. Enter the following command:
    - **`pest rsmpest`**
28. Observe Results. The program will produce several output files:
    - **`rsmpest.par`** – Optimized parameter values
    - **`rsmpest.res`** – Residuals and weighted residuals
    - **`rsmpest.rec`** – Record of the model run, including: best parameter values, residuals, parameter covariance matrix, correlation coefficient matrix, and the eigenvalues and eigenvectors of the covariance matrix.

The results in the file **`rsmpest.par`** show that PEST estimated values are different from the "true" values used to create the input data (**Table 13.1**). This problem is due to the fact that the parameters in the model are not independent. The water level in Cell 14 is determined by the ratios **`recharge/kweir`** and **`porosity/kweir`** rather than three independent parameters. Non-uniqueness is a common problem in complex models. This can be difficult to identify without a nonlinear parametric estimation software package such as PEST.

Table 13.1 Estimated and true values for parameters in **`rsmpest`** RSM

Parameter	"true"	Estimated
Well	14.47	32.53
Kweir	0.150	0.3372
porosity	0.200	0.4496

Activity 13.2 Refining PEST Runs

Overview

Activity 13.2 includes four exercises:

- **Exercise 13.2.1.** Run PEST using fixed parameters
- **Exercise 13.2.2.** Run PEST using prior information
- **Exercise 13.2.3.** Run PEST with uncertain data
- **Exercise 13.2.4.** Run PEST for predictive analysis

Exercise 13.2.1 Run PEST using fixed parameters

In this problem only two of the three parameters can be uniquely estimated. The value of recharge "**`well`**" will be held constant at **`14.47`**.

29. Use the PARameter REPlacement (**PARREP**) utility to replace the values in the PEST control file with the ones from the optimized run. (This provides the best starting point for future parameter optimizations.)
    - **`parrep rsmpest.par rsmpest.pst rsmpest1.pst`**
30. Edit the "**`*parameter data`**" section of the **`rsmpest1.pst`** file as follows:
    - **`well fixed factor 14.47 1.0e-10 1.0e+10 well 1.0 0.0 1`**
31. Run **PEST**
    - **`pest rsmpest1`**

    Observe the result that the parameter estimates are exactly the "true" values of the parameters. Note that the condition numbers are smaller for **`rsmpest1.cnd`** than for **`rsmpest.cnd`** and PEST does not do as much searching within each iteration.

Exercise 13.2.2 Run PEST using prior information

Frequently it is not possible to fix the parameter values due to spatial variability or parameter measurement uncertainty. However, it is possible to provide an *a priori* judgment concerning the weight of a given value or the relationship among values.

In this example, we assume that the preferred value for **`well recharge rate`** is **`14.47 m3/s`** or the **`log(14.47) = 1.16`**. Selection of the weight given to this value is usually based on the standard deviation of the parameter value. The weight should be large enough to influence the objective function without dominating it. In this example a value of **`10`** is selected.

32. Copy **`rsmpest.pst`** to **`rsmpest2.pst`**
33. Add the following line to the **`* prior information`** section:
    - **`pil  1.0 * log(well) = 1.16  10.0`**
34. Change the value of **`NPRIOR`** of **`rsmpest2.pst`** to **`1`**, as shown in **Fig. 13.12**.

    NPRIOR is a parameter that specifies the number of articles of prior information that are used in parameter estimation. In this case, we are using one value.

```
pcf
* control data
restart  estimation
    3    11     3     1     1
    1     1 single point   1   0   0
  5.0   2.0   0.3  0.03    10
  3.0   3.0 0.001  0
  0.1
   30  0.01     3     3  0.01     3
    1     1     1
```

Figure 13.12 Excerpt of **`rsmpest2.pst`** control file showing **`NPRIOR`** parameter

35. Run **`pestchek`** on **`rsmpest2.pst`**
    - **`pestchek rsmpest2.pst`**
36. Run **PEST**
    - **`pest rsmpest2`**
37. Observe the results in the **`rsmpest2.rec`** file;
    - Observe the number of iterations
    - Observe the change in lambda and Phi with each iteration and the change in the parameter values with each iteration. You can plot Phi vs iteration # to see how fast PEST converges.
    - Observe the values of the final residuals and eigenvalues and review lectures 13 and 14 for further interpretation.
    - Observe the effect of changing the *a priori* value from **`14.47`** to **`10.0`**.
    - Observe the effect of reducing the weight from **`10.0`** to **`5.0`**.

