Final Report
**Louisiana Coastal Wetlands
Modeling**

DE-FY22-96 PC 96226

Submitted to
The Department of Energy

Submitted By:
Dr. Ali Ansari, Associate Professor
Department of Engineering Technology
Virginia State University

Ms. Lilibeth de Los Santos, Graduate Student
Department of Economics
Virginia State University

#1
ACQUISITION & ASSISTANCE
2000 MAR - 1 A 9:42
USDOE-FETC

ACQUISITION & ASSISTANCE
1999 DEC - 3 A 10:25
USDOE-FETC

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, make any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Table of Contents

A.	Objective.....	1
B.	Abstract.....	1
C.	Introduction.....	2
D.	Data Source.....	10
E.	Land Loss Interpretation.....	10
F.	Data For The Polynomial Calculation.....	13
G.	Polynomial Regression.....	14
H.	Sensitivity.....	15
I.	Program Description.....	16
J.	Polynomial Models and Graphs.....	18
K.	Functions Descriptions.....	29
L.	Source Code.....	31

A. Objective

To design a computer simulator to describe the Louisiana Coastal Wetlands losses

B. Abstract

The Louisiana Coastal Plain is divided into 62 quadrangles. The land loss rates for each quadrangle are defined for 4 time periods: 1930's to 1956-58, 1956-58 to 1974, 1974 to 1983, and 1983 to 1990.

The rate of land loss for each quadrangle is described by a third order polynomial. A computer simulator is developed using C language to calculate the coefficients of the polynomials as well as graphical representation of the wetland losses vs. time.

C. Introduction

In General, any mathematical model may be describe as:

$$(1) \quad Y = F(X)$$

The objective of a model is to come up with a function F such that Y is relatively close to $\hat{Y}$ (relatively close defined as: $\sum_m (\hat{y}_i - y_i)^2 < \epsilon$). In control system theory, this is known as System Identification.

This study has treated the Louisiana costale wetlands as an unknown plant. The independent variable, X describes the number of years (input factor) under consideration during the wetlands losses measurement, and Y represents the corresponding wetlands losses (Area). A mathematical model for this plant is a set of rules that relates X to the output Y . This may be shown as:

In the ideal model $Y = \hat{Y}$. Depending upon the required accuracy and the data under consideration, the model may be represented by various systems such as: a linear system where F is a matrix, a polynomial, a neural network, etc. The model's parameters should be calculated in such a way as to minimize ϵ .

The intention of this study was not to produced the data to describe the rate of wetlands losses, but to utilize the available data to develop a mathematical model and a computer simulator to fit the available data. Sixty two models were developed under this study to describe the wetland

loss for each one of sixty two quadrangles. The maps of land loss rates for the four periods under this study are shown in the following figures (Courtesy of Joseph B. Dunbar.) Since the data describes the wetlands losses of quadrangles during four distinct time periods, a third degree polynomial model is a good candidate for each quadrangle.

[illegible]

LAND LOSS RATES
TIME 1 (1930'S-1956-58)

CHENIER PLAIN

DELTAIC PLAIN

RATES (MI²/YEAR)

0.0 - 1.0
1.0 - 2.0
2.0 - 4.0

GULF OF MEXICO

MISSISSIPPI LOUISIANA

MISSISSIPPI ALABAMA

SCALE

0 20 40 MI

Figure 2a. Summary of land loss rates in square miles per year for time period 1 (1930's to 1956-58)

CHENIER PLAIN

DELTAIC PLAIN

RATES (MI²/YEAR)

- 0.0 - 1.0
- 1.0 - 2.0
- 2.0 - 3.0
- 3.0 - 4.0

SCALE

0 20 40 MI

Figure 2b. Summary of land loss rates in square miles per year for time period 2 (1956-58 to 1974)

Figure 2c. Summary of land loss rates in square miles per year for time period 3 (1974 to 1983)

Figure 2d. Summary of land loss rates in square miles per year for time period 4 (1983 to 1990)

D. Data Source

The data utilized in this study is extracted from the Technical Report GL-90-2, Land Loss Rates, Report 3, Louisiana Coastal Plain, by J. B. Dunbar. The data is obtained by comparing four vintages of aerial photography with a base map (scale: 1:62,500). The base maps used were either US Coast and Geodetic Survey Air Photo Compilation Sheet(T-sheet) or early USGS 15-mintopographic quadrangle.

