10-7-2005

Application of Residual Mapping Calibration to a Transient Groundwater Flow Model

Jeremy White
University of South Florida

Application of Residual Mapping Calibration to a Transient Groundwater Flow Model

by

Jeremy White

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
Department of Geology
College of Arts and Sciences
University of South Florida

Major Professor: Mark Stewart, Ph.D.
H. L. Vacher, Ph.D.
Mark Rains, Ph.D.

Date of Approval:
October 7, 2005

Keywords: parameter estimation, automated parameter zonation, aquifer

Acknowledgements

I would like to acknowledge two key individuals who made this research possible. First and foremost, Dr. Mark Stewart, for his enduring patience, guidance and wisdom, and Jim Rumbaugh of Environmental Simulation International, who's help and support was put through test after test. I also would like to acknowledge my committee members, Dr. Len Vacher and Dr. Mark Rains. Finally, I would like to acknowledge Pinellas County for funding this project.

Table of Contents

List of Figures

List of Tables

Application of Residual Mapping to a Transient Groundwater Flow Model

Jeremy White

ABSTRACT

Residual mapping is an automated groundwater-model calibration technique which rapidly identifies parameter-zone configurations, while limiting tendencies to over-parameterize. Residual mapping analyzes the model residual, or the difference between model-calculated head and spatially-interpolated observation data, for non-random trends. These trends are entered in the model as parameter zones. The values of hydrologic variables in each parameter zone are then optimized, using parameter-estimation software. Statistics calculated by the parameter-estimation software are used to determine the statistical significance of the parameter zones. If the parameter-value ranges for adjacent zones do not have significant overlap, the zones are considered to be valid. This technique was applied to a finite-difference, transient groundwater flow model of a major municipal well field, located in west-central Florida. A computer conde automates the residual mapping process, making it practical for application to large, transient flow models. The calibration data set includes head values from 37 monitor wells over a period of 181 days, including a 96-day well-field scale aquifer-performance test. The transient residual-mapping technique identified five significant transmissivity zones and one leakance zone.

Introduction

Groundwater-model calibration has advanced from the traditional trial and error method for determining the size and shape of spatial zones for hydrologic parameters and the "best" parameter value for each parameter zone. The widespread use of parameter-estimation software has made determination of parameter values relatively simple, while determination of the parameter-zone shape and orientation remains a major task for groundwater-model calibration. In addition to determining the shape and orientation of parameter zonation, use of parameter zones with low statistical significance may adversely affect the predictive ability of a model. Using a large number of parameter zones may lower the calibration error, measured by the sum of the difference between observed and calculated heads, but may reduce the predictive ability of a model by increasing non-uniqueness.

The two common parameter-zone-characterization techniques for finite-difference models are zonation and geostatistics. The zonation technique involves distributing parameters of uniform parameter values in polygons of varying shape and orientation. The value of the parameter within each zone is then either manually adjusted or, more recently, optimized using parameter-estimation software to minimize the difference between measured data and model output. Geostatistical characterization usually

involves the use of pilot points and kriging. RamaRoa et al. (1995) introduced an iterative, automated-distribution technique which uses pilot points as synthetic transmissivity data. The pilot points are located and optimized objectively to minimize the difference between model outputs and field measurements. This technique produces several acceptable, calibrated, head distributions. However, Cooley (2000) shows that this technique may rely on invalid assumptions, which encourages over-parameterization and adversely effects uncertainty analysis. Doherty (2003) develops a statistical parameter-distribution technique which utilizes pilot points, parameter-estimation software and regularization techniques. The values of the pilot points are optimized with parameter-estimation software and then the values of parameters are distributed to each cell from the pilot points through kriging, which produces continuously-varying parameter fields. Over-parameterization is limited through the use of a regularization function within the optimization software. The regularization functionality limits the parameter-value variation of the pilot points based on *a-priori* knowledge and relaxation of the objective function of optimization. While this approach produces aesthetically pleasing parameter distributions, substantial data coverage is needed to verify the statistical significance of the continuous parameter fields.

The purpose of this study is to develop an automated-zonation technique for identifying parameter distribution, referred to as the residual-mapping method. This method can rapidly identify valid parameter zones for transient groundwater-flow models. The statistical significance of each parameter zone is readily determined, reducing the risk of using statistically insignificant zones, and increasing model predictive ability.

Residual Mapping Methodology

The technique of residual mapping, developed by Trout (2001), identifies parameter-zone shapes and orientation by analyzing the model residual, or difference between model-calculated heads and observed heads, for non-random spatial trends. Trout showed that for a simple, one-dimensional model (Figure 1), the maximum residual occurred at zone boundaries (Figure 2). This technique was extended to the calibration of a regional steady-state model developed for an area in west-central Florida, with acceptable results (Trout 2001).

Following the technique developed by Trout (2001), the residual-mapping calibration process for a steady-state model begins by determining the best parameter values for a single zone that covers the entire modeled area for each of the most sensitive parameters, usually transmissivity, leakance, and recharge. Then, the most sensitive parameter is optimized while holding all other parameters constant. In this case, the parameter estimation software PEST, developed by Watermark Numerical Computing, was used to optimize the single-zone parameter value of transmissivity. Once the optimization is complete, the model is run using the optimized value and the model-calculated head is subtracted from the values of head estimated from observed values at each model cell center, creating residual maps. This requires that the observed data be spatially

interpolated to a grid with the same dimension as the model. The sensitivity of the residual-mapping method to actual field variations of transmissivity depends on the degree to which the observed data allow the estimated heads at each cell to accurately represent the actual field values of head.

Once the residual map is created, it is analyzed in both horizontal dimensions (X and Y) for local maxima and minima, as well as slope breaks and inflection points. The locations of these features are plotted and continuous spatial trends in the location of these features are identified and joined to create closed polygons. These closed polygons are used in the model as parameter zones. The newly identified parameter zones are optimized simultaneously for one variable, while still holding all other parameters constant, to determine the significance of each individual zone. PEST results include calculated statistics, which, among others, includes 95% confidence intervals and eigen values, calculated for each zone optimized.

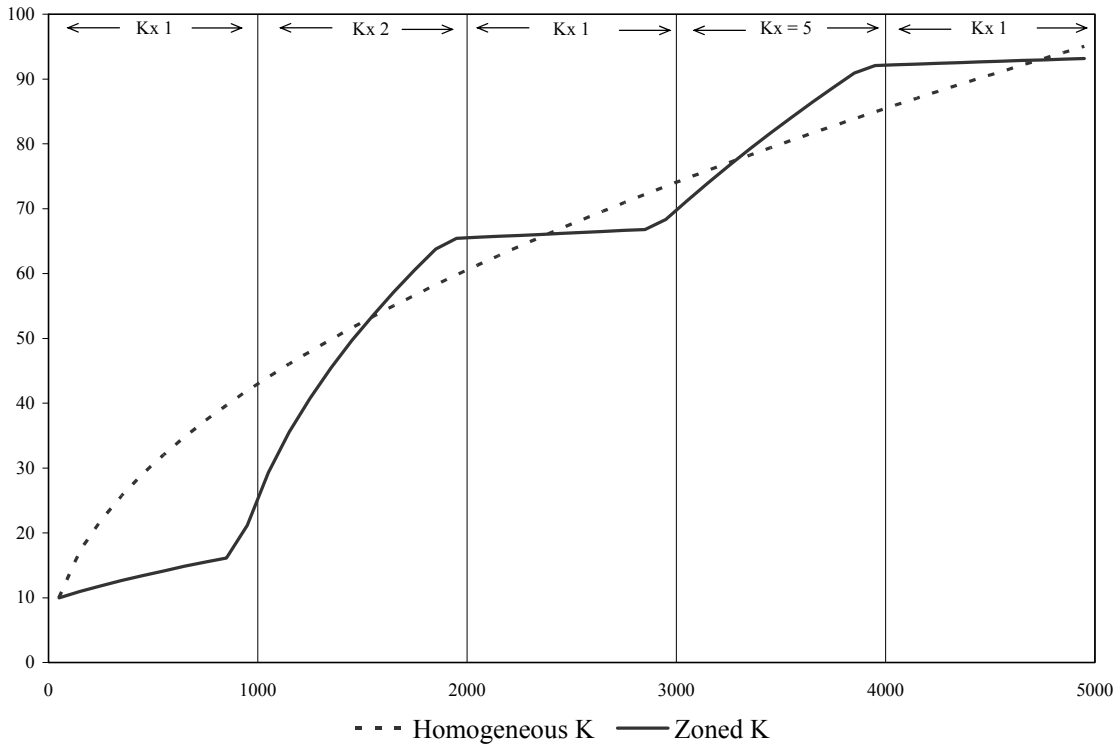

Figure 1 – Head distribution for a one-dimensional finite difference flow model with five conductivity zones and a homogeneous conductivity model (adapted from Trout, 2001).

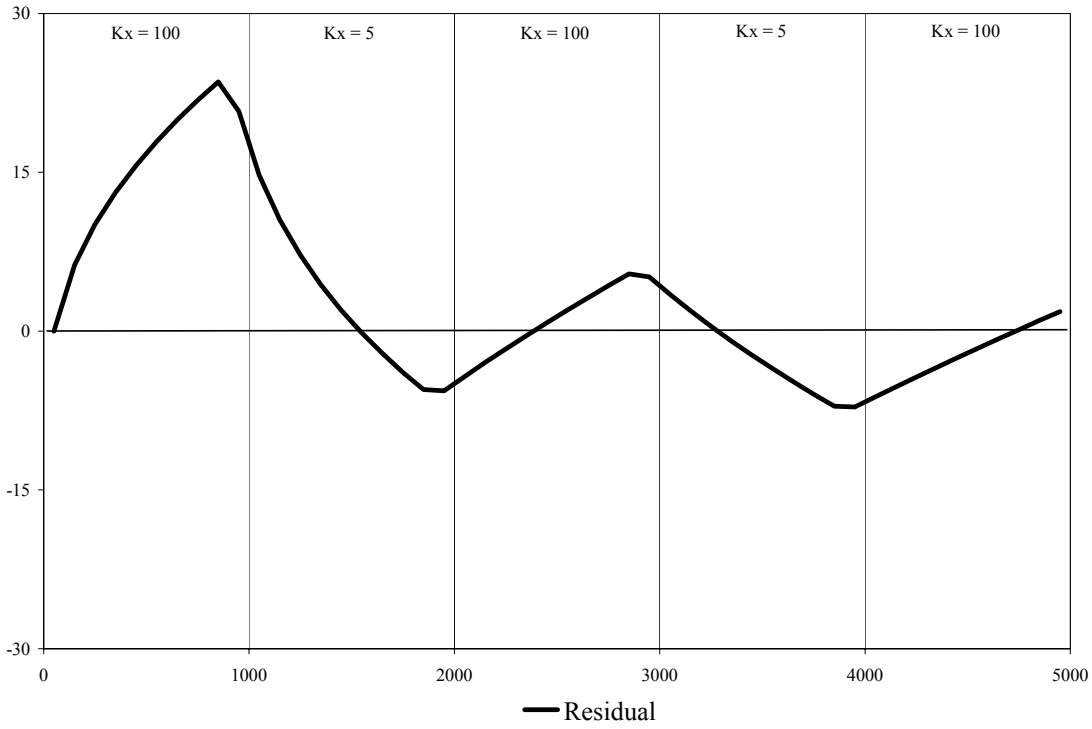

Figure 2 – Difference between heads from the five zone model of Figure 1 and the heads from the homogeneous conductivity model (adapted from Trout, 2001).

Using these two statistics, a determination can quickly be made as to the significance of a particular zone. If a zone has a relatively tight confidence interval and a relatively small eigen value, it is presumed to be significant, as long as no other zone for the same parameter has a confidence interval that is over the same range. If a zone has a relatively broad confidence interval and a relatively large eigen value, or has a confidence interval range similar to another zone, it is combined with adjacent zones.

This entire process is completed for each parameter of interest. The first mapping iteration is completed once all of the sensitive parameters have gone through the residual mapping process. At this point, a second and third iteration may be completed to determine if the residual can be lowered any further by a different zonation. Residual mapping is completed once the residual cannot be lowered any further while maintaining significant zonations for sensitive parameters. Finally, all of the zones for each of the parameters are optimized simultaneously, to determine the final parameter values and statistical significance of each parameter.

To facilitate the application of residual mapping to a transient model, a simple script code was developed to aid in the creation of residual maps, as well as to analyze the residual maps. While Trout (2001) showed that zone boundaries may also be represented by inflection points and slope breaks as well as local maxima and minima in residuals, the residual-mapping code analyzes the residual for local maxima and minima only. The residual-mapping code analyzes the residual from each time-step from the model,

producing a residual map for each time step. Once each of the time-step residual maps are analyzed, the code also produces a total residual map, which is also analyzed for local maxima and minima.

After the code completes analysis of the residual maps, the total number of maxima/minima occurrences at each cell-center is summed for each time-step. These data are then weighted by frequency and plotted as a three-dimensional histogram of frequency-weighted occurrences of maxima/minima. This plot is overlain with the locations of maxima/minima from the analysis of the total residual. From this combined plot, continuous trends of maxima/minima occurrences are identified, and the resulting parameter zones are entered into the model.

Calibration Example

To illustrate the residual mapping method, a model was developed for the Cross Bar Ranch Wellfield (CBRW), north-central Pasco County, in west-central Florida. The wellfield is operated by Tampa Bay Water, and is currently permitted for a daily average withdrawal of 100,000 m^3 per day (30 million gallons per day (mgd)). The well field occupies approximately 30 km^2 (8,000 acres) and has 17 production wells (Figure 3).

The regional hydrogeology of the model domain consists of a surficial sand aquifer system overlying a clayey semi-permeable layer (SPL), and the basal carbonate sequences that make up the Upper Floridan Aquifer (UFA) (Hutchinson, 1985). While the sand units that comprise the surficial aquifer system are laterally extensive, the unit may not be saturated over the whole domain, depending on the position of potentiometric surface of the UFA and the competence of the SPL (Stewart and Langevin, 1999).

The data set used to calibrate the model was collected during a 1997 wellfield-scale aquifer-performance test (APT) performed by the Southwest Florida Water Management District. The APT was performed by lowering the production at CBRW to 57,000 m^3 /d (15 mgd) for 28 days, then increasing the production to 120,000 m^3 /d (32.6 mgd) for 66 days. Average daily flow rates for each of the 17 production wells are included in the data set.

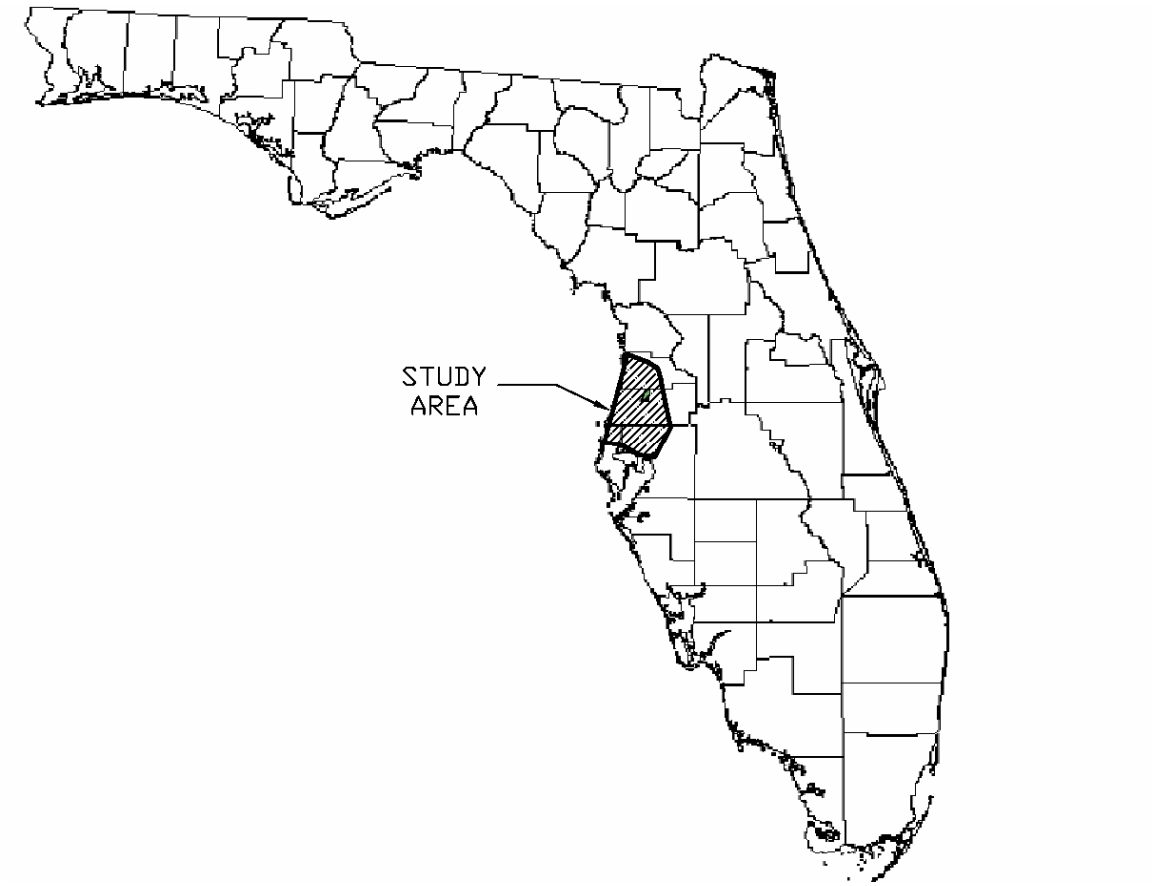

Figure 3 – Study area location.