Exercise 13.2.3 Run PEST with uncertain data

The prior activities were performed with "perfect" data based on the results of the original model. The data available for modeling is never this good. There will be uncertainty in the data as a result of spatial and temporal variability. This can be evaluated by introducing uncertainty into the data and evaluating the results.

38. Copy **`rsmpest1.pst`** to **`rsmpest3.pst`**
39. Modify the "**`*observation`**" data in **`rsmpest3.pst`** as follows:
    - **`head4`** from **`14.3594`** to **`14.2594`**
    - **`head5`** from **`18.4991`** to **`14.8000`**
    - **`head6`** from **`29.7073`** to **`19.4073`**
    - **`head7`** from **`44.9230`** to **`46.4230`**
40. Run **PESTCHEK** to make sure there are no errors:
    - **`pestchek rsmpest3`**
41. Run **PEST**:
    - **`pest rsmpest3`**
42. Observe the effect of the variable data.

    Note from the **`rsmpest3.rec`** file that phi is not as small as before.

    Note the 95 percent confidence limits for **`kweir`** and **`porosity`** are still reasonable.

PEST is a reasonably robust methodology.

Exercise 13.2.4 Run PEST for predictive analysis

PEST provides a predictive capability for interpolation and extrapolation. This is illustrated using the following steps:

43. Copy **rsmpest3.pst** to **rsmpest4.pst**
44. Edit **rsmpest4.pst**.
    - Replace "**estimation**" with "**prediction**" on the third line.
    - On line 4, change the last integer from **1** to **2**
    - In the ***observation groups** add a new line with the word "**predict**"
    - Change the ***observation data** name of the last line from "**obsgroup**" to "**predict**":
      - **head11 96.4712 1.0 predict**
    - Add a new *** predictive analysis** section at the bottom of **rsmpest4.pst**, as shown in **Fig. 13.13:**

```
* predictive analysis
1
0.5 0.55 1.0
0.00 0.005 1.0 2.0 1
000 0.05
4 0.00 0.005 4
```

Figure 13.13 Excerpt of **rsmpest4.pst** control file showing *** predictive analysis** section

The first value (**1**) is the control variable **NPREDMAXMIN** which tells **PEST** whether to estimate the values as maximum values satisfying the calibration constraints.

The value **0.5,** on the next line, is the variable **PD1,** which is the value of the objective function below which the solution is acceptable.

The remaining values are variables that control the prediction process and are described in the PEST User Manual.

45. Run **PESTCHEK** to make sure that there are no errors:
    - **pestchek rsmpest4**
46. Run **PEST**:
    - **pest rsmpest4**
47. Observe the results in the **rsmpest4.rec** file.

Note the value of **head11** is very close to the observed value as a result of having a good estimation of **kweir** and **porosity**. Although application of the

predictive method for a typically complex RSM is more complicated than shown in this exercise, this example demonstrates the predictive method for a simple model.

Activity 13.3 Bonus PEST Utilities

Overview

Activity 13.3 introduces three bonus utilities of PEST:

- **Bonus 13.3.1.** Building Optimized Model Input Files
- **Bonus 13.3.2.** Calculating Residuals
- **Bonus 13.3.3.** Creating the Jacobian matrix

Bonus 13.3.1 Building Optimized Model Input Files

Following a **PEST** run it is useful to incorporate the final optimized parameter values back into the **`rsmpest.xml`** file. The **`tempchek`** utility can be used as follows:

- **`tempchek rsmpest.xml.tpl rsmpest.xml rsmpest2.par`**

The parameter values from the **`.par`** file are placed back into the **`.xml`** file.

Bonus 13.3.2 Calculating Residuals

Sometimes it is useful to evaluate the residuals of the initial model. This can be accomplished by setting the control parameter **`NOPTMAX`** in the **`rsmpest.pst`** file to "**`0`**" to stop optimization iteration and write out the residuals, as shown in **Fig. 13.14**.