<u>Date</u>	<u>Source</u>	<u>Original Scale</u>
1932-33	US Coast and Geodetic Survey Air Photo Compilation Sheets (T-sheet)	1:20,000
1933-55	USGS Topographic Quadrangle	1:62,500
1933-35	Tobin Surveys, Black & White Aerial Photo Mosaics	1:62,500
1956-58	Tobin Surveys, Black & White Aerial Photo Mosaics	1:62,500
1974	NASA Color	1:120,000
1983	National High Altitude Program Color	1:58,000
1990	NASA	1:62,500

E. Land Loss Interpretation

The wetland loss is defined as changes from land to water over a period of approximately 60 years. The details of measuring wetlands loss can be found in the aforementioned report. The Louisiana Coastal Plain land loss rates for four time periods is summarized in the following tables:

Table 1 - Louisiana Coastal Plain Land Loss Rates

<u>Quadrangle Name</u>	<u>Time Period 1</u>	<u>Avg. Loss in Square miles/year</u>	<u>Time Period 2</u>	<u>Avg. Loss in Square miles/year</u>	<u>Time Period 3</u>	<u>Avg. Loss in Square miles/year</u>	<u>Time Period 4</u>	<u>Avg. Loss in Square miles/year</u>
Abbeville	1935-1954	0.07	1954-1974	0.23	1974-1983	0.24	1983-1990	0.43
Barataria	1939-1956	1.08	1956-1974	1.20	1974-1983	0.70	1983-1990	1.06
Bay Dogris	1932-1958	0.42	1958-1974	1.44	1974-1983	1.26	1983-1990	1.15
Bayou Du Large	1932-1958	0.18	1958-1974	1.61	1974-1983	0.65	1983-1990	0.47
Bayou Sale	1937-1956	0.31	1956-1974	0.36	1974-1983	0.19	1983-1990	0.07
Belle Isle	1940-1956	0.38	1956-1974	0.32	1974-1983	0.15	1983-1990	0.07
Black Bay	1932-1958	0.21	1958-1974	0.37	1974-1983	0.52	1983-1990	0.22
Bonnet Carre	1936-1958	0.10	1958-1974	0.44	1974-1983	0.19	1983-1990	0.07
Breton Island	1932-1958	0.26	1958-1974	0.18	1974-1983	0.11	1983-1990	0.05
Caillou Bay	1932-1958	0.22	1958-1974	0.40	1974-1983	0.43	1983-1990	0.21
Cameron	1933-1955	0.08	1955-1974	2.47	1974-1983	0.60	1983-1990	0.25
Cat Island	1932-1958	0.07	1958-1974	0.09	1974-1983	0.11	1983-1990	0.07
Chef Menteur	1932-1958	0.49	1958-1974	0.41	1974-1983	0.28	1983-1990	0.28
Chen. Au Tigre	1935-1951	0.08	1951-1974	0.36	1974-1983	0.13	1983-1990	0.32
Constance Bayou	1932-1955	0.64	1955-1974	0.82	1974-1983	0.50	1983-1990	0.59
Covington	1932-1958	0.02	1958-1974	0.18	1974-1983	0.02	1983-1990	0.18
Cut Off	1939-1958	0.22	1958-1974	0.53	1974-1983	0.39	1983-1990	0.51
Derouen	1932-1956	0.24	1956-1974	0.22	1974-1983	0.24	1983-1990	0.25
Dulac	1932-1958	0.37	1958-1974	0.98	1974-1983	1.99	1983-1990	1.49
East Delta	1932-1958	1.17	1958-1974	1.90	1974-1983	0.27	1983-1990	0.34
Empire	1932-1958	0.35	1958-1974	1.12	1974-1983	2.66	1983-1990	2.08
Forked Island	1935-1955	0.01	1955-1974	0.15	1974-1983	0.14	1983-1990	0.27
Fort Livingston	1932-1958	0.34	1958-1974	0.53	1974-1983	0.89	1983-1990	0.54
Gibson	1939-1958	0.11	1958-1974	1.50	1974-1983	0.45	1983-1990	0.54
Grand Lake East	1932-1955	0.29	1955-1974	0.41	1974-1983	1.54	1983-1990	0.62
Grand Lake West	1933-1955	0.05	1955-1974	1.12	1974-1983	1.30	1983-1990	0.44
Hahnville	1935-1958	0.11	1958-1974	0.57	1974-1983	0.43	1983-1990	0.17
Hog Bayou	1932-1955	0.54	1955-1974	0.72	1974-1983	0.15	1983-1990	0.56
Houma	1939-1958	0.13	1958-1974	0.24	1974-1983	0.17	1983-1990	0.29
Jeanerette	1937-1956	0.08	1956-1974	0.08	1974-1983	0.06	1983-1990	0.10