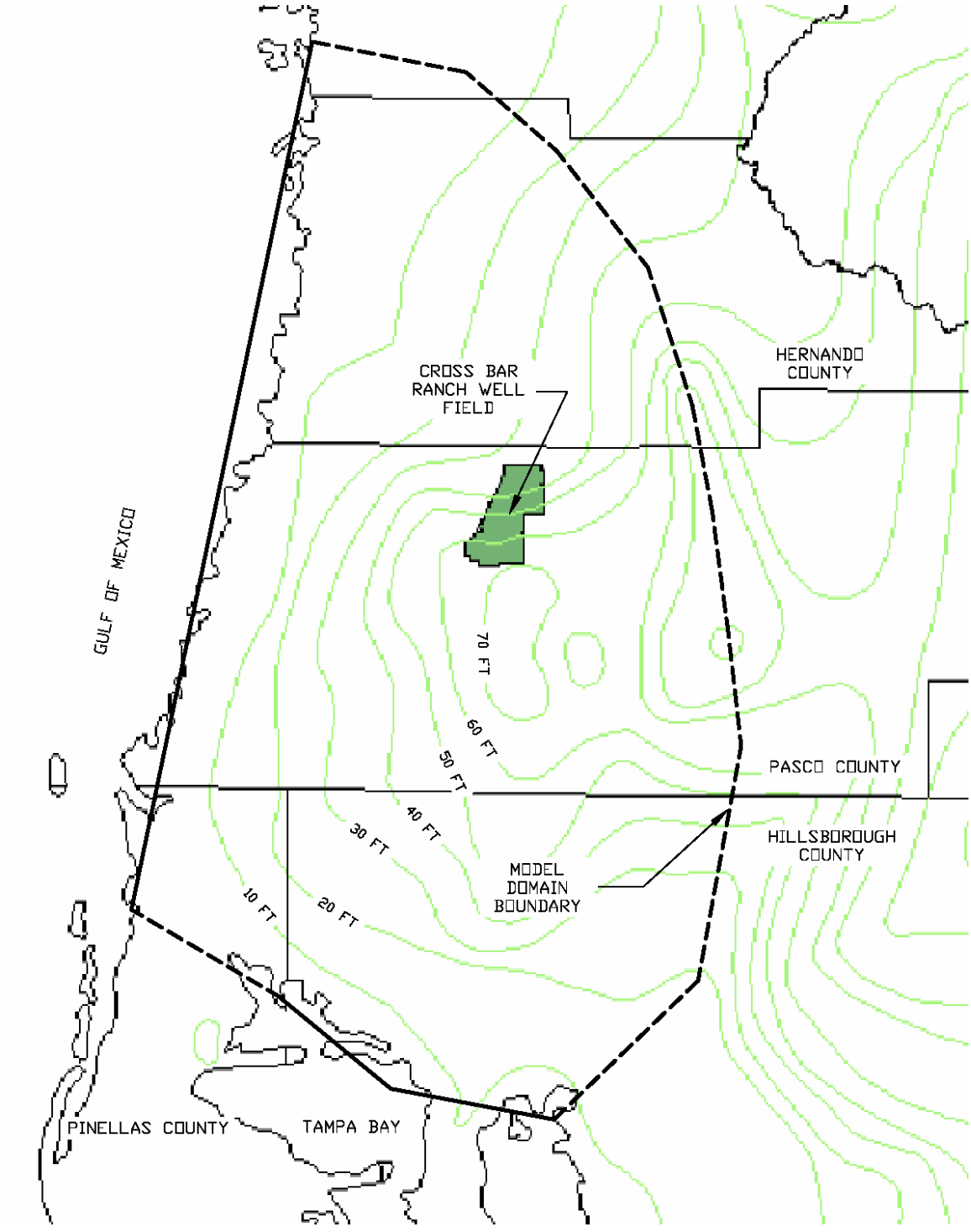

Figure 4 – Map of the Cross Bar Ranch Wellfield model domain. Dashed lines represent no-flow boundaries, while solid lines represent constant-head boundaries. Light green lines represent the inferred Upper Floridan Aquifer potentiometer surface, May 2003 (SWFWMD, 2003).

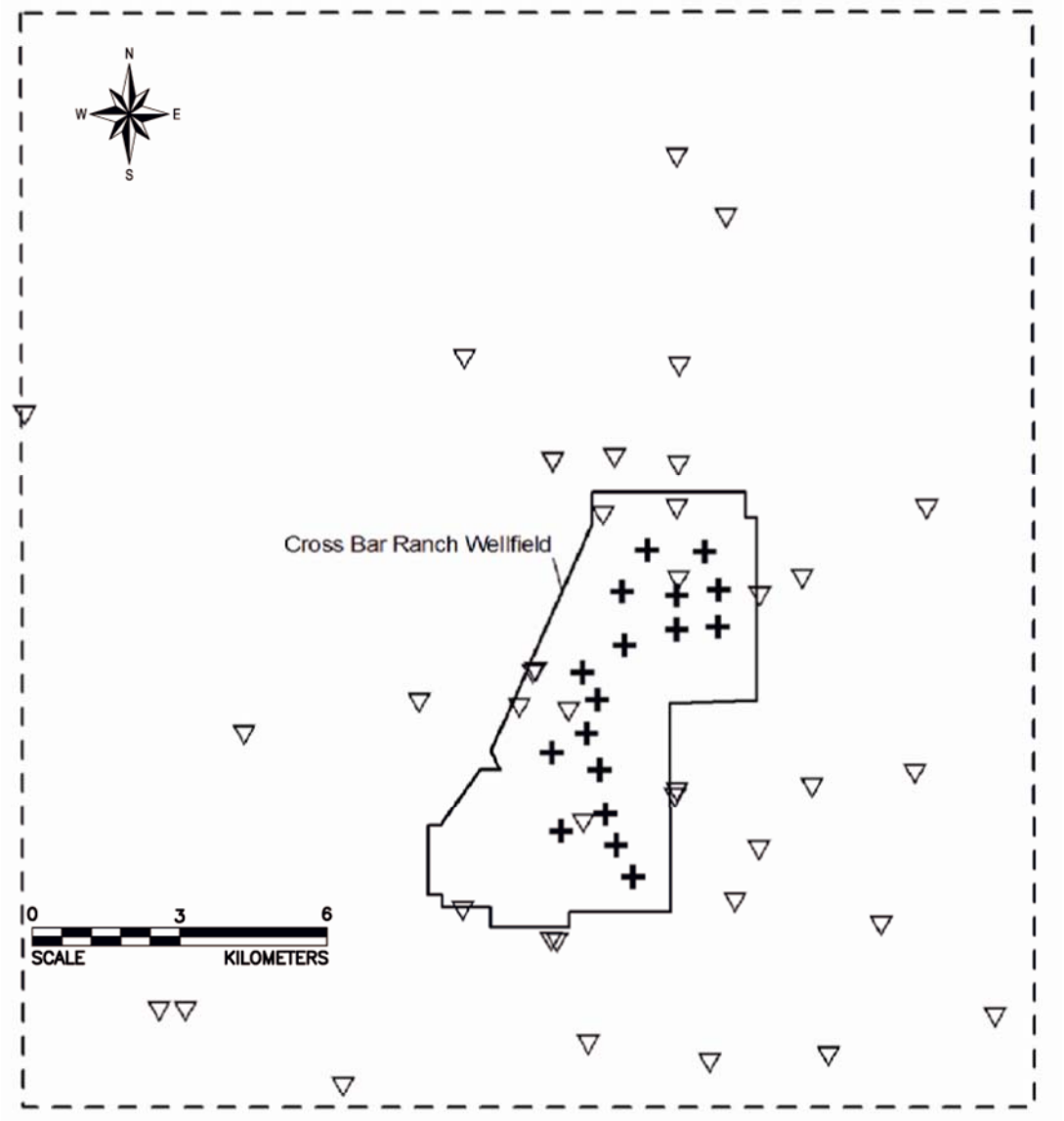

Figure 5 – Location map of Cross Bar Ranch Wellfield, production and monitoring well locations. Production wells are indicated by crosses. Triangles indicate the location of monitoring wells. Bold, dashed line indicates the extent of the residual mapping focus area.

Monitoring of the APT included 37 UFA monitor wells located in the area of the wellfield. Monitoring data from 181 days, beginning January 1, 1997, were used to calibrate the transient model.

The model was conceptualized as a transient, three-layer, non-uniform grid, finite-difference groundwater flow model, using the USGS code MODFLOW (McDonald and Harbaugh 1988) to solve the model. Grid spacing ranges from 1.6 km by 1.6 km (1.0 mile by 1.0 mile) in the region around the wellfield, to 0.40 km by 0.40 km (0.25 miles by 0.25 miles) in the immediate area of the wellfield. Boundary conditions include a constant-head boundary condition along the Gulf of Mexico and Tampa Bay, as well as no-flow boundaries along persistent flow lines from the Green Swamp potentiometric high to the Gulf of Mexico and Tampa Bay. The model consists of 20 variable-length stress periods to discritize the 181 days of monitoring for the calibration data set. The data set contains approximately 5000 transient head targets for calibration.

Results

The first iteration of residual-mapping calibration for the Cross Bar Ranch Wellfield (CBRW) identified five zones of transmissivity and one zone of leakance. The output of the residual-mapping code for the first transmissivity iteration showed reasonable agreement between total-residual maxima/minima and frequency-weighted maxima/minima, as well as strong continuous patterns (Figure 6). Optimization of this zonation indicated that all five zones were significant (Table 1).

The residual output for the first leakance iteration revealed some disparity between total-residual maxima/minima and frequency-weighted maxima/minima (Figure 7). Optimization of this zonation resulted in confidence intervals that were over the same range. For this reason, the three zones were collapsed into a single zone of leakance. An objective function value of 5.27E+04 m was calculated using the transmissivity and leakance zonation of the first iteration.

A second iteration of residual mapping was completed with similar results. The output of the residual mapping procedure for the second iteration of transmissivity identified five significant zones, although in a different orientation (Figure 8). The five zones in the second iteration share some boundaries with the five zones of the first iteration (Figure 6). However, parameter optimization of the zones of the second iteration reveals that one

of the five zones is not significant, and so it was collapsed into the zones that surround it. A second iteration of residual mapping for leakance identified seven possible zones (Figure 9). However the parameter optimization for these zones reveals that the confidence intervals of the seven possible leakance zones encompass the same range. So, the seven zones of leakance were collapsed into a single zone. The final zonations of transmissivity and leakance from the second residual mapping iteration produce an objective function value of 4.42E+04 m. Since the second mapping iteration lowered the objective function, while still maintaining significant zones, the zonation produced from the second residual mapping iteration was used as the final zonation for calibration.

The statistical results of the final optimization of all zones of transmissivity and leakance are summarized in Table 2. These results indicate that each of the five parameters is well resolved, as each parameter is bounded closely by the corresponding 95% confidence interval, and the corresponding eigen values are relatively small.

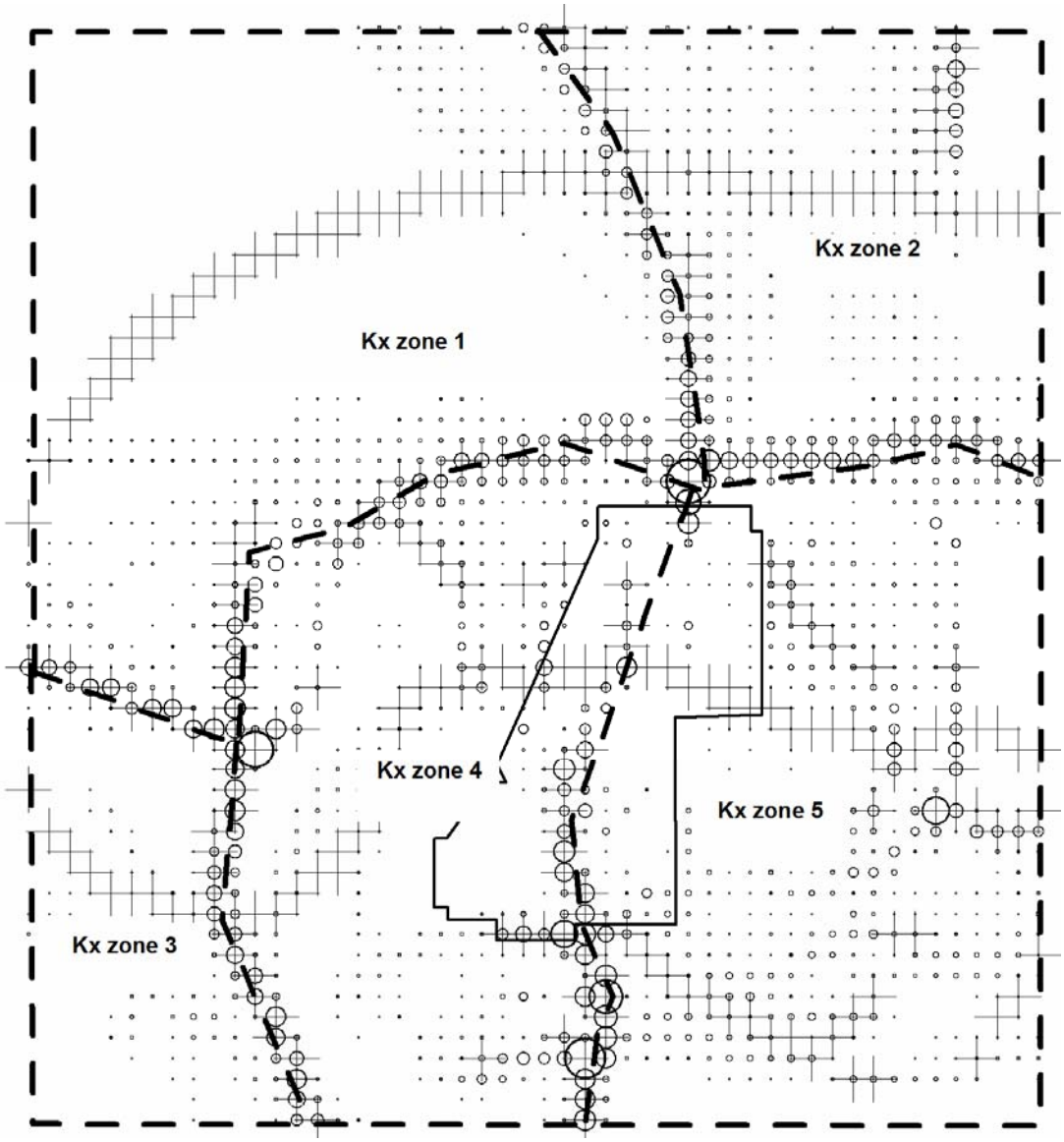

Figure 6 - Transmissivity zonation, first iteration. Five zones were identified from residual mapping, all statistically significant. Crosses are total residual maxima/minima, circles are frequency-weighted maxima/minima, summed from each time step. Bold, dashed lines represent the zone boundaries derived from the residual trends.

Table 1 - Results of parameter optimization of residual mapping derived parameter zones.

Parameter	Estimated Value	Lower 95% Confidence Limit	Upper 95% Confidence Limit	Eigen Value
Kx 1	395.715	298.338	524.876	5.63E-05
Kx 2	164.485	150.035	180.326	1.90E-04
Kx 3	53.1203	47.1881	59.7984	3.12E-04
Kx 4	28.2618	26.2092	30.4751	5.26E-04
Kx 5	84.988	79.9705	90.3203	1.26E-03
Kx 6	78.9828	76.0458	82.0333	2.30E-05
Kx 7	1234.72	928.284	1642.32	1.17E-04
Kx 8	43.2521	1.93125	968.674	7.95E-04
Kx 9	6.7179	5.61273	8.04067	1.50E-03
Kx 10	275.293	249.943	303.215	0.4782
Kz 1	1.27E-02	3.45E-03	4.66E-02	1.59E-03
Kz 2	5.40E-03	1.83E-03	1.60E-02	8.10E-02
Kz 3	1.62E-02	2.02E-03	0.128987	4.33E-02
Kz 4	1.57E-03	1.65E-05	0.150082	4.55E-03
Kz 5	5.06E-03	3.89E-38	6.56E+32	2.46E-02
Kz 6	9.02E-05	2.27E-08	0.357698	0.2938
Kz 7	8.25E-04	8.59E-38	7.93E+30	0.3134
Kz 8	2.33E-04	4.15E-05	1.31E-03	0.7368
Kz 9	9.44E-03	1.66E-04	0.537132	125.6
Kz 10	3.84E-04	1.33E-04	1.11E-03	495.5
Kz 11	5.098	5.68E-130	4.58E+130	4401

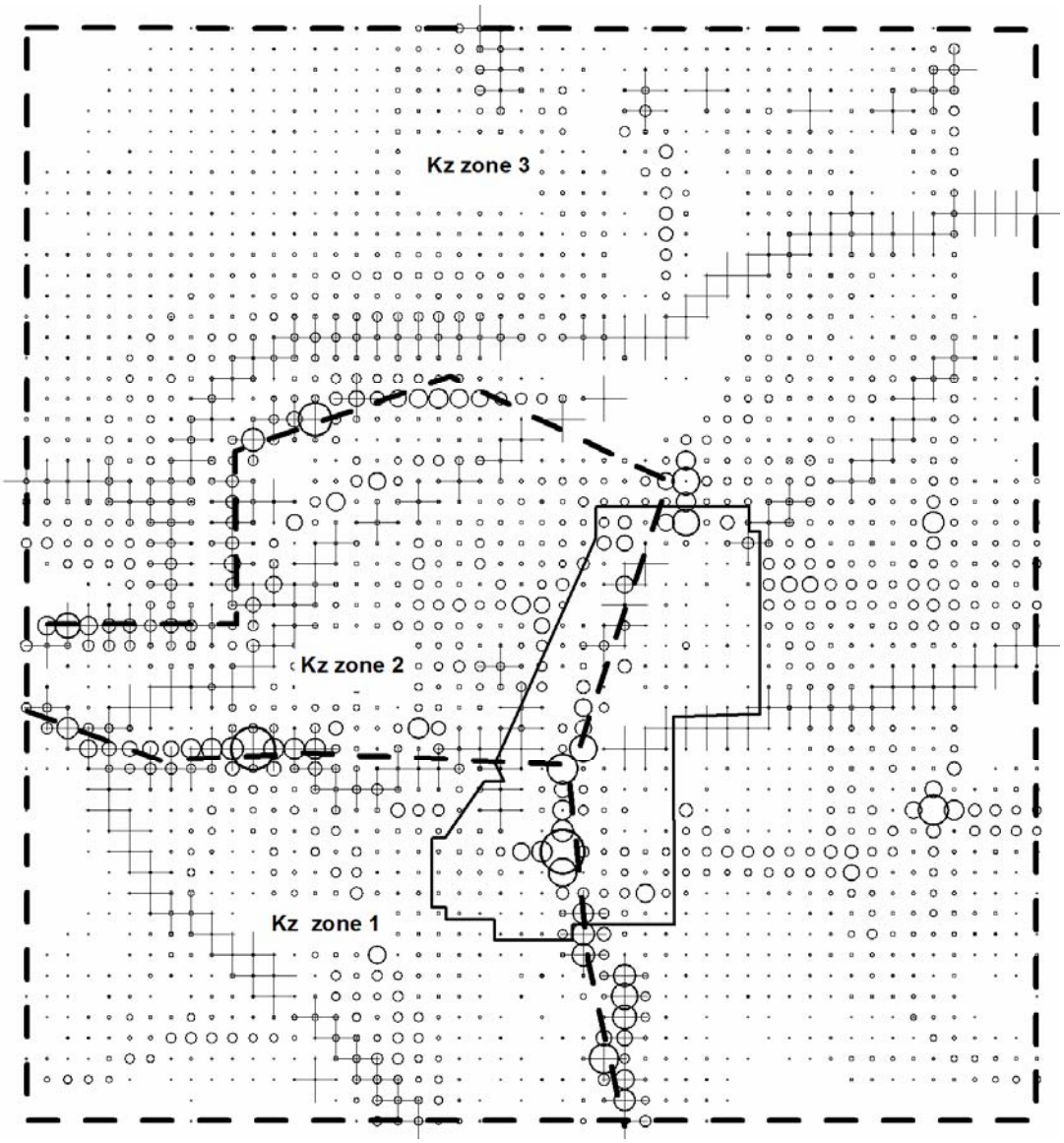

Figure 7 - Leakance zonation, first iteration. Three zones were identified by residual mapping. All three zones have confidence intervals over the same range, so the zones were deemed insignificant and collapsed into a single zone.