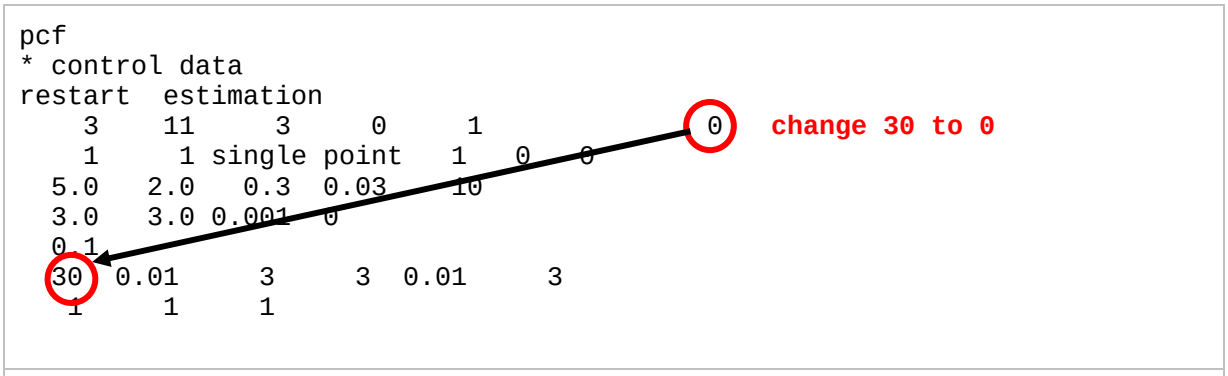


Figure 13.14. Excerpt of rsmpest.pst control file showing location for changing NOPTMAX value.

Typically, the number of optimization iterations is set large enough to allow the model to obtain a solution. To calculate the observation residuals only, only one run is required and it is not necessary to calculate the Jacobian Matrix.

Bonus 13.3.3 Creating the Jacobian Matrix

When the value of **NOPTMAX** is set to **"-1"**, **PEST** will calculate the objective function and the Jacobian Matrix. It will also calculate the:

- Parameter Covariance Matrix,
- Correlation Coefficient Matrix
- Covariance Matrix Eigenvalues and Eigenvectors

which are found in the .MTT file. These values are useful in diagnosing the behavior of the parameters in the model.

```
pcf
* control data
restart  estimation
    3    11     3     0     1              -1   change 30 to -1
    1     1 single point   1   0   0
  5.0   2.0   0.3  0.03     10
  3.0   3.0 0.001  0
  0.1
  30  0.01     3     3  0.01     3
   1     1     1
```

Figure 13.15. Excerpt of rsmpost.pst control file showing location for changing NOPTMAX value.

Answers for Lab 13

Exercise 13.1.1

Compare results with those in the lab13 directory.

Exercise 13.1.2

Compare results with those in the lab13 directory.

Exercise 13.1.3

Compare results with those in the lab13 directory.

Exercise 13.1.4

Compare results with those in the lab13 directory.

Exercise 13.1.5

2. In rampest.rec, Phi is quickly reduced from 15274 to 2.14E-5 in six iterations. However, the resulting parameter estimates are different from the "true" values (Table 13.1) because the problem is over-parameterized as identified in Equation 4 at the beginning of the lab. This is evident by the difference between the largest eigenvalue and the second largest eigenvalue found in rampest.rec file. They differ by a factor of 100,000.

Review of the eigenvectors shows that the three components are the same magnitude. This indicates that there is a large degree of uncertainty in the parameter estimates. Similarly, the parameter correlation coefficient matrix show perfect correlation between the parameters. Also provided in the run record file are the 95% confidence limits which are small due to the low objective function obtained in the PEST run. (Doherty 2005)

Exercise 13.2.1

Compare results with those in the lab13 directory.

Exercise 13.2.2

6. The results are the exact true values. The "suggested" value for **`well recharge`** with a weight of **`10.0`** forced the model essentially to hold the value of **`well`** constant and optimize the other parameters.

- Number of iterations:
- Change in lambda and Phi and parameter values:
- Values of final residuals and eigenvalues:
- **PEST** behaves the same but the optimized values are different.
- The **condition numbers** are much higher and the ratio of the high: lowest **eigenvalues** is higher indicating a more unstable solution.

Exercise 13.2.3

Compare results with those in the lab13 directory.

Exercise 13.2.4

Compare results with those in the lab13 directory.

Bonus Exercise 13.3.1

Compare results with those in the lab13 directory.

Bonus Exercise 13.3.2

Compare results with those in the lab13 directory.

Bonus Exercise 13.3.3

Compare results with those in the lab13 directory.

Index