Table 2-Continue

<u>Quadrangle Name</u>	<u>Time Period 1</u>	<u>Avg. Loss in Square miles/year</u>	<u>Time Period 2</u>	<u>Avg. Loss in Square miles/year</u>	<u>Time Period 3</u>	<u>Avg. Loss in Square miles/year</u>	<u>Time Period 4</u>	<u>Avg. Loss in Square miles/year</u>
Johnsons Bayou	1933-1955	0.09	1955-1974	3.12	1974-1983	1.02	1983-1990	0.30
Lac des Allemands	1945-1958	0.13	1958-1974	0.11	1974-1983	0.66	1983-1990	0.19
Lake Decade	1931-1956	0.25	1956-1974	1.31	1974-1983	0.38	1983-1990	0.26
Lake Felicity	1932-1958	0.29	1958-1974	1.32	1974-1983	1.61	1983-1990	1.35
Leeville	1932-1958	0.28	1958-1974	0.40	1974-1983	0.90	1983-1990	0.73
Marsh Island	1932-1956	0.23	1956-1974	0.39	1974-1983	0.24	1983-1990	0.25
Mitchell Key	1932-1956	0.05	1956-1974	0.03	1974-1983	0.07	1983-1990	0.02
Morgan City	1931-1956	0.20	1956-1974	1.37	1974-1983	0.93	1983-1990	0.31
Morgan Harbor	1932-1958	0.19	1958-1974	0.32	1974-1983	0.38	1983-1990	0.32
Mount Airy	1939-1958	0.05	1958-1974	0.08	1974-1983	0.08	1983-1990	0.12
New Orleans	1935-1958	0.17	1958-1974	0.26	1974-1983	0.14	1983-1990	0.15
Oyster Bayou	1931-1956	0.07	1956-1974	0.18	1974-1983	0.15	1983-1990	0.07
Pecan Island	1935-1951	0.06	1951-1974	0.79	1974-1983	0.75	1983-1990	0.62
Point Chicot	1932-1958	0.08	1958-1974	0.08	1974-1983	0.07	1983-1990	0.15
Point au Fer	1931-1956	0.11	1956-1974	0.16	1974-1983	0.17	1983-1990	0.11
Pointe ala Hache	1932-1958	0.28	1958-1974	0.75	1974-1983	0.71	1983-1990	0.75
Pontchatoula	1939-1958	0.07	1958-1974	0.09	1974-1983	0.08	1983-1990	0.05
Rigolets	1932-1958	0.11	1958-1974	0.24	1974-1983	0.26	1983-1990	0.12
Slidell	1939-1958	0.06	1958-1974	0.15	1974-1983	0.05	1983-1990	0.04
Southwest Pass	1932-1958	0.10	1958-1974	0.12	1974-1983	0.02	1983-1990	0.02
Spanish Fort	1936-1958	0.03	1958-1974	0.01	1974-1983	0.003	1983-1990	0.01
Springfield	1939-1958	0.01	1958-1974	0.01	1974-1983	0.03	1983-1990	0.003
St. Bernard	1932-1958	0.29	1958-1974	1.23	1974-1983	0.70	1983-1990	0.26
Sulphur	1933-1955	0.05	1955-1974	1.82	1974-1983	0.40	1983-1990	0.28
Sweet Lake	1933-1955	0.13	1955-1974	1.80	1974-1983	0.84	1983-1990	0.68
Terrebonne Bay	1932-1958	0.18	1958-1974	0.29	1974-1983	0.49	1983-1990	0.35
Thibodaux	1949-1958	0.003	1958-1974	0.02	1974-1983	0.07	1983-1990	0.26
Three Mile Bay	1932-1958	0.08	1958-1974	0.11	1974-1983	0.10	1983-1990	0.19
Timbalier Bay	1932-1958	0.21	1958-1974	0.22	1974-1983	0.41	1983-1990	0.31
Venice	1932-1958	0.61	1958-1974	1.50	1974-1983	0.54	1983-1990	1.60
West Delta	1932-1958	1.41	1958-1974	2.00	1974-1983	1.04	1983-1990	1.64
Yscloskey	1932-1958	0.12	1958-1974	0.60	1974-1983	0.53	1983-1990	0.14