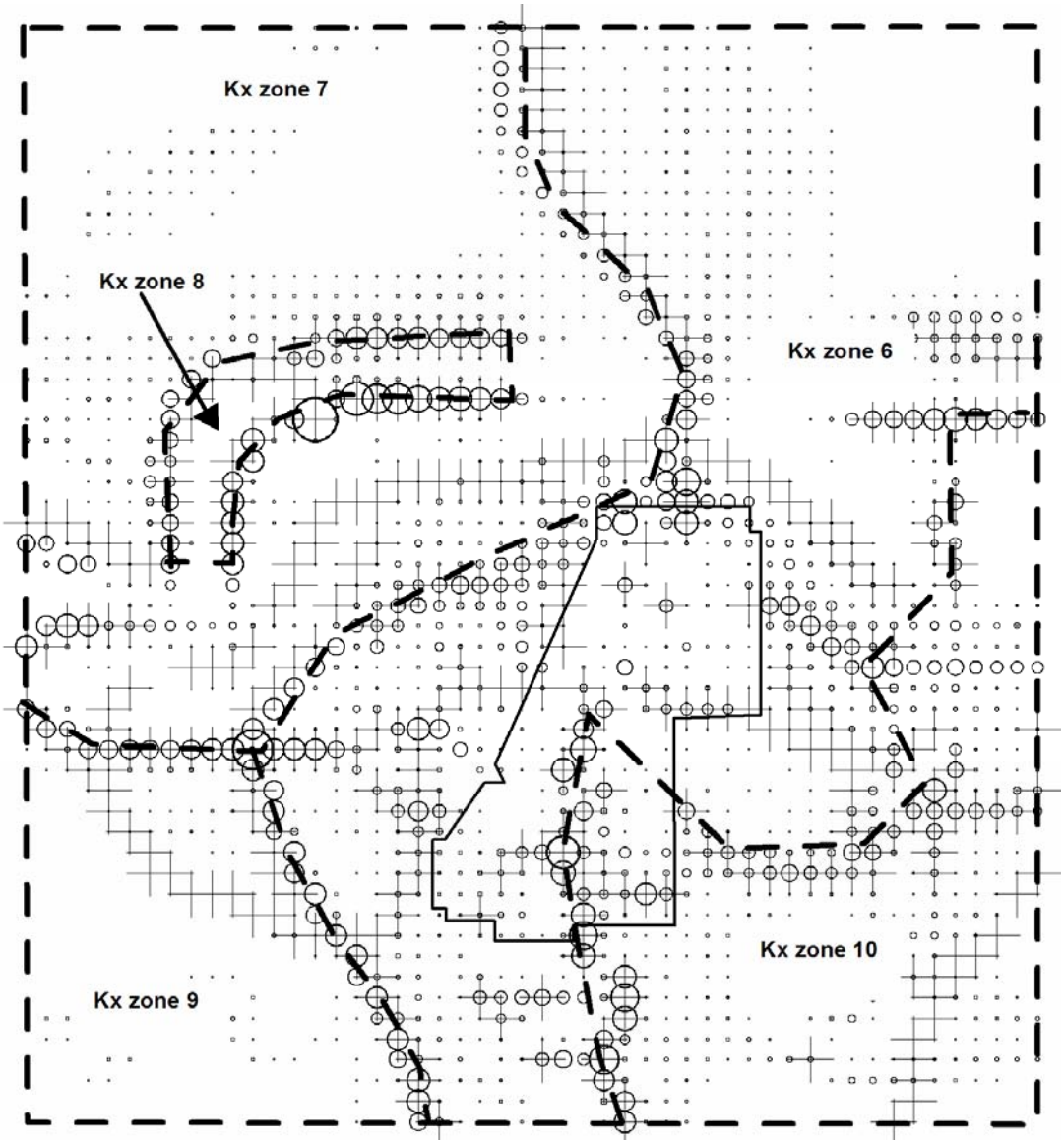

Figure 8 - Transmissivity zonation, second iteration. Five zones were identified by residual mapping. Zone 18 was deemed insignificant and collapsed into zone 17.

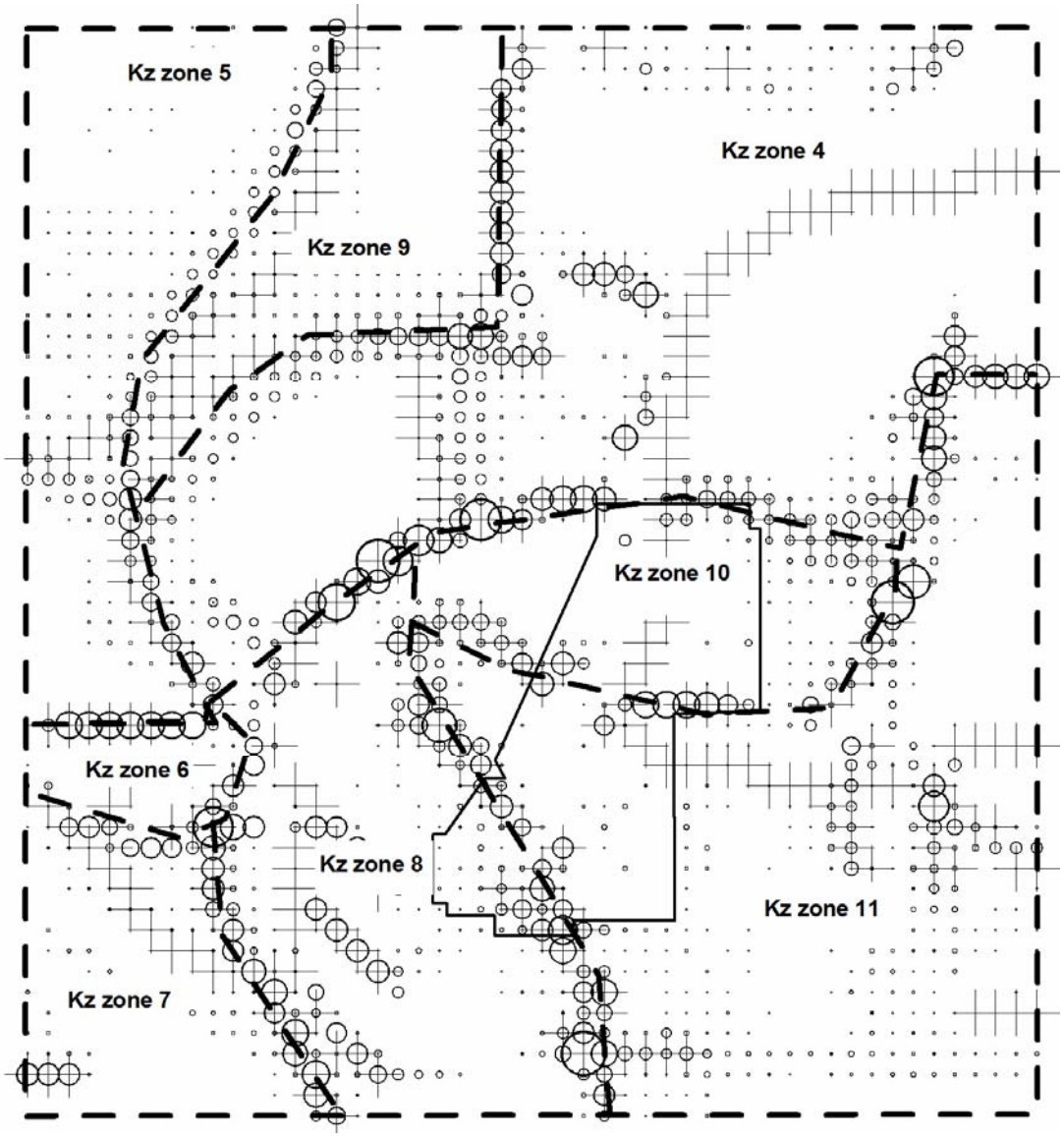

Figure 9 - Leakance zonation, second iteration. Although some zones have more broad intervals than others, all zones have overlapping confidence intervals. Therefore, only a single zone was deemed significant.

Table 2 – Statistical optimization results of the final zonation. All parameters are well resolved.

Parameter	Value	Lower 95% Confidence	Upper 95% Confidence	Eigen Value
Kx 6	69.5872	66.6683	72.6339	2.52E-05
Kx 7	693.094	625.804	767.621	7.10E-05
Kx 9	6.32939	5.24741	7.63446	8.43E-04
Kx 10	358.403	320.65	400.602	1.72E-03
Kz 1	0.285357	0.205087	0.397046	5.64E-03

Discussion

While the residual mapping technique produced four valid zones for transmissivity, only one valid zone for leakance was identified. Realistically, leakance at CBRW may vary more than is characterized by the single parameter value. However, the single leakance zone was found to be the maximum leakance parameterization that the model and corresponding data set could support. The results of the first residual-mapping analysis for leakance indicate three possible zones of leakance (Figure 7). Parameter optimization of the three leakance zones reveals that the parameter zones are statically insignificant, with overlapping confidence intervals that extend over the same range (Table 1). Even though the eigen values of these three zones are reasonably low, the three leakance zones were collapsed into one significant zone because three zones of leakance could not be supported statistically with largely over-lapping confidence intervals. It is this functionality within the residual mapping process that limits over-parameterization. If parameter zones have largely overlapping confidence intervals, then separate zones can not be statistically supported.

In general, the most significant calibration parameters for models of the surficial/Floridan system are recharge, leakance and Floridan transmissivity (Hutchinson, 1985). In this

calibration example of the residual mapping method, transmissivity (*Kx* of model layer 3) and leakance (*Kz* of model layer 2) were optimized.

Although Trout (2001) included recharge zonation in the residual mapping process, recharge was not calibrated for the CBRW model. Rather, a reasonable value was assigned to recharge and evapotranspiration, following the procedure of Motz (1978), where maximum ET (at land surface) is set equal to recharge. Recharge was not calibrated due to the lack of actual recharge data for the period of calibration.

The data set used in the calibration, while seemingly large, still may not adequately characterize the complex flow regime in the area of the wellfield. The location of CBRW coincides with the regional transition from a semi-confined Upper Floridan Aquifer to a largely unconfined Upper Floridan Aquifer (Basso, 1998). This transition is complex and poorly understood, as no direct stratigraphic evidence for loss of confinement has been identified (Upchurch, 1993). This transition may manifest itself in the uncertainty and insignificance of a multiple leakance zonation.

While traditional calibration techniques suggest that a model may support as many parameters as data points, many fewer parameters were identified during the residual mapping calibration example described here. The data set used for calibration contained 37 wells and over 5000 transient data points, yet only five zones of transmissivity and one zone of leakance were identified. While this parameterization may seem over-simplified, no additional statistically-significant zones could be supported.

Conclusions

The residual-mapping technique can rapidly identify the spatial distribution of statistically-valid parameter zones for transient, finite-difference groundwater-flow models. The technique, which includes the use of parameter-estimation software, rapidly identifies possible parameter zonations that are statistically significant while reducing tendencies to over-parameterize. A simple script code, developed to automate the residual-mapping technique, makes the application to transient models more manageable and feasible.

In the test case for the automated residual-mapping code, a transient finite-difference groundwater-flow model was developed for the Cross Bar Ranch Wellfield in Pasco County, west-central Florida. The model was calibrated exclusively using the residual mapping process, which produced five parameter zones for transmissivity, and one parameter zone for leakance. All of the zones in the final zonation where deemed independent and significant by analyzing the statistics calculated by the parameter-estimation software, PEST.

A copy of the residual mapping source code used in this example can be obtained by contacting the corresponding author. This code is supplied "as is", and the University of South Florida accepts no responsibility or liability for any errors or omissions.

References Cited

Basso, R and Senh, S. 1998. Summary Report of Drilling and Testing for Cross Bar Ranch Wellfield Regional Monitoring. Southwest Florida Water Management District.

Cooley, Richard. 2000. An Analysis of the Pilot Point Methodology for Automated Calibration of an Ensemble of Conditionally Simulated Transmissivity Fields. Water Resources Research, vol 36, no 4, pp 1159-1163.

Doherty, John. 2003. Ground Water Model Calibration Using Pilot Points and Regularization. Groundwater, vol 41, no 2, pp 170-177.

Hutchinson, C.B. 1985. Hydrogeology of the Cross Bar Ranch Well-field Area and Projected Impact of Pumping, Pasco County, Florida, U.S. Geological Survey Water Resources Investigation Report 85-4001.

McDonald, M.G., and Harbaugh, A.W., 1988, A modular three-dimensional finite-difference ground-water flow model: U.S. Geological Survey Techniques of Water-Resources Investigations, book 6, chap. A1, 586 p.

Motz, L. 1981. Well field drawdowns using coupled aquifer model. Ground Water 19, no 2: 172-179.

RamaRoa, B, et al. 1995. Pilot Point Methodology for Automated Calibration of an Ensemble of Conditionally Simulated Transmissivity Fields 1. Theory and Computational Experiments. Water Resources Research, vol 31, no 3, pp 475-493

Stewart M.and Langevin, C. 1999. Post audit of a numerical prediction of wellfield drawdown in a semiconfined aquifer system. Ground Water, vol.37, no.2, pp.245-252, Apr 1999

SWFWMD. 2004. Online GIS data distribution website. http://www.swfwmd.state.fl.us/data/gis/

Trout, K. 2001. Improving Groundwater Models With the Use of Parameter Estimation Software. University of South Florida Dissertation.

Upchurch, S. 1993. Cross Bar Ranch Wellfield Water Resource Evaluation. ERM-South, Inc. Tampa, Fl

Appendix A: Residual Mapping Source Code

```
use Math::Complex;#uses a built-in perl module to handle math operations
print " Name of text file? \n";
$txt_file = <STDIN>;
chomp($txt_file);
open (FILE1, "$txt_file") || die "File Access Failure\n";
open (CENTERS, "CELLCENTERS.dat") || die "FUCK ME\n";
#open FILEHANDLE, ">OUT.dat" or die " --> Write File Failure! <--";  # open file for
output of ln conversion data
@raw, @temp, @CO2, @xDist, @yDist; #define arrays
$a=0, $b=0, $c=0, $d=0, $e=0,$x=0; #define varibles
while(<CENTERS>)
{
        @raw = split("",$_);
        $cellcenters [$x] = $raw[0];
        $x++;
        $cellcenters [$x] = $raw[1];
        $x++;

}
$day = 1;
while(<FILE1>) #while loop to read in data from file to array1
{
        @raw = split(" ",$_);
        $size = scalar(@raw);
        $line = 0;
        #print "$raw[0], $raw[1], $raw[2], $raw[3], $raw[4]\n";
        $c++;

        if($c >= 257167)
        {
                for($line=0;$line<5;$line++)
                {
                        if($LineCount == 0)
                        {
                                $FileTitle = "$raw[2]-$raw[3]-$raw[0]-$raw[1]-$raw[4]-
$raw[5]";
                                open FILEHANDLE, ">compday$day.dat" or die " -->
Write File Failure! <--";  # open file for output of ln conversion data
                                print "LC $LineCount, compday$day.dat -- $FileTitle --
$c\n";

                        }
                        if(($LineCount >= 1) && ($LineCount <= 1412))
```

```
                        {

                                print FILEHANDLE "$cellcenters[$q] $cellcenters[$q+1]
$raw[$line]\n";
                                #print "$cellcenters[$q], $cellcenters[$q+1],
$raw[$line]\n";
                                $q = $q +2;

                        }
                        if($LineCount == 1413)
                        {
                                #print "@raw\n";
                        }
                        if($LineCount == 4239)
                        {
                                #print "@raw\n";
                                $LineCount = 0;
                                $q = 0;
                                $day++;
                        }

                }
                $LineCount++;

        }
}


#open FILEHANDLE, ">$FileTitle.dat" or die " --> Write File Failure! <--";  # open file
for output of ln conversion data
#for($t=0;$t<$y;$t++)
#{

#       for($r=0;$r<;$r++)
#       {
#               print FILEHANDLE " $filedata[$y][$r]\n";
#       }
#}

print "\n\ntotal lines $c\n\n";

close FILEHANDLE;#close open file connections
close (FILE1);
```

```
pen STRESS, ">stressper.dat" || die "STRESS FAILED\n";
print "What Mapping Parameter and Iteration (MPI)?\n";
$MPI = <STDIN>;
print STRESS "1,$MPI";
close(STRESS);
open FILE1, ">$MPI-TotalMaxMin.dat" || die "Failure\n";
open FILE2, ">$MPI-TotalResidDiff.dat" || die "FAILURE\n";
print FILE1 "";
print FILE2 "";
close FILE1;
close FILE2;
```

```
$day=1;
print "$day\n";
while($day<=181)
{
	$e=0, $c=0;
	print "$day\n";
	print "looking for file comp$day.dat\n";
	open (FILE1, "compday$day.dat") || die "File Access Failure\n";
	print "found compday$day.dat\n";
	while(<FILE1>) #while loop to read in data from file to array1
	{
		@raw = split(" ", $_);

		if(($raw[0] >= 64020) && ($raw[0] <= 128700) && ($raw[1] >= 148500)
&& ($raw[1] <= 218460) && ($raw[2] != 999) && ($e > 3))
		{
			#print"$raw[0],$raw[1],$raw[2]\n";
			$filtered[0][$c] = $raw[0];
			$filtered[1][$c] = $raw[1];
			$filtered[2][$c] = $raw[2];
			$c++;
		}
		$e++;
	}
	close (FILE1);
	print "Total cell count = $c\n";
	print "Looking for write file compday$day.dat\n";
	open FILEHANDLE, ">compday$day.dat" or die " --> Write File Failure! <--";
	print "Found write file compday$day.dat\n";
	for($i=0;$i<$c;$i++)
	{
		print FILEHANDLE "$filtered[0][$i] $filtered[1][$i] $filtered[2][$i]\n";
		#print "$filtered[0][$i] $filtered[1][$i] $filtered[2][$i]\n";
	}

	close FILEHANDLE;
	$day++;

}

}
```

```
#CBRW vg15  USES DERIVATIVES & ABSMAXMIN & BOTH FILTERS-5X5 MA-
& ABS ON MAXMIN SEARCH OR DIFF SEARCH!!!
#ANALYZES RESID's
use Math::Complex;#uses a built-in perl module to handle math operations
use time::local;
BasicInput();
ReadFiles();

CalcResid();

CalcFirstDervX();
CalcFirstDervY();
AnaFirstDerv();
PrintFiles();




sub BasicInput
{
        open STRESS, "stressper.dat" || die "STRESS FAILED\n";
        while(<STRESS>)
        {
                chomp($_);
                #print "$_\n";
                @raw = split(",", $_);
                $stressperiod = $raw[0];
                $MPI = $raw[1];
                #print "$stressperiod, $MPI\n";
                last;
        }
        close(STRESS);
#       opendir(DIR, ".");
#
#       @files = readdir(DIR);
#       closedir(DIR);
#       while($n <= scalar(@files))
#       {
#               print "$files[$n]          ";
#               $n++;
#               if($n%3 == 0)
#               {
```

```
#                       print "\n";
#               }
#
#       }
        print "\n -- CBRW version g15c -- Analyzes resid's -- BATCH controlled";
        $loc_tim = localtime();
        print "\n         -- Day$stressperiod -- $MPI";
        print "\n         Start $loc_tim\n";
        $gwvfile = "compday$stressperiod.dat";
        chomp($gwvfile);
        open FILE1, $gwvfile || die "File Access Failure\n";
#       print "  Name of Observed value ASCII file?\n          ";
        $surFile[$i] = "obs$stressperiod.dat";
        chomp($surFile[$i]);
        open FILE2, $surFile[0] || die "File Access Failure sur 1\n";
#       print "  Value for NO FLOW cells in tecplot ASCII file?\n   ";
        $noflow = 999;
        chomp($noflow);
#       print"   Stress Period Prefix?\n           ";
        $prefix = "Day$stressperiod";
        chomp($prefix);
#       print "  Mapping Parameter and Iterations?\n ";
#       print "file check\n\ncompfile = $gwvfile, targfile = $surFile[$i], Prefix = $prefix,
MPI = $MPI\n";
        return;
}
sub ReadFiles
{
        while(<FILE1>) #while loop to read in data from file and perfrom operations
        {
                chomp($_);
                @raw = split(" ", $_);
                $comp[0][$c] = $raw[0];
                $comp[1][$c] = $raw[1];
                $comp[2][$c] = 1;
                $comp[3][$c] = $raw[2];
                #print "$comp[0][$c], $comp[1][$c], $comp[2][$c], $comp[3][$c]\n";
                $c++;
        }
        while(<FILE2>) #while loop to read in data from file and perfrom operations
        {
                        chomp($_);
                        @raw = split(" ", $_);
```

```
                        $obs[0][$lay] = $raw[0];
                        $obs[1][$lay] = $raw[1];
                        $obs[3][$lay] = $raw[2];
                        $obs[2][$lay] = 1;
                        #print "          $obs[0][$lay],
$obs[1][$lay],$obs[2][$lay],$obs[3][$lay]\n";
                        $lay++;

        }


        return;
}

sub CalcResid
{
        print "CalcResid\n";
        $r=0,$t=0, $total=0, $RDifftotal=0;
        for($compcount=0;$compcount<$c;$compcount++)
        {
                for($obscount=0;$obscount<$lay;$obscount++)
                {

                        if(($comp[0][$compcount] == $obs[0][$obscount]) &&
($comp[1][$compcount] == $obs[1][$obscount]) && ($comp[3][$compcount] !=
$noflow))
                        {

                                $residual[0][$r] = $comp[0][$compcount];
                                $residual[1][$r] = $comp[1][$compcount];
                                $residual[2][$r] = $comp[2][$compcount];
                                $residual[3][$r] = $comp[3][$compcount]-
$obs[3][$obscount];
                                $Rtotal += $residual[3][$r];
                                if($r != 0)
                                {
                                        $RDifftotal += $residual[3][$r] - $residual[3][$r-1];
                                }
                                $r++;
                        }
                }
        }
```

```
	$Rmean = $Rtotal/$r;
	for($st=0;$st<$r;$st++)
	{
		$Rtemp += ($residual[3][$st] - $Rmean)*($residual[3][$st] - $Rmean)
	}
	$Rvar = $Rtemp/($st-1);
	$Rstd = sqrt($Rvar);
	#print " $Rstd, $Rvar, $st, $r\n";
	return();
}
sub CalcFirstDervX()
{
	$dwx=0;
	$FDcount=0;
	$FDtotal =0;
	print "\nCalcFirstDervX";
	$ystart = 100000000;
	$yend = 0;
	for($o=0;$o<$r;$o++)
	{
		if($residual[1][$o] < $ystart)
		{
			$ystart = $residual[1][$o];
			$fy = $o;
		}
		if($residual[1][$o] > $yend)
		{
			$yend = $residual[1][$o];
			$ly = $o;
		}
	}
	$ythis = $fy;
	$yclose = 0,$ydist=0;
	for($iy=$ystart;$iy<=$yend;$iy+=$ystep)
	{
		#print "$iy\n";
		$xstart = 1000000000;
		$xend = 0;
		for($o=0;$o<$r;$o++)
		{
			if(($residual[0][$o] < $xstart)&& ($residual[1][$o] == $iy))
			{
				$xstart = $residual[0][$o];
```

```
                        $fx = $o;
                }
                if(($residual[0][$o] > $xend)&& ($residual[1][$o] == $iy))
                {
                        $xend = $residual[0][$o];
                        $lx = $o;
                }
        }
        $xthis=$fx;
        $xclose=0;$xdist=0,;
        for($ix=$xstart;$ix<$xend;$ix+=$xstep)
        {
                $xdist=1000000.0;
                for($v=0;$v<$r;$v++)
                {
                        if(($residual[0][$v]-$residual[0][$xthis] < $xdist) &&
($residual[0][$v] > $residual[0][$xthis]))
                        {
                                if(($residual[1][$v] == $iy) && ($residual[0][$v] -
$residual[0][$xthis] > 0))
                                {
                                        $xdist = $residual[0][$v]-
$residual[0][$xthis] ;
                                        $xclose = $v;
                                }
                        }
                }
                $fDervx[0][$fdx] = $residual[0][$xthis];
                $fDervx[1][$fdx] = $iy;
                $fDervx[2][$fdx] = 1;
                #print "($residual[3][$xclose] -
$residual[3][$xthis])/($residual[0][$xclose]-$residual[0][$xthis])";
                $fDervx[3][$fdx] = $residual[3][$xthis];
                $FDtotal += $fDervx[3][$fdx];
                $FDcount++;
                $xstep = $residual[0][$xclose] - $residual[0][$xthis];
                $xthis = $xclose;
                $xclose = 0;
                $fdx++;

        }
        $ydist = 100000000;
        for($t=0;$t<$r;$t++)
```

```
		{
			if(($residual[1][$t] - $residual[1][$ythis] < $ydist) &&
($residual[1][$t] > $residual[1][$ythis]))
			{
				if($residual[1][$t] - $residual[1][$ythis] > 0)
				{
					$ydist = $residual[1][$t] - $residual[1][$ythis];
					$yclose = $t;
					#print "$ydist, $ythis, $yclose $residual[1][$ythis] ,
$residual[1][$yclose]    $iy\n";
				}
			}

		}
		$ystep = $residual[1][$yclose] - $residual[1][$ythis];
		print ".";
		$ythis = $yclose;
		$yclose = 0;
		if($iy == $residual[1][$ly])
		{
			$ystep = 1000000;
			#print "Last --> $residual[1][$ythis], $residual[1][$ly]\n";
		}



	}
	return();
}
sub CalcFirstDervY()
{
	print "\nCalcFirstDervY";

	$xstart = 100000000;
	$xend = 0;
	for($o=0;$o<$r;$o++)
	{
		if($residual[0][$o] < $xstart)
		{
			$xstart = $residual[0][$o];
			$fx = $o;
		}
		if($residual[0][$o] > $xend)
```

```
		{
			$xend = $residual[0][$o];
			$lx = $o;
		}
	}
	$xthis = $fx;
	$xclose = 0,$xdist=0;
	for($ix=$xstart;$ix<=$xend;$ix+=$xstep)
	{
		$ystart = 1000000000;
		$yend = 0;

		for($o=0;$o<$r;$o++)
		{
			if(($residual[1][$o] < $ystart)&& ($residual[0][$o] == $ix))
			{
				$ystart = $residual[1][$o];
				$fy = $o;

			}
			if(($residual[1][$o] > $yend)&& ($residual[0][$o] == $ix))
			{
				$yend = $residual[1][$o];
				$ly = $o;


			}
		}
		$ythis=$fy;
		$yclose=0;$ydist=0,;
		for($iy=$ystart;$iy<$yend;$iy+=$ystep)
		{

			$ydist=1000000.0;
			for($v=0;$v<$r;$v++)
			{
				if(($residual[1][$v]-$residual[1][$ythis] < $ydist) &&
($residual[1][$v] > $residual[1][$ythis]))
				{
					if(($residual[0][$v] == $ix) && ($residual[1][$v] -
$residual[1][$ythis] > 0))
					{
```

```
                                        $ydist = $residual[1][$v]-
$residual[1][$ythis] ;
                                        $yclose = $v;

                                }
                        }
                }

                $fDervy[0][$fdy] = $ix;
                $fDervy[1][$fdy] = $residual[1][$ythis];
                $fDervy[2][$fdy] = 1;
                $fDervy[3][$fdy] = $residual[3][$ythis];
                $FDtotal += $fDervy[3][$fdy];
                $FDcount++;
                $ystep = $residual[1][$yclose] - $residual[1][$ythis];
                $ythis = $yclose;
                $yclose = 0;
                $fdy++;
        }
        $xdist = 100000000;
        for($t=0;$t<$r;$t++)
        {
                if(($residual[0][$t] - $residual[0][$xthis] < $xdist) &&
($residual[0][$t] > $residual[0][$xthis]))
                {
                        if($residual[0][$t] - $residual[0][$xthis] > 0)
                        {
                                $xdist = $residual[0][$t] - $residual[0][$xthis];
                                $xclose = $t;
                                #print "$xdist, $xthis, $xclose $residual[0][$xthis] ,
$residual[0][$xclose]    $ix\n";
                        }
                }
        }
        $xstep = $residual[0][$xclose] - $residual[0][$xthis];
        $xthis = $xclose;
        print ".";
        $xclose = 0;
        if($ix == $residual[0][$lx])
        {
                $xstep = 1000000;
                #print "Last --> $residual[0][$xthis], $residual[0][$lx]\n";
        }
```

```
	}
	return();
}
sub AnaFirstDerv()
{
	#print "\nAnalyzing First Derv";

	$FDmean = $FDtotal/$FDcount;
	$FDtemp=0;
	for($fdst=0;$fdst<$fdx;$fdst++)
	{
		$FDtemp += ($fDervx[3][$fdst] - $FDmean)*($fDervx[3][$fdst] -
$FDmean);
	}
	for($fdst=0;$fdst<$fdy;$fdst++)
	{
		$FDtemp += ($fDervy[3][$fdst] - $FDmean)*($fDervy[3][$fdst] -
$FDmean);
	}
	$FDvar = $FDtemp/($FDcount-1);
	$FDstd = sqrt($FDvar);
	$shit = $fdx+$fdy;
	#print "	Use a Max/Min Threshold -->1 for Yes<--?\n	";
	$MMchoice = 0;
	chomp($MMchoice);
	if($MMchoice != 1)
	{
		$crit = .00000000000001;
	}
	else
	{
		print "\nresidual mean = $Rmean, s= $Rstd\nFD mean = $FDmean, s =
$FDstd\n	What MAX/MIN Threshold?\n	";
		$crit = <STDIN>;
		chomp($crit);
	}
	$amm=0;
	$ythis = $fDervx[1][0];
	$absMax = -1000000000;
	$absMin = 10000000000;
	for($id=0;$id<$fdx;$id++)
	{
		if(($fDervx[1][$id-3] == $ythis) && ($fDervx[1][$id+3] == $ythis))
```

```
		{
			$behind = abs($fDervx[3][$id] - $fDervx[3][$id-1]);
			$ahead = abs($fDervx[3][$id] - $fDervx[3][$id+1]);
			#print "$behind, $ahead\n";
			#print"$fDervx[0][$id], $fDervx[1][$id], $fDervx[2][$id],
$fDervx[3][$id]\n";
			if(($fDervx[3][$id] > $fDervx[3][$id-1]) && ($fDervx[3][$id] >
$fDervx[3][$id+1]) &&
				($fDervx[3][$id-1] > $fDervx[3][$id-2]) &&
($fDervx[3][$id+1] > $fDervx[3][$id+2]) &&
				($fDervx[3][$id-2] > $fDervx[3][$id-3]) &&
($fDervx[3][$id+2] > $fDervx[3][$id+3]))
			{
				if(($behind > $crit) || ($ahead > $crit))
				{
					$fmaxMinx[0][$fmmx] = $fDervx[0][$id];
					$fmaxMinx[1][$fmmx] = $fDervx[1][$id];
					$fmaxMinx[2][$fmmx] = 1;
					$fmaxMinx[3][$fmmx] = "xMax";
					#print "xMAX found at $fmaxMinx[0][$fmmx],
$fmaxMinx[1][$fmmx], $fmaxMinx[3][$fmmx]\n";
					$fmmx++;
					$e++;
					if($fDervx[3][$id] > $absMax)
					{
						$absMax = $fDervx[3][$id];
						$tempMax[0] = $fDervx[0][$id];
						$tempMax[1] = $fDervx[1][$id];
						$tempMax[2] = $fDervx[2][$id];
						$tempMax[3] = $fDervx[3][$id];
						$tempMax[4] = "XABSMAX";
						#print "XAbsMax = $fDervx[3][$id] at
$fDervx[0][$id], $fDervx[1][$id]\n";
					}
				}
			}
			else
			{
				$behind = $fDervx[3][$id-1] - $fDervx[3][$id];
				$ahead = $fDervx[3][$id+1] - $fDervx[3][$id];
				if(($fDervx[3][$id] < $fDervx[3][$id-1]) &&
($fDervx[3][$id] < $fDervx[3][$id+1])&&
```

```
                                        ($fDervx[3][$id-1] < $fDervx[3][$id-2]) &&
($fDervx[3][$id+1] < $fDervx[3][$id+2])&&
                                        ($fDervx[3][$id-2] < $fDervx[3][$id-3]) &&
($fDervx[3][$id+2] < $fDervx[3][$id+3]))
                                {
                                        if(($behind > $crit) || ($ahead > $crit))
                                        {
                                                $fmaxMinx[0][$fmmx] = $fDervx[0][$id];
                                                $fmaxMinx[1][$fmmx] = $fDervx[1][$id];
                                                $fmaxMinx[2][$fmmx] = 1;
                                                $fmaxMinx[3][$fmmx] = "xMin";
                                                #print "xMIN found at
$fmaxMinx[0][$fmmx], $fmaxMinx[1][$fmmx], $fmaxMinx[3][$fmmx]\n";
                                                $fmmx++;
                                                $e++;
                                                if($fDervx[3][$id] < $absMin)
                                                {
                                                        $absMin = $fDervx[3][$id];
                                                        $tempMin[0] = $fDervx[0][$id];
                                                        $tempMin[1] = $fDervx[1][$id];
                                                        $tempMin[2] = $fDervx[2][$id];
                                                        $tempMin[3] = $fDervx[3][$id];
                                                        $tempMin[4] = "XABSMIN";
                                                        #print "XAbsMax = $fDervx[3][$id]
at $fDervx[0][$id], $fDervx[1][$id]\n";
                                                }
                                        }
                                }
                        }

                }
                else
                {
                        #print "UPDATE-- $fDervx[0][$id], $fDervx[1][$id],
$fDervx[2][$id], $fDervx[3][$id]\n";
                        $ythis = $fDervx[1][$id];
                        if($absMax > -100000)
                        {
                                $AMM[0][$amm] = $tempMax[0];