F. Data For The Polynomial Calculation

The data has been enhanced to have one to one annual correspondence. The following is the data describing Abbeville quadrangle. Here, the independent variable x , is the time (year), and the dependent variable y , is the wetland loss (square mile).

Example: Abbeville Quadrangle		
Year	Loss(y)	x
1935	0.07	0
1936	0.07	1
1937	0.07	2
.	.	.
.	.	.
1954	0.07	19
1955	0.23	20
1956	0.23	21
.	.	.
.	.	.
1974	0.23	39
1975	0.24	40
1976	0.24	41
.	.	.
.	.	.
1983	0.24	48
1984	0.43	49
1985	0.43	50
1986	0.43	51
1987	0.43	52
1988	0.43	53
1989	0.43	54
1990	0.43	55

Total number of points = 56.

G. Polynomial Regression-The least- squares method

To fit an m th-degree polynomial to n points (x_i, y_i) , where $x_i : 0...55$, Time(years) starting at 1935 $y_i : \text{wetlands lost}(\text{square miles})$.

Model :

$$(2) \quad y_i = a_0 + a_1x_i + a_2x_i^2 + \dots + a_mx_i^m + e_i$$

where e is the residual (error term). Then,

$$e_i = y_i - (a_0 + a_1x_i + a_2x_i^2 + \dots + a_mx_i^m)$$

Where, n is the number of points (years), and m is the degree of the polynomial.

The sum of the square of the residuals S_r may be expressed as follow:

$$(3) \quad S_r = \sum_{i=1}^n (y_i - a_0 - a_1x_i - a_2x_i^2 - \dots - a_mx_i^m)^2$$

To minimize S_r , all partial derivative of S_r with respect to a_i must be set to zero. This yields *

(4):

$$\frac{\partial S_r}{\partial a_0} = -2 \sum_{i=1}^n (y_i - a_0 - a_1x_i - a_2x_i^2 - \dots - a_mx_i^m)$$

$$\frac{\partial S_r}{\partial a_1} = -2 \sum_{i=1}^n x_i (y_i - a_0 - a_1x_i - a_2x_i^2 - \dots - a_mx_i^m)$$

$$\frac{\partial S_r}{\partial a_2} = -2 \sum_{i=1}^n x_i^2 (y_i - a_0 - a_1x_i - a_2x_i^2 - \dots - a_mx_i^m)$$

$$\frac{\partial S_r}{\partial a_m} = -2 \sum_{i=1}^n x_i^m (y_i - a_0 - a_1x_i - a_2x_i^2 - \dots - a_mx_i^m)$$

These equations should be set to zero. This results the following set of equations:

$$a_0n + a_1\sum x_i + a_2\sum x_i^2 + \dots + a_m\sum x_i^m = \sum y_i$$

$$a_0\sum x_i + a_1\sum x_i^2 + a_2\sum x_i^3 + \dots + a_m\sum x_i^{m+1} = \sum x_i y_i$$

$$a_0 \sum x_i^2 + a_1 \sum x_i^3 + a_2 \sum x_i^4 + \dots + a_m \sum x_i^{m+2} = \sum x_i^2 y_i$$

$$a_0 \sum x_i^m + a_1 \sum x_i^{m+1} + a_2 \sum x_i^{m+2} + \dots + a_m \sum x_i^{2m} = \sum x_i^m y_i$$

Where all summations are from $i = 1$ through n .