                                $AMM[1][$amm] = $tempMax[1];
                                $AMM[2][$amm] = $tempMax[2];
                                $AMM[3][$amm] = $tempMax[3];
                                $AMM[4][$amm] = $tempMax[4];
```

```
                        $amm++;
                    }
                    if($absMin < 10000000)
                    {
                        $AMM[0][$amm] = $tempMin[0];
                        $AMM[1][$amm] = $tempMin[1];
                        $AMM[2][$amm] = $tempMin[2];
                        $AMM[3][$amm] = $tempMin[3];
                        $AMM[4][$amm] = $tempMin[4];
                        $amm++;
                    }

                    $absMax = -10000000000;
                    $absMin = 10000000000;

                }
            }
            $xthis = $fDervy[0][0];
            $absMax = -1000000000;
            $absMin = 10000000000;
            for($id=0;$id<$fdy;$id++)
            {
                if(($fDervy[0][$id-2] == $xthis) && ($fDervy[0][$id+2] == $xthis))
                {
                    $behind = abs($fDervy[3][$id] - $fDervy[3][$id-1]);
                    $ahead = abs($fDervy[3][$id] - $fDervy[3][$id+1]);
                    #print"$fDervy[0][$id], $fDervy[1][$id], $fDervy[2][$id],
$fDervy[3][$id]\n";
                    if(($fDervy[3][$id] > $fDervy[3][$id-1]) && ($fDervy[3][$id] >
$fDervy[3][$id+1]) &&
                        ($fDervy[3][$id-1] > $fDervy[3][$id-2]) &&
($fDervy[3][$id+1] > $fDervy[3][$id+2])&&
                        ($fDervy[3][$id-2] > $fDervy[3][$id-3]) &&
($fDervy[3][$id+2] > $fDervy[3][$id+3]))
                    {
                        if(($behind > $crit) || ($ahead > $crit))
                        {
                            $fmaxMiny[0][$fmmy] = $fDervy[0][$id];
                            $fmaxMiny[1][$fmmy] = $fDervy[1][$id];
                            $fmaxMiny[2][$fmmy] = 1;
                            $fmaxMiny[3][$fmmy] = "yMax";
                            #print "yMAX found at $fmaxMiny[0][$fmmy],
$fmaxMiny[1][$fmmy],$fmaxMiny[3][$fmmy]\n";
```

```
                                        $fmmy++;
                                        $e++;
                                        if($fDervy[3][$id] > $absMax)
                                        {
                                                $absMax = $fDervy[3][$id];
                                                $tempMax[0] = $fDervy[0][$id];
                                                $tempMax[1] = $fDervy[1][$id];
                                                $tempMax[2] = $fDervy[2][$id];
                                                $tempMax[3] = $fDervy[3][$id];
                                                $tempMax[4] = "YABSMAX";
                                                #print "YAbsMax = $fDervy[3][$id] at
$fDervy[0][$id], $fDervy[1][$id]\n";
                                        }
                                }
                        }
                        else
                        {
                                $behind = $fDervy[3][$id-1] - $fDervy[3][$id];
                                $ahead = $fDervy[3][$id+1] - $fDervy[3][$id];
                                if(($fDervy[3][$id] < $fDervy[3][$id-1]) &&
($fDervy[3][$id] < $fDervy[3][$id+1])&&
                                        ($fDervy[3][$id-1] < $fDervy[3][$id-2]) &&
($fDervy[3][$id+1] < $fDervy[3][$id+2])&&
                                        ($fDervy[3][$id-2] < $fDervy[3][$id-3]) &&
($fDervy[3][$id+2] < $fDervy[3][$id+3]))
                                {
                                        if(($behind > $crit) || ($ahead > $crit))
                                        {
                                                $fmaxMiny[0][$fmmy] = $fDervy[0][$id];
                                                $fmaxMiny[1][$fmmy] = $fDervy[1][$id];
                                                $fmaxMiny[2][$fmmy] = 1;
                                                $fmaxMiny[3][$fmmy] = "yMin";
                                                #print "yMIN found at
$fmaxMiny[0][$fmmy], $fmaxMiny[1][$fmmy],$fmaxMiny[3][$fmmy]\n";
                                                $fmmy++;
                                                $e++;
                                                if($fDervy[3][$id] < $absMin)
                                                {
                                                        $absMin = $fDervy[3][$id];
                                                        $tempMin[0] = $fDervy[0][$id];
                                                        $tempMin[1] = $fDervy[1][$id];
                                                        $tempMin[2] = $fDervy[2][$id];
                                                        $tempMin[3] = $fDervy[3][$id];
```

```
                                                        $tempMin[4] = "YABSMIN";
                                                        #print "YAbsMin = $fDervy[3][$id]
at $fDervy[0][$id], $fDervy[1][$id]\n";
                                                }
                                        }
                                }
                        }

                }
                else
                {
                        #print "UPDATE-- $fDervy[0][$id], $fDervy[1][$id],
$fDervy[2][$id], $fDervy[3][$id]\n";
                        $xthis = $fDervy[0][$id];
                        if($absMax > -100000)
                        {
                                $AMM[0][$amm] = $tempMax[0];
                                $AMM[1][$amm] = $tempMax[1];
                                $AMM[2][$amm] = $tempMax[2];
                                $AMM[3][$amm] = $tempMax[3];
                                $AMM[4][$amm] = $tempMax[4];
                                $amm++;
                        }
                        if($absMin < 10000000)
                        {
                                $AMM[0][$amm] = $tempMin[0];
                                $AMM[1][$amm] = $tempMin[1];
                                $AMM[2][$amm] = $tempMin[2];
                                $AMM[3][$amm] = $tempMin[3];
                                $AMM[4][$amm] = $tempMin[4];
                                $amm++;
                        }
                        $absMax = -10000000000;
                        $absMin = 10000000000;

                }
        }
        $l = $fmmy+$fmmx;
        print "\n\nFound $l MaxMin";
        return();
}
sub CalcResidDiff()
{
```

```
	#print "	Use a Differnece in Residual Threshold --> 1 for Yes<--\n   ";
	$DRchoice = 1;
	$RDiffMean = $RDifftotal/($r-1);
	chomp($DRchoice);
	if($DRchoice != 1)
	{
		$DRcrit = .000000000001;
	}
	else
	{
		#print "\nAverage Resid Diff = $RDiffMean, Resid s= $Rstd\nFD Mean =
$FDmean, s = $FDstd\n	What Diff Threshold?\n	";
		$DRcrit = 1.5;
		chomp($DRcrit);
	}

	$ystart = 100000000;
	$yend = 0;
	#print "\nSorting X -->\n";
	for($o=0;$o<$r;$o++)
	{

		if($residual[1][$o] < $ystart)
		{
			$ystart = $residual[1][$o];
			$fy = $o;
		}
		if($residual[1][$o] > $yend)
		{
			$yend = $residual[1][$o];
			$ly = $o;
		}
	}

	$ythis = $fy;
	$yclose = 0,$ydist=0;
	for($iy=$ystart;$iy<=$yend;$iy+=$ystep)
	{
		#print "$iy\n";
		$xstart = 1000000000;
		$xend = 0;
		for($o=0;$o<$r;$o++)
		{
```

```
                        if(($residual[0][$o] < $xstart)&& ($residual[1][$o] == $iy))
                        {
                                $xstart = $residual[0][$o];
                                $fx = $o;
                        }
                        if(($residual[0][$o] > $xend)&& ($residual[1][$o] == $iy) )
                        {
                                $xend = $residual[0][$o];
                                $lx = $o;
                        }
                }
                $xthis=$fx;
                $xclose=0;$xdist=0,;
                for($ix=$xstart;$ix<$xend;$ix+=$xstep)
                {
                        #print "$ix\n";
                        $xdist=1000000.0;
                        for($v=0;$v<$r;$v++)
                        {

                                if(($residual[0][$v]-$residual[0][$xthis] < $xdist) &&
($residual[0][$v] > $residual[0][$xthis]))
                                {
                                        if($residual[1][$v] == $iy)
                                        {
                                                $xdist = $residual[0][$v]-
$residual[0][$xthis] ;
                                                $xclose = $v;


                                        }
                                }

                        }

                        $xsort[0][$xs] = $residual[0][$xthis];
                        $xsort[1][$xs] = $residual[1][$xthis];
                        $xsort[2][$xs] = $residual[2][$xthis];
                        $xsort[3][$xs] = $residual[3][$xthis];
                        #print
"$xsort[0][$xs],$xsort[1][$xs],$xsort[2][$xs],$xsort[3][$xs]\n";
                        $xstep = $residual[0][$xclose] - $residual[0][$xthis];
                        $xthis = $xclose;
```

```
                        $xclose = 0;
                        $xs++;
                        if($residual[0][$xthis] == $residual[0][$lx])
                        {
                                #print "last x --> $residual[0][$lx]\n";
                                $xsort[0][$xs] = $residual[0][$lx];
                                $xsort[1][$xs] = $residual[1][$lx];
                                $xsort[2][$xs] = $residual[2][$lx];
                                $xsort[3][$xs] = $residual[3][$lx];
                                $xs++;
                                $xstep = 10000000;


                        }
                }
                $ydist = 100000000;
                for($t=0;$t<$r;$t++)
                {
                        if(($residual[1][$t] - $residual[1][$ythis] < $ydist) &&
($residual[1][$t] > $residual[1][$ythis]))
                        {
                                if($residual[1][$t] - $residual[1][$ythis] > 0)
                                {
                                        $ydist = $residual[1][$t] - $residual[1][$ythis];
                                        $yclose = $t;
                                        #print "$ydist, $ythis, $yclose $residual[1][$ythis] ,
$residual[1][$yclose]    $iy\n";
                                }
                        }

                }
                $ystep = $residual[1][$yclose] - $residual[1][$ythis];
                $ythis = $yclose;
                $yclose = 0;
                if($iy == $residual[1][$ly])
                {
                        $ystep = 1000000;
                        #print "Last --> $residual[1][$ythis], $residual[1][$ly]\n";
                }
        }

        #print "\nSearching X Resid Diff's -->\n";
        $dr=0;
```

```
	$ythis = $xsort[1][0];
	for($t=0;$t<$xs;$t++)
	{
		if(($xsort[1][$t-1] == $ythis) && ($xsort[1][$t+1] == $ythis))
		{

			if(abs($xsort[3][$t] - $xsort[3][$t-1]) > $DRcrit)
			{
				$DiffResid[0][$dr] = $xsort[0][$t];
				$DiffResid[1][$dr] = $xsort[1][$t];
				$DiffResid[2][$dr] = $xsort[2][$t];
				$DiffResid[3][$dr] = abs($xsort[3][$t]-$xsort[3][$t-1]);
				$DiffResid[4][$dr] = "Xdiff";
				#print "$DiffResid[0][$dr], $DiffResid[1][$dr],
$DiffResid[2][$dr], $DiffResid[3][$dr]\n";
				$dr++;
			}

		}
		else
		{
			if($xsort[1][$t+1] != $ythis)
			{
				#print "Updating $xsort[0][$t] --> $xsort[0][$t+1],
$xsort[1][$t] --> $xsort[1][$t+1]\n";
				$ythis = $xsort[1][$t+1];

			}
		}
	}


	#print "\nSorting Y --> \n";
	$xstart = 100000000;
	$xend = 0;
	for($o=0;$o<$xs;$o++)
	{

		if($xsort[0][$o] < $xstart)
		{
			$xstart = $xsort[0][$o];
			$fx = $o;
		}
```

```
		if($xsort[0][$o] > $xend)
		{
			$xend = $xsort[0][$o];
			$lx = $o;
		}
	}
	#print "\n$xstart to $xend\n";
	$xthis = $fx;
	$xclose = 0,$xdist=0;
	for($ix=$xstart;$ix<=$xend;$ix+=$xstep)
	{
		#print "$iy\n";
		$ystart = 1000000000;
		$yend = 0;
		for($o=0;$o<$xs;$o++)
		{
			if(($xsort[1][$o] < $ystart)&& ($xsort[0][$o] == $ix))
			{
				$ystart = $xsort[1][$o];
				$fy = $o;
			}
			if(($xsort[1][$o] > $yend)&& ($xsort[0][$o] == $ix) )
			{
				$yend = $xsort[1][$o];
				$ly = $o;
			}
		}
		$ythis=$fy;
		$yclose=0;$ydist=0,;
		for($iy=$ystart;$iy<$yend;$iy+=$ystep)
		{
			#print "$ix\n";
			$ydist=1000000.0;
			for($v=0;$v<$xs;$v++)
			{

				if(($xsort[1][$v]-$xsort[1][$xthis] < $ydist) &&
($xsort[1][$v] > $xsort[1][$ythis]))
				{
					if($xsort[0][$v] == $ix)
					{
						$ydist = $xsort[1][$v]-$xsort[1][$ythis] ;
						$yclose = $v;
```

```
                                        }
                                }

                        }

                        $ysort[0][$ys] = $xsort[0][$ythis];
                        $ysort[1][$ys] = $xsort[1][$ythis];
                        $ysort[2][$ys] = $xsort[2][$ythis];
                        $ysort[3][$ys] = $xsort[3][$ythis];
                        #print
"$ysort[0][$ys],$ysort[1][$ys],$ysort[2][$ys],$ysort[3][$ys]\n";
                        $ystep = $xsort[1][$yclose] - $xsort[1][$ythis];
                        $ythis = $yclose;
                        $yclose = 0;
                        $ys++;
                        if($xsort[1][$ythis] == $xsort[1][$ly])
                        {
                                #print "last y --> $sortedx[1][$ly]\n";
                                $ysort[0][$ys] = $xsort[0][$ly];
                                $ysort[1][$ys] = $xsort[1][$ly];
                                $ysort[2][$ys] = $xsort[2][$ly];
                                $ysort[3][$ys] = $xsort[3][$ly];
                                $ys++;



                        }
                }
                $xdist = 100000000;
                for($t=0;$t<$xs;$t++)
                {
                        if(($xsort[0][$t] - $xsort[0][$xthis] < $xdist) && ($xsort[0][$t] >
$xsort[0][$xthis]))
                        {
                                if($xsort[0][$t] - $xsort[0][$xthis] > 0)
                                {
                                        $xdist = $xsort[0][$t] - $xsort[0][$xthis];
                                        $xclose = $t;

                                }
                        }
```

```
		}
		$xstep = $xsort[0][$xclose] - $xsort[0][$xthis];
		$xthis = $xclose;
		$xclose = 0;
		if($ix == $xsort[0][$lx])
		{
			$xstep = 1000000;
			#print "Last --> $xsort[0][$xthis], $xsort[0][$lx]\n";
		}
	}
	#print "\nSearching  Y Residual Diffs-->\n";
	$xthis = $ysort[0][0];
	for($t=0;$t<$ys;$t++)
	{
		if(($ysort[0][$t-1] == $xthis) && ($ysort[0][$t+1] == $xthis))
		{

			if(abs($ysort[3][$t] - $ysort[3][$t-1]) > $DRcrit)
			{
				$DiffResid[0][$dr] = $ysort[0][$t];
				$DiffResid[1][$dr] = $ysort[1][$t];
				$DiffResid[2][$dr] = $ysort[2][$t];
				$DiffResid[3][$dr] = abs($ysort[3][$t]-$ysort[3][$t-1]);
				$DiffResid[4][$dr] = "Ydiff";
				#print "$DiffResid[0][$dr], $DiffResid[1][$dr],
$DiffResid[2][$dr], $DiffResid[3][$dr]\n";
				$dr++;
			}
		}
		else
		{
			if($ysort[0][$t+1] != $xthis)
			{
				#print "Updating $ysort[0][$t] --> $ysort[0][$t+1],
$ysort[1][$t] --> $ysort[1][$t+1]\n";
				$xthis = $ysort[0][$t+1];

			}
		}
	}
	print "\nFound $dr Residual Diffs";
	return();
```

```
}
sub DurbinWatson()
{
	$dwtx=0;
	#print "\nDurbinWatson - X";
	$ythis = $DurbWatx[1][0];
	for($dw=0;$dw<$dwx;$dw++)
	{
		#print "
$DurbWatx[0][$dw],$DurbWatx[1][$dw],$DurbWatx[2][$dw],$DurbWatx[3][$dw]\n";
		if($DurbWatx[1][$dw-1] != $ythis)
		{
			#print "
$DurbWatx[0][$dw],$DurbWatx[1][$dw],$DurbWatx[2][$dw],$DurbWatx[3][$dw] --
Starting\n";
			#$totSum += ($DurbWatx[3][$dw]*$DurbWatx[3][$dw]);
			$xstart = $DurbWatx[0][$dw];
			#print " Starting --> $totSum, $eSum\n";
		}
		if($DurbWatx[1][$dw+1] != $ythis)
		{
			$totSum += ($DurbWatx[3][$dw]*$DurbWatx[3][$dw]);
			$eSum += (($DurbWatx[3][$dw]-$DurbWatx[3][$dw-
1])*($DurbWatx[3][$dw]-$DurbWatx[3][$dw-1]));
			$DWX[0][$dwtx] = "X";
			$DWX[1][$dwtx] = $DurbWatx[1][$dw];
			$DWX[2][$dwtx] = 1;
			$DWX[3][$dwtx] = $eSum/$totSum;
			#print "
$DurbWatx[0][$dw],$DurbWatx[1][$dw],$DurbWatx[2][$dw],$DurbWatx[3][$dw] --
Ending\n";
			#print "Writing --> $totSum, $eSum --> $DWX[0][$dwtx],
$DWX[1][$dwtx], $DWX[2][$dwtx], $DWX[3][$dwtx]\n";
			$dwtx++;
			$eSum=0;
			$totSum=0;
			$ythis = $DurbWatx[1][$dw+1];
		}
		else
		{
			if(($DurbWatx[1][$dw-1] == $ythis) && ($DurbWatx[1][$dw+1]
== $ythis))
			{
```

```
                                #print "
$DurbWatx[0][$dw],$DurbWatx[1][$dw],$DurbWatx[2][$dw],$DurbWatx[3][$dw] --
Adding\n";
                                $totSum += ($DurbWatx[3][$dw]*$DurbWatx[3][$dw]);
                                $eSum += (($DurbWatx[3][$dw]-$DurbWatx[3][$dw-
1])*($DurbWatx[3][$dw]-$DurbWatx[3][$dw-1]));
                                #print "Adding --> $totSum, $eSum\n";
                        }
                }

        }
        $dwty=0;
        #print "\nDurbinWatson - Y";
        $xthis = $DurbWaty[0][0];
        for($dw=0;$dw<$dwy;$dw++)
        {
                if($DurbWaty[0][$dw-1] != $xthis)
                {
                        #print "
$DurbWaty[0][$dw],$DurbWaty[1][$dw],$DurbWaty[2][$dw],$DurbWaty[3][$dw] --
Starting\n";
                        #$totSum += ($DurbWaty[3][$dw]*$DurbWaty[3][$dw]);
                        $ystart = $DurbWaty[1][$dw];
                        #print " Starting --> $totSum, $eSum\n";
                }
                if($DurbWaty[0][$dw+1] != $xthis)
                {
                        $totSum += ($DurbWaty[3][$dw]*$DurbWaty[3][$dw]);
                        $eSum += (($DurbWaty[3][$dw]-$DurbWaty[3][$dw-
1])*($DurbWaty[3][$dw]-$DurbWaty[3][$dw-1]));
                        $DWY[0][$dwty] = "Y";
                        $DWY[1][$dwty] = $DurbWaty[0][$dw];
                        $DWY[2][$dwty] = 1;
                        $DWY[3][$dwty] = $eSum/$totSum;
                        #print "
$DurbWaty[0][$dw],$DurbWaty[1][$dw],$DurbWaty[2][$dw],$DurbWaty[3][$dw] --
Ending\n";
                        #print "Writing --> $totSum, $eSum --> $DWY[0][$dwty],
$DWY[1][$dwty], $DWY[2][$dwty], $DWY[3][$dwty]\n";
                        $dwty++;
                        $eSum=0;
                        $totSum=0;
                        $xthis = $DurbWaty[0][$dw+1];
```

```
                }
                else
                {
                        if(($DurbWaty[0][$dw-1] == $xthis) && ($DurbWaty[0][$dw+1]
== $xthis))
                        {
                                #print "
$DurbWaty[0][$dw],$DurbWaty[1][$dw],$DurbWaty[2][$dw],$DurbWaty[3][$dw] --
Adding\n";
                                $totSum += ($DurbWaty[3][$dw]*$DurbWaty[3][$dw]);
                                $eSum += (($DurbWaty[3][$dw]-$DurbWaty[3][$dw-
1])*($DurbWaty[3][$dw]-$DurbWaty[3][$dw-1]));
                                #print "Adding --> $totSum, $eSum\n";
                        }
                }
        }
        return();
}
sub PrintFiles()
{
        print" \nPrintFiles\n";


        open FILEHANDLE1, ">$MPI-$prefix-15-residual.dat" || die "NO Worky\n";
        open FILEHANDLE2, ">$MPI-$prefix-15-fDervx.dat" || die "NO Worky\n";
        open FILEHANDLE3, ">$MPI-$prefix-15-fDervy.dat" || die "NO Worky\n";
        open FILEHANDLE4, ">$MPI-$prefix-15-MaxMin.dat" || die "NO Worky\n";
        open FILEHANDLE5, ">$MPI-$prefix-15-DurbinWatson.dat" || die "NO
Worky\n";
        open FILEHANDLE6, ">$MPI-$prefix-15-DiffResid.dat" || die "NO Worky\n";
        open FILEHANDLE7, ">$MPI-$prefix-15-AbsMaxMin.dat" || die "NO
Worky\n";
        for($i=0;$i<$r;$i++)
        {
                print
FILEHANDLE1"$residual[0][$i],$residual[1][$i],$residual[2][$i],$residual[3][$i]\n";

        }
        for($i=0;$i<$fdx;$i++)
        {
                print FILEHANDLE2
"$fDervx[0][$i],$fDervx[1][$i],$fDervx[2][$i],$fDervx[3][$i]\n";
        }
```

```
	for($i=0;$i<$fdy;$i++)
	{
		print FILEHANDLE3
"$fDervy[0][$i],$fDervy[1][$i],$fDervy[2][$i],$fDervy[3][$i]\n";
	}
	for($i=0;$i<$fmmx;$i++)
	{
		print FILEHANDLE4
"$fmaxMinx[0][$i],$fmaxMinx[1][$i],$fmaxMinx[3][$i]\n";
	}
	for($i=0;$i<$fmmy;$i++)
	{
		print FILEHANDLE4
"$fmaxMiny[0][$i],$fmaxMiny[1][$i],$fmaxMiny[3][$i]\n";
	}

	for($i=0;$i<$amm;$i++)
	{
		print FILEHANDLE7 "$AMM[0][$i],$AMM[1][$i], $AMM[2][$i],
$AMM[3][$i], $AMM[4][$i]\n";
	}
	open(STRESS, ">stressper.dat") || "stress re-write failed\n";
	$stressperiod++;
	print STRESS "$stressperiod,$MPI";
	open(FD, ">>$MPI-TotalMaxMin.dat") or die("Couldn't open
TotalMaxMin.dat\n") ;
	for($i=0;$i<$fmmx;$i++)
	{
		print FD "$fmaxMinx[0][$i],$fmaxMinx[1][$i],$fmaxMinx[3][$i]\n";
	}
	for($i=0;$i<$fmmy;$i++)
	{
		print FD "$fmaxMiny[0][$i],$fmaxMiny[1][$i],$fmaxMiny[3][$i]\n";
	}
	open(FE, ">>$MPI-TotalResidDiff.dat") or die("Couldn't open
TotalResidDiff.dat\n") ;
	for($i=0;$i<$dr;$i++)
	{
		print FE "$DiffResid[0][$i],$DiffResid[1][$i], $DiffResid[2][$i],
$DiffResid[3][$i], $DiffResid[4][$i]\n";
	}
	open TOTRES,"$MPI-TotalResidual.dat" || "File Failure TotalResidual\n";
	$totres=0;
```

```
	while(<TOTRES>)
	{
		@raw = split(",",$_);
		$AbsResid[0][$totres] = $raw[0];
		$AbsResid[1][$totres] = $raw[1];
		$AbsResid[2][$totres] = $raw[2];
		$totres++;
	}
	close TOTRES;
	open TOTRESID,">$MPI-TotalResidual.dat" || "FILE FAILURE TOTAL
RESIDUAL WRITE\n";
	for($nr=0;$nr<$r;$nr++)
	{
		if((($AbsResid[0][$nr] == $residual[0][$nr]) && ($AbsResid[1][$nr] ==
$residual[1][$nr])) || (($AbsResid[0][$nr] = " ") &&
		($AbsResid[1][$nr] = " ")))
		{
			$AbsResid[0][$nr] = $residual[0][$nr];
			$AbsResid[1][$nr] = $residual[1][$nr];
			$AbsResid[2][$nr] += abs($residual[3][$nr]);
			#print "$AbsResid[2][$nr]=$residual[3][$nr]\n";
			print TOTRESID
"$AbsResid[0][$nr],$AbsResid[1][$nr],$AbsResid[2][$nr]\n";

		}
	}
	close FD;
	close FE;
	close TOTRES;
	close STRESS;
	close FILEHANDLE1;
	close FILEHANDLE2;
	close FILEHANDLE3;
	close FILEHANDLE4;
	close FILEHANDLE5;
	close FILEHANDLE6;
	close FILEHANDLE7;
	$loc_tim = localtime();
	print "Finish $loc_tim\n";
	return();
}
```

```
#TOTAL RESIDUAL ANALYZER  MAXMIN SEARCH OR DIFF SEARCH!!!
#ANALYZES TOTAL RESID's
use Math::Complex;#uses a built-in perl module to handle math operations

#BasicInput();
#ReadFiles();
CalcResid();
FilterResid();
CalcFirstDervX();
CalcFirstDervY();
AnaFirstDerv();
PrintFiles();




sub BasicInput
{
        opendir(DIR, ".");
        @files = readdir(DIR);
        closedir(DIR);
        while($n <= scalar(@files))
        {
                print "$files[$n]          ";
                $n++;
                if($n%3 == 0)
                {
                        print "\n";
                }

        }
        print "\n\n -- CBRW version g15 -- Analyzes resid's -- \n\n";
        print "\n\n\n    Name of Computed value ASCII file?\n       ";
        $gwvfile = <STDIN>;
        chomp($gwvfile);
        open FILE1, $gwvfile || die "File Access Failure\n";
        print "  Name of Observed value ASCII file?\n         ";
        $surFile[$i] = <STDIN>;
        chomp($surFile[$i]);
        open FILE2, $surFile[0] || die "File Access Failure sur 1\n";
        print "  Value for NO FLOW cells in tecplot ASCII file?\n   ";
        $noflow = 10000;
```

```
	chomp($noflow);
	print"   Stress Period Prefix?\n           ";
	$prefix = <STDIN>;
	chomp($prefix);
	print "  Mapping Parameter and Iterations?\n ";
	$MPI = <STDIN>;
	chomp($MPI);
	return;
}
sub ReadFiles
{
	while(<FILE1>) #while loop to read in data from file and perfrom operations
	{
		chomp($_);
		@raw = split(" ", $_);
		$comp[0][$c] = $raw[0];
		$comp[1][$c] = $raw[1];
		$comp[2][$c] = 1;
		$comp[3][$c] = $raw[2];
		#print " single layer -->$comp[0][$c], $comp[1][$c], $comp[2][$c],
$comp[3][$c]\n";
		$c++;
	}
	while(<FILE2>) #while loop to read in data from file and perfrom operations
	{
			chomp($_);
			@raw = split(" ", $_);

			$obs[0][$lay] = $raw[0];
			$obs[1][$lay] = $raw[1];
			$obs[3][$lay] = $raw[2];
			$obs[2][$lay] = 1;
			$lay++;

	}


	return;
}

sub CalcResid
{
	print "\nCalcResid\n";
```

```
	open STRESS, "stressper.dat" || die "STRESS FAILED\n";
	while(<STRESS>)
	{
		chomp($_);
		#print "$_\n";
		@raw = split(",", $_);
		$stressperiod = $raw[0];
		$MPI = $raw[1];
		#print "$stressperiod, $MPI\n";
		last;
	}
	close(STRESS);
	$r=0,$t=0, $total=0, $RDifftotal=0;
	open FILE10,"$MPI-TotalResidual.dat" || die "File access Failure";
	while(<FILE10>)
	{
		@raw = split(",", $_);

		$residual[0][$r] = $raw[0];
		$residual[1][$r] = $raw[1];
		$residual[2][$r] = 1;
		$residual[3][$r] = $raw[2];
		$Rtotal += $residual[3][$r];
		if($r != 0)
		{
			$RDifftotal += $residual[3][$r] - $residual[3][$r-1];
		}
		$r++;
	}
	$Rmean = $Rtotal/$r;
	for($st=0;$st<$r;$st++)
	{
		$Rtemp += ($residual[3][$st] - $Rmean)*($residual[3][$st] - $Rmean)
	}
	$Rvar = $Rtemp/($st-1);
	$Rstd = sqrt($Rvar);
	#print " $Rstd, $Rvar, $st, $r\n";
	return();
}
sub FilterResid()
{
	print "  Use Filter? -->1 for Gaussin, 2 for Moving Average<--\n     ";
	$fchoice = 0;
```

```
chomp($fchoice);
if(($fchoice != 1) && ($fchoice != 2))
{
	return();
}
else
{
	print "\nFiltering Residual\n";

	$ystart = 100000000;
	$yend = 0;
	print "\nSorting X -->\n";
	for($o=0;$o<$r;$o++)
	{

		if($residual[1][$o] < $ystart)
		{
			$ystart = $residual[1][$o];
			$fy = $o;
		}
		if($residual[1][$o] > $yend)
		{
			$yend = $residual[1][$o];
			$ly = $o;
		}
	}

	$ythis = $fy;
	$yclose = 0,$ydist=0;
	for($iy=$ystart;$iy<=$yend;$iy+=$ystep)
	{
		#print "$iy\n";
		$xstart = 1000000000;
		$xend = 0;
		for($o=0;$o<$r;$o++)
		{
			if(($residual[0][$o] < $xstart)&& ($residual[1][$o] ==
$iy))
			{
				$xstart = $residual[0][$o];
				$fx = $o;
			}
```

```
                        if(($residual[0][$o] > $xend)&& ($residual[1][$o] == $iy)
)
                        {
                                $xend = $residual[0][$o];
                                $lx = $o;
                        }
                }
                $xthis=$fx;
                $xclose=0;$xdist=0,;
                for($ix=$xstart;$ix<$xend;$ix+=$xstep)
                {
                        #print "$ix\n";
                        $xdist=1000000.0;
                        for($v=0;$v<$r;$v++)
                        {

                                if(($residual[0][$v]-$residual[0][$xthis] < $xdist)
&& ($residual[0][$v] > $residual[0][$xthis]))
                                {
                                        if($residual[1][$v] == $iy)
                                        {
                                                $xdist = $residual[0][$v]-
$residual[0][$xthis] ;
                                                $xclose = $v;


                                        }
                                }

                        }

                        $xsort[0][$xs] = $residual[0][$xthis];
                        $xsort[1][$xs] = $residual[1][$xthis];
                        $xsort[2][$xs] = $residual[2][$xthis];
                        $xsort[3][$xs] = $residual[3][$xthis];
                        #print
"$xsort[0][$xs],$xsort[1][$xs],$xsort[2][$xs],$xsort[3][$xs]\n";
                        $xstep = $residual[0][$xclose] - $residual[0][$xthis];
                        $xthis = $xclose;
                        $xclose = 0;
                        $xs++;
                        if($residual[0][$xthis] == $residual[0][$lx])
                        {
```

```
                                        #print "last x --> $residual[0][$lx]\n";
                                        $xsort[0][$xs] = $residual[0][$lx];
                                        $xsort[1][$xs] = $residual[1][$lx];
                                        $xsort[2][$xs] = $residual[2][$lx];
                                        $xsort[3][$xs] = $residual[3][$lx];
                                        $xs++;
                                        $xstep = 10000000;


                                }
                        }
                        $ydist = 100000000;
                        for($t=0;$t<$r;$t++)
                        {
                                if(($residual[1][$t] - $residual[1][$ythis] < $ydist) &&
($residual[1][$t] > $residual[1][$ythis]))
                                {
                                        if($residual[1][$t] - $residual[1][$ythis] > 0)
                                        {
                                                $ydist = $residual[1][$t] -
$residual[1][$ythis];
                                                $yclose = $t;
                                                #print "$ydist, $ythis, $yclose
$residual[1][$ythis] , $residual[1][$yclose]    $iy\n";
                                        }
                                }

                        }
                        $ystep = $residual[1][$yclose] - $residual[1][$ythis];
                        $ythis = $yclose;
                        $yclose = 0;
                        if($iy == $residual[1][$ly])
                        {
                                $ystep = 1000000;
                                #print "Last --> $residual[1][$ythis], $residual[1][$ly]\n";
                        }
                }
                if($fchoice == 1)
                {
                        print "\nFiltering X --> Gaussian\n";
                        $ythis = $xsort[1][0];
                        for($t=0;$t<$xs;$t++)
                        {
```

```
                                if(($xsort[1][$t-3] == $ythis) && ($xsort[1][$t+3] ==
$ythis))
                                {
                                        $sortedx[0][$sdx] = $xsort[0][$t];
                                        $sortedx[1][$sdx] = $xsort[1][$t];
                                        $sortedx[2][$sdx] = $xsort[2][$t];
                                        $sortedx[3][$sdx] = ($xsort[3][$t-
3]*.006)+($xsort[3][$t-2]*.061)+($xsort[3][$t-1]*.242)+($xsort[3][$t]*.383)

        +($xsort[3][$t+3]*.006)+($xsort[3][$t+2]*.061)+($xsort[3][$t+1]*.242);
                                        #print
"$sortedx[0][$sdx],$sortedx[1][$sdx],$sortedx[3][$sdx]\n";
                                        $sdx++;
                                }
                                else
                                {
                                        if($xsort[1][$t+1] != $ythis)
                                        {
                                                #print "Updating $xsort[0][$t] -->
$xsort[0][$t+1], $xsort[1][$t] --> $xsort[1][$t+1]\n";
                                                $ythis = $xsort[1][$t+1];

                                        }
                                }
                        }
                }
                if($fchoice == 2)
                {
                        print "\nFiltering X --> moving average\n";
                        $ythis = $xsort[1][0];
                        for($t=0;$t<$xs;$t++)
                        {
                                if(($xsort[1][$t-2] == $ythis) && ($xsort[1][$t+2] ==
$ythis))
                                {
                                        $sortedx[0][$sdx] = $xsort[0][$t];
                                        $sortedx[1][$sdx] = $xsort[1][$t];
                                        $sortedx[2][$sdx] = $xsort[2][$t];
                                        $sortedx[3][$sdx] = ($xsort[3][$t-
2]*.125)+($xsort[3][$t-1]*.125)+($xsort[3][$t]*.5)

        +($xsort[3][$t+1]*.125)+($xsort[3][$t+2]*.125);
```

```
                                        #print
"$sortedx[0][$sdx],$sortedx[1][$sdx],$sortedx[3][$sdx]\n";
                                        $sdx++;
                                }
                                else
                                {
                                        if($xsort[1][$t+1] != $ythis)
                                        {
                                                #print "Updating $xsort[0][$t] -->
$xsort[0][$t+1], $xsort[1][$t] --> $xsort[1][$t+1]\n";
                                                $ythis = $xsort[1][$t+1];

                                        }
                                }
                        }
                }

                print "\nSorting Y --> \n";
                $xstart = 100000000;
                $xend = 0;
                for($o=0;$o<$sdx;$o++)
                {

                        if($sortedx[0][$o] < $xstart)
                        {
                                $xstart = $sortedx[0][$o];
                                $fx = $o;
                        }
                        if($sortedx[0][$o] > $xend)
                        {
                                $xend = $sortedx[0][$o];
                                $lx = $o;
                        }
                }
                #print "\n$xstart to $xend\n";
                $xthis = $fx;
                $xclose = 0,$xdist=0;
                for($ix=$xstart;$ix<=$xend;$ix+=$xstep)
                {
                        #print "$iy\n";
                        $ystart = 1000000000;
                        $yend = 0;
                        for($o=0;$o<$sdx;$o++)
```

```
                {
                        if(($sortedx[1][$o] < $ystart)&& ($sortedx[0][$o] == $ix))
                        {
                                $ystart = $sortedx[1][$o];
                                $fy = $o;
                        }
                        if(($sortedx[1][$o] > $yend)&& ($sortedx[0][$o] == $ix) )
                        {
                                $yend = $sortedx[1][$o];
                                $ly = $o;
                        }
                }
                $ythis=$fy;
                $yclose=0;$ydist=0,;
                for($iy=$ystart;$iy<$yend;$iy+=$ystep)
                {
                        #print "$ix\n";
                        $ydist=1000000.0;
                        for($v=0;$v<$sdx;$v++)
                        {

                                if(($sortedx[1][$v]-$sortedx[1][$xthis] < $ydist)
&& ($sortedx[1][$v] > $sortedx[1][$ythis]))
                                {
                                        if($sortedx[0][$v] == $ix)
                                        {
                                                $ydist = $sortedx[1][$v]-
$sortedx[1][$ythis] ;
                                                $yclose = $v;


                                        }
                                }

                        }

                        $ysort[0][$ys] = $sortedx[0][$ythis];
                        $ysort[1][$ys] = $sortedx[1][$ythis];
                        $ysort[2][$ys] = $sortedx[2][$ythis];
                        $ysort[3][$ys] = $sortedx[3][$ythis];
                        #print
"$ysort[0][$ys],$ysort[1][$ys],$ysort[2][$ys],$ysort[3][$ys]\n";
                        $ystep = $sortedx[1][$yclose] - $sortedx[1][$ythis];
```

```
                        $ythis = $yclose;
                        $yclose = 0;
                        $ys++;
                        if($sortedx[1][$ythis] == $sortedx[1][$ly])
                        {
                                #print "last y --> $sortedx[1][$ly]\n";
                                $ysort[0][$ys] = $sortedx[0][$ly];
                                $ysort[1][$ys] = $sortedx[1][$ly];
                                $ysort[2][$ys] = $sortedx[2][$ly];
                                $ysort[3][$ys] = $sortedx[3][$ly];
                                $ys++;



                        }
                }
                $xdist = 100000000;
                for($t=0;$t<$sdx;$t++)
                {
                        if(($sortedx[0][$t] - $sortedx[0][$xthis] < $xdist) &&
($sortedx[0][$t] > $sortedx[0][$xthis]))
                        {
                                if($sortedx[0][$t] - $sortedx[0][$xthis] > 0)
                                {
                                        $xdist = $sortedx[0][$t] -
$sortedx[0][$xthis];
                                        $xclose = $t;

                                }
                        }

                }
                $xstep = $sortedx[0][$xclose] - $sortedx[0][$xthis];
                $xthis = $xclose;
                $xclose = 0;
                if($ix == $sortedx[0][$lx])
                {
                        $xstep = 1000000;
                        #print "Last --> $sortedx[0][$xthis], $sortedx[0][$lx]\n";
                }
        }
        if($fchoice == 1)
        {
```

```
                print "\nFiltering Y -->Gaussian\n";
                $xthis = $ysort[0][0];
                for($t=0;$t<$ys;$t++)
                {
                        if(($ysort[0][$t-3] == $xthis) && ($ysort[0][$t+3] ==
$xthis))
                        {
                                $sortedy[0][$sdy] = $ysort[0][$t];
                                $sortedy[1][$sdy] = $ysort[1][$t];
                                $sortedy[2][$sdy] = $ysort[2][$t];
                                $sortedy[3][$sdy] = ($ysort[3][$t-
3]*.006)+($ysort[3][$t-2]*.061)+($ysort[3][$t-1]*.242)+($ysort[3][$t]*.383)

        +($ysort[3][$t+3]*.006)+($ysort[3][$t+2]*.061)+($ysort[3][$t+1]*.242);
                                #print
"$sortedy[0][$sdy],$sortedy[1][$sdy],$sortedy[3][$sdy]\n";
                                $sdy++;
                        }
                        else
                        {
                                if($ysort[0][$t+1] != $xthis)
                                {
                                        #print "Updating $ysort[0][$t] -->
$ysort[0][$t+1], $ysort[1][$t] --> $ysort[1][$t+1]\n";
                                        $xthis = $ysort[0][$t+1];

                                }
                        }
                }
        }
        if($fchoice == 2)
        {
                print "\nFiltering Y -->Moving Average\n";
                $xthis = $ysort[0][0];
                for($t=0;$t<$ys;$t++)
                {
                        if(($ysort[0][$t-2] == $xthis) && ($ysort[0][$t+2] ==
$xthis))
                        {
                                $sortedy[0][$sdy] = $ysort[0][$t];
                                $sortedy[1][$sdy] = $ysort[1][$t];
                                $sortedy[2][$sdy] = $ysort[2][$t];
```

```
                                        $sortedy[3][$sdy] = ($ysort[3][$t-
2]*.125)+($ysort[3][$t-1]*.125)+($ysort[3][$t]*.5)

        +($ysort[3][$t+1]*.125)+($ysort[3][$t+2]*.125);
                                        #print
"$sortedy[0][$sdy],$sortedy[1][$sdy],$sortedy[3][$sdy]\n";
                                        $sdy++;
                                }
                                else
                                {
                                        if($ysort[0][$t+1] != $xthis)
                                        {
                                                #print "Updating $ysort[0][$t] -->
$ysort[0][$t+1], $ysort[1][$t] --> $ysort[1][$t+1]\n";
                                                $xthis = $ysort[0][$t+1];

                                        }
                                }
                        }
                }

                @residual = 0;
                for($r=0;$r<$sdy;$r++)
                {
                        $residual[0][$r] = $sortedy[0][$r];
                        $residual[1][$r] = $sortedy[1][$r];
                        $residual[2][$r] = $sortedy[2][$r];
                        $residual[3][$r] = $sortedy[3][$r];
                        $Rtotal += $residual[3][$r];
                        if($r != 0)
                        {
                                $RDifftotal += $residual[3][$r] - $residual[3][$r-1];
                        }

                }
                $Rmean = $Rtotal/$r;
                for($st=0;$st<$r;$st++)
                {
                        $Rtemp += ($residual[3][$st] - $Rmean)*($residual[3][$st] -
$Rmean)
                }
                $Rvar = $Rtemp/($st-1);
                $Rstd = sqrt($Rvar);
```

```
                #print " $Rstd, $Rvar, $st, $r\n";
                return();
        }
}
sub CalcFirstDervX()
{
        $dwx=0;
        $FDcount=0;
        $FDtotal =0;
        print "\nCalcFirstDervX\n";
        $ystart = 100000000;
        $yend = 0;
        for($o=0;$o<$r;$o++)
        {
                if($residual[1][$o] < $ystart)
                {
                        $ystart = $residual[1][$o];
                        $fy = $o;
                }
                if($residual[1][$o] > $yend)
                {
                        $yend = $residual[1][$o];
                        $ly = $o;
                }
        }
        $ythis = $fy;
        $yclose = 0,$ydist=0;
        for($iy=$ystart;$iy<=$yend;$iy+=$ystep)
        {
                #print "$iy\n";
                $xstart = 1000000000;
                $xend = 0;
                for($o=0;$o<$r;$o++)
                {
                        if(($residual[0][$o] < $xstart)&& ($residual[1][$o] == $iy))
                        {
                                $xstart = $residual[0][$o];
                                $fx = $o;
                        }
                        if(($residual[0][$o] > $xend)&& ($residual[1][$o] == $iy))
                        {
                                $xend = $residual[0][$o];
                                $lx = $o;
```

```
                }
        }
        $xthis=$fx;
        $xclose=0;$xdist=0,;
        for($ix=$xstart;$ix<$xend;$ix+=$xstep)
        {
                $xdist=1000000.0;
                for($v=0;$v<$r;$v++)
                {
                        if(($residual[0][$v]-$residual[0][$xthis] < $xdist) &&
($residual[0][$v] > $residual[0][$xthis]))
                        {
                                if(($residual[1][$v] == $iy) && ($residual[0][$v] -
$residual[0][$xthis] > 0))
                                {
                                        $xdist = $residual[0][$v]-
$residual[0][$xthis] ;
                                        $xclose = $v;
                                }
                        }
                }
                $fDervx[0][$fdx] = $residual[0][$xthis];
                $fDervx[1][$fdx] = $iy;
                $fDervx[2][$fdx] = 1;
                #print "($residual[3][$xclose] -
$residual[3][$xthis])/($residual[0][$xclose]-$residual[0][$xthis])";
                $fDervx[3][$fdx] = $residual[3][$xthis];
                $FDtotal += $fDervx[3][$fdx];
                $FDcount++;
                $xstep = $residual[0][$xclose] - $residual[0][$xthis];
                $DurbWatx[0][$dwx] = $residual[0][$xthis];
                $DurbWatx[1][$dwx] = $iy;
                $DurbWatx[2][$dwx] = 1;
                $DurbWatx[3][$dwx] = $residual[3][$xthis];
                #print "$DurbWatx[0][$dwx], $DurbWatx[1][$dwx],
$DurbWatx[3][$dwx]\n";
                #print"$fDervx[0][$fdx], $fDervx[1][$fdx], $fDervx[2][$fdx],
$fDervx[3][$fdx]\n";
                $xthis = $xclose;
                $xclose = 0;
                $fdx++;
                $dwx++;
        }
```

```
                $ydist = 100000000;
                for($t=0;$t<$r;$t++)
                {
                        if(($residual[1][$t] - $residual[1][$ythis] < $ydist) &&
($residual[1][$t] > $residual[1][$ythis]))
                        {
                                if($residual[1][$t] - $residual[1][$ythis] > 0)
                                {
                                        $ydist = $residual[1][$t] - $residual[1][$ythis];
                                        $yclose = $t;
                                        #print "$ydist, $ythis, $yclose $residual[1][$ythis] ,
$residual[1][$yclose]    $iy\n";
                                }
                        }

                }
                $ystep = $residual[1][$yclose] - $residual[1][$ythis];
                print ".";
                $ythis = $yclose;
                $yclose = 0;
                if($iy == $residual[1][$ly])
                {
                        $ystep = 1000000;
                        #print "Last --> $residual[1][$ythis], $residual[1][$ly]\n";
                }



        }
        return();
}
sub CalcFirstDervY()
{
        print "\nCalcFirstDervY\n";

        $xstart = 100000000;
        $xend = 0;
        for($o=0;$o<$r;$o++)
        {
                if($residual[0][$o] < $xstart)
                {
                        $xstart = $residual[0][$o];
                        $fx = $o;
```

```
		}
		if($residual[0][$o] > $xend)
		{
			$xend = $residual[0][$o];
			$lx = $o;
		}
	}
	$xthis = $fx;
	$xclose = 0,$xdist=0;
	for($ix=$xstart;$ix<=$xend;$ix+=$xstep)
	{
		$ystart = 1000000000;
		$yend = 0;

		for($o=0;$o<$r;$o++)
		{
			if(($residual[1][$o] < $ystart)&& ($residual[0][$o] == $ix))
			{
				$ystart = $residual[1][$o];
				$fy = $o;

			}
			if(($residual[1][$o] > $yend)&& ($residual[0][$o] == $ix))
			{
				$yend = $residual[1][$o];
				$ly = $o;


			}
		}
		$ythis=$fy;
		$yclose=0;$ydist=0,;
		for($iy=$ystart;$iy<$yend;$iy+=$ystep)
		{

			$ydist=1000000.0;
			for($v=0;$v<$r;$v++)
			{
				if(($residual[1][$v]-$residual[1][$ythis] < $ydist) &&
($residual[1][$v] > $residual[1][$ythis]))
				{
					if(($residual[0][$v] == $ix) && ($residual[1][$v] -
$residual[1][$ythis] > 0))
```

```
                                        {
                                                $ydist = $residual[1][$v]-
$residual[1][$ythis] ;
                                                $yclose = $v;

                                        }
                                }
                        }

                        $fDervy[0][$fdy] = $ix;
                        $fDervy[1][$fdy] = $residual[1][$ythis];
                        $fDervy[2][$fdy] = 1;
                        $fDervy[3][$fdy] = $residual[3][$ythis];
                        $FDtotal += $fDervy[3][$fdy];
                        $FDcount++;
                        $ystep = $residual[1][$yclose] - $residual[1][$ythis];
                        $DurbWaty[0][$dwy] = $ix;
                        $DurbWaty[1][$dwy] = $residual[1][$ythis];
                        $DurbWaty[2][$dwy] = 1;
                        $DurbWaty[3][$dwy] = $residual[3][$ythis];

                        #print"$fDervy[0][$fdy], $fDervy[1][$fdy], $fDervy[2][$fdy],
$fDervy[3][$fdy]\n";
                        #print "$ystep,$fdy,$yclose,$ythis,$ydist      $residual[1][$yclose]
- $residual[1][$ythis]\n";
                        $ythis = $yclose;
                        $yclose = 0;
                        $fdy++;
                        $dwy++;
                }
                $xdist = 100000000;
                for($t=0;$t<$r;$t++)
                {
                        if(($residual[0][$t] - $residual[0][$xthis] < $xdist) &&
($residual[0][$t] > $residual[0][$xthis]))
                        {
                                if($residual[0][$t] - $residual[0][$xthis] > 0)
                                {
                                        $xdist = $residual[0][$t] - $residual[0][$xthis];
                                        $xclose = $t;
                                        #print "$xdist, $xthis, $xclose $residual[0][$xthis] ,
$residual[0][$xclose]    $ix\n";
                                }
```

```
                        }
                }
                $xstep = $residual[0][$xclose] - $residual[0][$xthis];
                $xthis = $xclose;
                print ".";
                $xclose = 0;
                if($ix == $residual[0][$lx])
                {
                        $xstep = 1000000;
                        #print "Last --> $residual[0][$xthis], $residual[0][$lx]\n";
                }
        }
        return();
}
sub AnaFirstDerv()
{
        print "\nAnalyzing First Derv\n";

        $FDmean = $FDtotal/$FDcount;
        $FDtemp=0;
        for($fdst=0;$fdst<$fdx;$fdst++)
        {
                $FDtemp += ($fDervx[3][$fdst] - $FDmean)*($fDervx[3][$fdst] -
$FDmean);
        }
        for($fdst=0;$fdst<$fdy;$fdst++)
        {
                $FDtemp += ($fDervy[3][$fdst] - $FDmean)*($fDervy[3][$fdst] -
$FDmean);
        }
        $FDvar = $FDtemp/($FDcount-1);
        $FDstd = sqrt($FDvar);
        $shit = $fdx+$fdy;
        print "  Use a Max/Min Threshold -->1 for Yes<--?\n          ";
        $MMchoice = 0;
        chomp($MMchoice);
        if($MMchoice != 1)
        {
                $crit = .00000000000001;
        }
        else
        {
```

```
		print "\nresidual mean = $Rmean, s= $Rstd\nFD mean = $FDmean, s = $FDstd\n	What MAX/MIN Threshold?\n	";
		$crit = <STDIN>;
		chomp($crit);
	}
	$amm=0;
	$ythis = $fDervx[1][0];
	$absMax = -1000000000;
	$absMin = 10000000000;
	for($id=0;$id<$fdx;$id++)
	{
		if(($fDervx[1][$id-3] == $ythis) && ($fDervx[1][$id+3] == $ythis))
		{
			$behind = abs($fDervx[3][$id] - $fDervx[3][$id-1]);
			$ahead = abs($fDervx[3][$id] - $fDervx[3][$id+1]);
			#print "$behind, $ahead\n";
			#print"$fDervx[0][$id], $fDervx[1][$id], $fDervx[2][$id], $fDervx[3][$id]\n";
			if(($fDervx[3][$id] > $fDervx[3][$id-1]) && ($fDervx[3][$id] > $fDervx[3][$id+1]) &&
				($fDervx[3][$id-1] > $fDervx[3][$id-2]) && ($fDervx[3][$id+1] > $fDervx[3][$id+2]) &&
				($fDervx[3][$id-2] > $fDervx[3][$id-3]) && ($fDervx[3][$id+2] > $fDervx[3][$id+3]))
			{
				if(($behind > $crit) || ($ahead > $crit))
				{
					$fmaxMinx[0][$fmmx] = $fDervx[0][$id];
					$fmaxMinx[1][$fmmx] = $fDervx[1][$id];
					$fmaxMinx[2][$fmmx] = 1;
					$fmaxMinx[3][$fmmx] = "xMax";
					#print "xMAX found at $fmaxMinx[0][$fmmx], $fmaxMinx[1][$fmmx], $fmaxMinx[3][$fmmx]\n";
					$fmmx++;
					$e++;
					if($fDervx[3][$id] > $absMax)
					{
						$absMax = $fDervx[3][$id];
						$tempMax[0] = $fDervx[0][$id];
						$tempMax[1] = $fDervx[1][$id];
						$tempMax[2] = $fDervx[2][$id];
						$tempMax[3] = $fDervx[3][$id];
						$tempMax[4] = "XABSMAX";
```

```
                                        #print "XAbsMax = $fDervx[3][$id] at
$fDervx[0][$id], $fDervx[1][$id]\n";
                                    }
                                }
                            }
                            else
                            {
                                $behind = $fDervx[3][$id-1] - $fDervx[3][$id];
                                $ahead = $fDervx[3][$id+1] - $fDervx[3][$id];
                                if(($fDervx[3][$id] < $fDervx[3][$id-1]) &&
($fDervx[3][$id] < $fDervx[3][$id+1])&&
                                    ($fDervx[3][$id-1] < $fDervx[3][$id-2]) &&
($fDervx[3][$id+1] < $fDervx[3][$id+2])&&
                                    ($fDervx[3][$id-2] < $fDervx[3][$id-3]) &&
($fDervx[3][$id+2] < $fDervx[3][$id+3]))
                                {
                                    if(($behind > $crit) || ($ahead > $crit))
                                    {
                                        $fmaxMinx[0][$fmmx] = $fDervx[0][$id];
                                        $fmaxMinx[1][$fmmx] = $fDervx[1][$id];
                                        $fmaxMinx[2][$fmmx] = 1;
                                        $fmaxMinx[3][$fmmx] = "xMin";
                                        #print "xMIN found at
$fmaxMinx[0][$fmmx], $fmaxMinx[1][$fmmx], $fmaxMinx[3][$fmmx]\n";
                                        $fmmx++;
                                        $e++;
                                        if($fDervx[3][$id] < $absMin)
                                        {
                                            $absMin = $fDervx[3][$id];
                                            $tempMin[0] = $fDervx[0][$id];
                                            $tempMin[1] = $fDervx[1][$id];
                                            $tempMin[2] = $fDervx[2][$id];
                                            $tempMin[3] = $fDervx[3][$id];
                                            $tempMin[4] = "XABSMIN";
                                            #print "XAbsMax = $fDervx[3][$id]
at $fDervx[0][$id], $fDervx[1][$id]\n";
                                        }
                                    }
                                }
                            }

                        }
                        else
```

```
		{
			#print "UPDATE-- $fDervx[0][$id], $fDervx[1][$id],
$fDervx[2][$id], $fDervx[3][$id]\n";
			$ythis = $fDervx[1][$id];
			if($absMax > -100000)
			{
				$AMM[0][$amm] = $tempMax[0];
				$AMM[1][$amm] = $tempMax[1];
				$AMM[2][$amm] = $tempMax[2];
				$AMM[3][$amm] = $tempMax[3];
				$AMM[4][$amm] = $tempMax[4];
				$amm++;
			}
			if($absMin < 10000000)
			{
				$AMM[0][$amm] = $tempMin[0];
				$AMM[1][$amm] = $tempMin[1];
				$AMM[2][$amm] = $tempMin[2];
				$AMM[3][$amm] = $tempMin[3];
				$AMM[4][$amm] = $tempMin[4];
				$amm++;
			}

			$absMax = -10000000000;
			$absMin = 10000000000;

		}
	}
	$xthis = $fDervy[0][0];
	$absMax = -1000000000;
	$absMin = 10000000000;
	for($id=0;$id<$fdy;$id++)
	{
		if(($fDervy[0][$id-2] == $xthis) && ($fDervy[0][$id+2] == $xthis))
		{
			$behind = abs($fDervy[3][$id] - $fDervy[3][$id-1]);
			$ahead = abs($fDervy[3][$id] - $fDervy[3][$id+1]);
			#print"$fDervy[0][$id], $fDervy[1][$id], $fDervy[2][$id],
$fDervy[3][$id]\n";
			if(($fDervy[3][$id] > $fDervy[3][$id-1]) && ($fDervy[3][$id] >
$fDervy[3][$id+1]) &&
				($fDervy[3][$id-1] > $fDervy[3][$id-2]) &&
($fDervy[3][$id+1] > $fDervy[3][$id+2])&&
```

```
                        ($fDervy[3][$id-2] > $fDervy[3][$id-3]) &&
($fDervy[3][$id+2] > $fDervy[3][$id+3]))
                {
                        if(($behind > $crit) || ($ahead > $crit))
                        {
                                $fmaxMiny[0][$fmmy] = $fDervy[0][$id];
                                $fmaxMiny[1][$fmmy] = $fDervy[1][$id];
                                $fmaxMiny[2][$fmmy] = 1;
                                $fmaxMiny[3][$fmmy] = "yMax";
                                #print "yMAX found at $fmaxMiny[0][$fmmy],
$fmaxMiny[1][$fmmy],$fmaxMiny[3][$fmmy]\n";
                                $fmmy++;
                                $e++;
                                if($fDervy[3][$id] > $absMax)
                                {
                                        $absMax = $fDervy[3][$id];
                                        $tempMax[0] = $fDervy[0][$id];
                                        $tempMax[1] = $fDervy[1][$id];
                                        $tempMax[2] = $fDervy[2][$id];
                                        $tempMax[3] = $fDervy[3][$id];
                                        $tempMax[4] = "YABSMAX";
                                        #print "YAbsMax = $fDervy[3][$id] at
$fDervy[0][$id], $fDervy[1][$id]\n";
                                }
                        }
                }
                else
                {
                        $behind = $fDervy[3][$id-1] - $fDervy[3][$id];
                        $ahead = $fDervy[3][$id+1] - $fDervy[3][$id];
                        if(($fDervy[3][$id] < $fDervy[3][$id-1]) &&
($fDervy[3][$id] < $fDervy[3][$id+1])&&
                                ($fDervy[3][$id-1] < $fDervy[3][$id-2]) &&
($fDervy[3][$id+1] < $fDervy[3][$id+2])&&
                                ($fDervy[3][$id-2] < $fDervy[3][$id-3]) &&
($fDervy[3][$id+2] < $fDervy[3][$id+3]))
                        {
                                if(($behind > $crit) || ($ahead > $crit))
                                {
                                        $fmaxMiny[0][$fmmy] = $fDervy[0][$id];
                                        $fmaxMiny[1][$fmmy] = $fDervy[1][$id];
                                        $fmaxMiny[2][$fmmy] = 1;
                                        $fmaxMiny[3][$fmmy] = "yMin";
```

```
                                        #print "yMIN found at
$fmaxMiny[0][$fmmy], $fmaxMiny[1][$fmmy],$fmaxMiny[3][$fmmy]\n";
                                        $fmmy++;
                                        $e++;
                                        if($fDervy[3][$id] < $absMin)
                                        {
                                                $absMin = $fDervy[3][$id];
                                                $tempMin[0] = $fDervy[0][$id];
                                                $tempMin[1] = $fDervy[1][$id];
                                                $tempMin[2] = $fDervy[2][$id];
                                                $tempMin[3] = $fDervy[3][$id];
                                                $tempMin[4] = "YABSMIN";
                                                #print "YAbsMin = $fDervy[3][$id]
at $fDervy[0][$id], $fDervy[1][$id]\n";
                                        }
                                }
                        }
                }

        }
        else
        {
                #print "UPDATE-- $fDervy[0][$id], $fDervy[1][$id],
$fDervy[2][$id], $fDervy[3][$id]\n";
                $xthis = $fDervy[0][$id];
                if($absMax > -100000)
                {
                        $AMM[0][$amm] = $tempMax[0];
                        $AMM[1][$amm] = $tempMax[1];
                        $AMM[2][$amm] = $tempMax[2];
                        $AMM[3][$amm] = $tempMax[3];
                        $AMM[4][$amm] = $tempMax[4];
                        $amm++;
                }
                if($absMin < 10000000)
                {
                        $AMM[0][$amm] = $tempMin[0];
                        $AMM[1][$amm] = $tempMin[1];
                        $AMM[2][$amm] = $tempMin[2];
                        $AMM[3][$amm] = $tempMin[3];
                        $AMM[4][$amm] = $tempMin[4];
                        $amm++;
                }
```

```
                        $absMax = -10000000000;
                        $absMin = 10000000000;

                }
        }
        $l = $fmmy+$fmmx;
        print "\n\nFound $l MaxMin\n";
        return();
}
sub CalcResidDiff()
{
        print "  Use a Differnece in Residual Threshold --> 1 for Yes<--\n   ";
        $DRchoice = 1;
        $RDiffMean = $RDifftotal/($r-1);
        chomp($DRchoice);
        if($DRchoice != 1)
        {
                $DRcrit = .000000000001;
        }
        else
        {
                print "\nAverage Resid Diff = $RDiffMean, Resid s= $Rstd\nFD Mean =
$FDmean, s = $FDstd\n         What Diff Threshold?\n         ";
                $DRcrit = 1.5;
                chomp($DRcrit);
        }

        $ystart = 100000000;
        $yend = 0;
        #print "\nSorting X -->\n";
        for($o=0;$o<$r;$o++)
        {

                if($residual[1][$o] < $ystart)
                {
                        $ystart = $residual[1][$o];
                        $fy = $o;
                }
                if($residual[1][$o] > $yend)
                {
                        $yend = $residual[1][$o];
                        $ly = $o;
                }
```

```
	}

	$ythis = $fy;
	$yclose = 0,$ydist=0;
	for($iy=$ystart;$iy<=$yend;$iy+=$ystep)
	{
		#print "$iy\n";
		$xstart = 1000000000;
		$xend = 0;
		for($o=0;$o<$r;$o++)
		{
			if(($residual[0][$o] < $xstart)&& ($residual[1][$o] == $iy))
			{
				$xstart = $residual[0][$o];
				$fx = $o;
			}
			if(($residual[0][$o] > $xend)&& ($residual[1][$o] == $iy) )
			{
				$xend = $residual[0][$o];
				$lx = $o;
			}
		}
		$xthis=$fx;
		$xclose=0;$xdist=0,;
		for($ix=$xstart;$ix<$xend;$ix+=$xstep)
		{
			#print "$ix\n";
			$xdist=1000000.0;
			for($v=0;$v<$r;$v++)
			{

				if(($residual[0][$v]-$residual[0][$xthis] < $xdist) &&
($residual[0][$v] > $residual[0][$xthis]))
				{
					if($residual[1][$v] == $iy)
					{
						$xdist = $residual[0][$v]-
$residual[0][$xthis] ;
						$xclose = $v;


					}
				}
```

```
                        }

                        $xsort[0][$xs] = $residual[0][$xthis];
                        $xsort[1][$xs] = $residual[1][$xthis];
                        $xsort[2][$xs] = $residual[2][$xthis];
                        $xsort[3][$xs] = $residual[3][$xthis];
                        #print
"$xsort[0][$xs],$xsort[1][$xs],$xsort[2][$xs],$xsort[3][$xs]\n";
                        $xstep = $residual[0][$xclose] - $residual[0][$xthis];
                        $xthis = $xclose;
                        $xclose = 0;
                        $xs++;
                        if($residual[0][$xthis] == $residual[0][$lx])
                        {
                                #print "last x --> $residual[0][$lx]\n";
                                $xsort[0][$xs] = $residual[0][$lx];
                                $xsort[1][$xs] = $residual[1][$lx];
                                $xsort[2][$xs] = $residual[2][$lx];
                                $xsort[3][$xs] = $residual[3][$lx];
                                $xs++;
                                $xstep = 10000000;


                        }
                }
                $ydist = 100000000;
                for($t=0;$t<$r;$t++)
                {
                        if(($residual[1][$t] - $residual[1][$ythis] < $ydist) &&
($residual[1][$t] > $residual[1][$ythis]))
                        {
                                if($residual[1][$t] - $residual[1][$ythis] > 0)
                                {
                                        $ydist = $residual[1][$t] - $residual[1][$ythis];
                                        $yclose = $t;
                                        #print "$ydist, $ythis, $yclose $residual[1][$ythis] ,
$residual[1][$yclose]    $iy\n";
                                }
                        }

                }
                $ystep = $residual[1][$yclose] - $residual[1][$ythis];
```

```
                $ythis = $yclose;
                $yclose = 0;
                if($iy == $residual[1][$ly])
                {
                        $ystep = 1000000;
                        #print "Last --> $residual[1][$ythis], $residual[1][$ly]\n";
                }
        }

        #print "\nSearching X Resid Diff's -->\n";
        $dr=0;
        $ythis = $xsort[1][0];
        for($t=0;$t<$xs;$t++)
        {
                if(($xsort[1][$t-1] == $ythis) && ($xsort[1][$t+1] == $ythis))
                {

                        if(abs($xsort[3][$t] - $xsort[3][$t-1]) > $DRcrit)
                        {
                                $DiffResid[0][$dr] = $xsort[0][$t];
                                $DiffResid[1][$dr] = $xsort[1][$t];
                                $DiffResid[2][$dr] = $xsort[2][$t];
                                $DiffResid[3][$dr] = abs($xsort[3][$t]-$xsort[3][$t-1]);
                                $DiffResid[4][$dr] = "Xdiff";
                                #print "$DiffResid[0][$dr], $DiffResid[1][$dr],
$DiffResid[2][$dr], $DiffResid[3][$dr]\n";
                                $dr++;
                        }

                }
                else
                {
                        if($xsort[1][$t+1] != $ythis)
                        {
                                #print "Updating $xsort[0][$t] --> $xsort[0][$t+1],
$xsort[1][$t] --> $xsort[1][$t+1]\n";
                                $ythis = $xsort[1][$t+1];

                        }
                }
        }
```

```
#print "\nSorting Y --> \n";
$xstart = 100000000;
$xend = 0;
for($o=0;$o<$xs;$o++)
{

	if($xsort[0][$o] < $xstart)
	{
		$xstart = $xsort[0][$o];
		$fx = $o;
	}
	if($xsort[0][$o] > $xend)
	{
		$xend = $xsort[0][$o];
		$lx = $o;
	}
}
#print "\n$xstart to $xend\n";
$xthis = $fx;
$xclose = 0,$xdist=0;
for($ix=$xstart;$ix<=$xend;$ix+=$xstep)
{
	#print "$iy\n";
	$ystart = 1000000000;
	$yend = 0;
	for($o=0;$o<$xs;$o++)
	{
		if(($xsort[1][$o] < $ystart)&& ($xsort[0][$o] == $ix))
		{
			$ystart = $xsort[1][$o];
			$fy = $o;
		}
		if(($xsort[1][$o] > $yend)&& ($xsort[0][$o] == $ix) )
		{
			$yend = $xsort[1][$o];
			$ly = $o;
		}
	}
	$ythis=$fy;
	$yclose=0;$ydist=0,;
	for($iy=$ystart;$iy<$yend;$iy+=$ystep)
	{
		#print "$ix\n";
```

```
                        $ydist=1000000.0;
                        for($v=0;$v<$xs;$v++)
                        {

                                if(($xsort[1][$v]-$xsort[1][$xthis] < $ydist) &&
($xsort[1][$v] > $xsort[1][$ythis]))
                                {
                                        if($xsort[0][$v] == $ix)
                                        {
                                                $ydist = $xsort[1][$v]-$xsort[1][$ythis] ;
                                                $yclose = $v;


                                        }
                                }

                        }

                        $ysort[0][$ys] = $xsort[0][$ythis];
                        $ysort[1][$ys] = $xsort[1][$ythis];
                        $ysort[2][$ys] = $xsort[2][$ythis];
                        $ysort[3][$ys] = $xsort[3][$ythis];
                        #print
"$ysort[0][$ys],$ysort[1][$ys],$ysort[2][$ys],$ysort[3][$ys]\n";
                        $ystep = $xsort[1][$yclose] - $xsort[1][$ythis];
                        $ythis = $yclose;
                        $yclose = 0;
                        $ys++;
                        if($xsort[1][$ythis] == $xsort[1][$ly])
                        {
                                #print "last y --> $sortedx[1][$ly]\n";
                                $ysort[0][$ys] = $xsort[0][$ly];
                                $ysort[1][$ys] = $xsort[1][$ly];
                                $ysort[2][$ys] = $xsort[2][$ly];
                                $ysort[3][$ys] = $xsort[3][$ly];
                                $ys++;



                        }
                }
                $xdist = 100000000;
                for($t=0;$t<$xs;$t++)
```

```
		{
			if(($xsort[0][$t] - $xsort[0][$xthis] < $xdist) && ($xsort[0][$t] > $xsort[0][$xthis]))
			{
				if($xsort[0][$t] - $xsort[0][$xthis] > 0)
				{
					$xdist = $xsort[0][$t] - $xsort[0][$xthis];
					$xclose = $t;

				}
			}

		}
		$xstep = $xsort[0][$xclose] - $xsort[0][$xthis];
		$xthis = $xclose;
		$xclose = 0;
		if($ix == $xsort[0][$lx])
		{
			$xstep = 1000000;
			#print "Last --> $xsort[0][$xthis], $xsort[0][$lx]\n";
		}
	}
	#print "\nSearching  Y Residual Diffs-->\n";
	$xthis = $ysort[0][0];
	for($t=0;$t<$ys;$t++)
	{
		if(($ysort[0][$t-1] == $xthis) && ($ysort[0][$t+1] == $xthis))
		{

			if(abs($ysort[3][$t] - $ysort[3][$t-1]) > $DRcrit)
			{
				$DiffResid[0][$dr] = $ysort[0][$t];
				$DiffResid[1][$dr] = $ysort[1][$t];
				$DiffResid[2][$dr] = $ysort[2][$t];
				$DiffResid[3][$dr] = abs($ysort[3][$t]-$ysort[3][$t-1]);
				$DiffResid[4][$dr] = "Ydiff";
				#print "$DiffResid[0][$dr], $DiffResid[1][$dr], $DiffResid[2][$dr], $DiffResid[3][$dr]\n";
				$dr++;
			}
		}
		else
		{
```

```
                        if($ysort[0][$t+1] != $xthis)
                        {
                                #print "Updating $ysort[0][$t] --> $ysort[0][$t+1],
$ysort[1][$t] --> $ysort[1][$t+1]\n";
                                $xthis = $ysort[0][$t+1];

                        }
                }
        }
        print "\n\nFound $dr Residual Diffs\n";
        return();
}
sub DurbinWatson()
{
        $dwtx=0;
        print "\nDurbinWatson - X\n";
        $ythis = $DurbWatx[1][0];
        for($dw=0;$dw<$dwx;$dw++)
        {
                #print "
$DurbWatx[0][$dw],$DurbWatx[1][$dw],$DurbWatx[2][$dw],$DurbWatx[3][$dw]\n";
                if($DurbWatx[1][$dw-1] != $ythis)
                {
                        #print "
$DurbWatx[0][$dw],$DurbWatx[1][$dw],$DurbWatx[2][$dw],$DurbWatx[3][$dw] --
Starting\n";
                        #$totSum += ($DurbWatx[3][$dw]*$DurbWatx[3][$dw]);
                        $xstart = $DurbWatx[0][$dw];
                        #print " Starting --> $totSum, $eSum\n";
                }
                if($DurbWatx[1][$dw+1] != $ythis)
                {
                        $totSum += ($DurbWatx[3][$dw]*$DurbWatx[3][$dw]);
                        $eSum += (($DurbWatx[3][$dw]-$DurbWatx[3][$dw-
1])*($DurbWatx[3][$dw]-$DurbWatx[3][$dw-1]));
                        $DWX[0][$dwtx] = "X";
                        $DWX[1][$dwtx] = $DurbWatx[1][$dw];
                        $DWX[2][$dwtx] = 1;
                        $DWX[3][$dwtx] = $eSum/$totSum;
                        #print "
$DurbWatx[0][$dw],$DurbWatx[1][$dw],$DurbWatx[2][$dw],$DurbWatx[3][$dw] --
Ending\n";
```

```
                        #print "Writing --> $totSum, $eSum --> $DWX[0][$dwtx],
$DWX[1][$dwtx], $DWX[2][$dwtx], $DWX[3][$dwtx]\n";
                        $dwtx++;
                        $eSum=0;
                        $totSum=0;
                        $ythis = $DurbWatx[1][$dw+1];
                }
                else
                {
                        if(($DurbWatx[1][$dw-1] == $ythis) && ($DurbWatx[1][$dw+1]
== $ythis))
                        {
                                #print "
$DurbWatx[0][$dw],$DurbWatx[1][$dw],$DurbWatx[2][$dw],$DurbWatx[3][$dw] --
Adding\n";
                                $totSum += ($DurbWatx[3][$dw]*$DurbWatx[3][$dw]);
                                $eSum += (($DurbWatx[3][$dw]-$DurbWatx[3][$dw-
1])*($DurbWatx[3][$dw]-$DurbWatx[3][$dw-1]));
                                #print "Adding --> $totSum, $eSum\n";
                        }
                }

        }
        $dwty=0;
        print "\nDurbinWatson - Y\n";
        $xthis = $DurbWaty[0][0];
        for($dw=0;$dw<$dwy;$dw++)
        {
                if($DurbWaty[0][$dw-1] != $xthis)
                {
                        #print "
$DurbWaty[0][$dw],$DurbWaty[1][$dw],$DurbWaty[2][$dw],$DurbWaty[3][$dw] --
Starting\n";
                        #$totSum += ($DurbWaty[3][$dw]*$DurbWaty[3][$dw]);
                        $ystart = $DurbWaty[1][$dw];
                        #print " Starting --> $totSum, $eSum\n";
                }
                if($DurbWaty[0][$dw+1] != $xthis)
                {
                        $totSum += ($DurbWaty[3][$dw]*$DurbWaty[3][$dw]);
                        $eSum += (($DurbWaty[3][$dw]-$DurbWaty[3][$dw-
1])*($DurbWaty[3][$dw]-$DurbWaty[3][$dw-1]));
                        $DWY[0][$dwty] = "Y";
```

```
                        $DWY[1][$dwty] = $DurbWaty[0][$dw];
                        $DWY[2][$dwty] = 1;
                        $DWY[3][$dwty] = $eSum/$totSum;
                        #print "
$DurbWaty[0][$dw],$DurbWaty[1][$dw],$DurbWaty[2][$dw],$DurbWaty[3][$dw] --
Ending\n";
                        #print "Writing --> $totSum, $eSum --> $DWY[0][$dwty],
$DWY[1][$dwty], $DWY[2][$dwty], $DWY[3][$dwty]\n";
                        $dwty++;
                        $eSum=0;
                        $totSum=0;
                        $xthis = $DurbWaty[0][$dw+1];
                }
                else
                {
                        if(($DurbWaty[0][$dw-1] == $xthis) && ($DurbWaty[0][$dw+1]
== $xthis))
                        {
                                #print "
$DurbWaty[0][$dw],$DurbWaty[1][$dw],$DurbWaty[2][$dw],$DurbWaty[3][$dw] --
Adding\n";
                                $totSum += ($DurbWaty[3][$dw]*$DurbWaty[3][$dw]);
                                $eSum += (($DurbWaty[3][$dw]-$DurbWaty[3][$dw-
1])*($DurbWaty[3][$dw]-$DurbWaty[3][$dw-1]));
                                #print "Adding --> $totSum, $eSum\n";
                        }
                }
        }
        return();
}
sub PrintFiles()
{
        print" \nPrintFiles\n";

        open FILEHANDLE4, ">$MPI-TotalResid-MaxMin.dat" || die "NO Worky\n";
        open FILEHANDLE6, ">$MPI-TotalResid-DiffResid.dat" || die "NO Worky\n";
        open FILEHANDLE7, ">$MPI-TotalResid-AbsMaxMin.dat" || die "NO
Worky\n";
        for($i=0;$i<$fmmx;$i++)
        {
                print FILEHANDLE4
"$fmaxMinx[0][$i],$fmaxMinx[1][$i],$fmaxMinx[3][$i]\n";
        }
```

```
	for($i=0;$i<$fmmy;$i++)
	{
		print FILEHANDLE4
"$fmaxMiny[0][$i],$fmaxMiny[1][$i],$fmaxMiny[3][$i]\n";
	}
	for($i=0;$i<$dr;$i++)
	{
		print FILEHANDLE6 "$DiffResid[0][$i],$DiffResid[1][$i],
$DiffResid[2][$i], $DiffResid[3][$i], $DiffResid[4][$i]\n";
	}
	for($i=0;$i<$amm;$i++)
	{
		print FILEHANDLE7 "$AMM[0][$i],$AMM[1][$i], $AMM[2][$i],
$AMM[3][$i], $AMM[4][$i]\n";
	}
	close FILEHANDLE4;
	close FILEHANDLE6;
	close FILEHANDLE7;
	return();
```

```
cd jeremy\research\model\cali-SOR\newnew
del T1*.*
bin2asc.exe
perl headsave.pl
perl setup.pl
perl filter.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
```

```
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
```

```
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
```

```
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
```

```
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl cbrwg15c.pl
perl TotResidCalc.pl
perl mmweight.pl
perl drweight.pl
```