The following is the representation of the above equations in the matrix form:

$$(5) \quad \begin{bmatrix} n & \sum x_i & \cdot & \sum x_i^m \\ \sum x_i & \sum x_i^2 & \cdot & \sum x_i^{m+1} \\ \cdot & \cdot & \cdot & \cdot \\ \sum x_i^m & \sum x_i^{m+1} & \cdot & \sum x_i^{2m} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \cdot \\ a_m \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \cdot \\ \sum x_i^m y_i \end{bmatrix}$$

or

$$(6) \quad Aa = D$$

where A is an $(m+1) \times (m+1)$ matrix, a is an $(m+1)$ -vector, and D is an $(m+1)$ -vector.

Standard Error:

The standard error of the estimate $S_{y/x}$ may be calculated using the following formula:

$$S_{y/x} = \sqrt{\frac{S_{y/x}}{n-(m+1)}}$$

Because of the nature of the data, $S_{y/x}$ will not provide the useful information at this stage of the modeling.

H. Sensitivity:

In general, sensitivity is defined as the changes in output due to the disturbance in input or some parameters of the system. This can be written as:

$$y + \Delta y = F(x + \Delta x) \text{ or}$$

$$\begin{aligned} y + \Delta y &= a_0 + a_1(x + \Delta x) + a_2(x + \Delta x)^2 + a_3(x + \Delta x)^3 \\ &= a_0 + a_1x + a_1\Delta x + a_2x^2 + a_2(2x\Delta x + (\Delta x)^2) + a_3x^3 + a_3(3x^2\Delta x + 3x(\Delta x)^2 + (\Delta x)^3) \end{aligned}$$

Therefore,

$$\Delta y = a_1 \Delta x + a_2(2x \Delta x + (\Delta x)^2) + a_3(3x^2 \Delta x + 3x(\Delta x)^2 + (\Delta x)^3)$$

and the ratio $\frac{\Delta y}{\Delta x}$ may be computed as follow:

$$\frac{\Delta y}{\Delta x} = a_1 + a_2(2x + \Delta x) + a_3(3x^2 + 3x\Delta x + (\Delta x)^2)$$

Here, the input factor is time. For a given model, this may be utilized to identify the contribution of various factors to the associated wetland loss in a given time. Of course, this requires additional studies and utilizing a complex data gathering and reproducing methods.

I. Program Description:

The program describes the rate of wetland loss for each quadrangle. The user selects a quadrangle via the mouse, then, the program calculates the parameters of each polynomial model and plots the model for the time under study.

The flowchart for the main function is given in the following page. Proceeding the flowchart, the polynomial model for each quadrangle and the respective graph is also presented. The simulator does not have the capability of printing a hard-copy. Matcad is utilized to print the graphs.

J. Polynomial Models and Graphs

1. Abbeville

$$y = 0.027271 + 0.008941x - 0.000204x^2 + 0.000003x^3$$

2. Barataria

$$y = 0.0950458 + 0.036650x - 0.001691x^2 + 0.000019x^3$$

3. Bay Dogris

$$y = 0.505867 - 0.047341x + 0.003580x^2 - 0.00046x^3$$

4. Bayou Du Large

$$y = 0.230670 - 0.051880x + 0.004626x^2 - 0.000066x^3$$

5. Bayou Sale

$$y = 0.300779 - 0.000309x + 0.000269x^2 - 0.000007x^3$$

6. Belle Isle

$$y = 0.372218 + 0.002912x - 0.000217x^2 + 0x^3$$

7. Black Bay

$$y = 0.277728 - 0.022355x + 0.001321x^2 + 0.000017x^3$$

8. Bonnet Carre

$$y = 0.083621 - 0.006449x + 0.001074x^2 + 0.000019x^3$$

9. Breton Island

$$y = 0.253295 + 0.002873x - 0.000178x^2 + 0.000001x^3$$

10. Caillou Bay

$$y = 0.266973 - 0.017550x + 0.001129x^2 - 0.000015x^3$$

11. Cameron

$$y = -0.274431 + 0.020816x + 0.004280x^2 - 0.000085x^3$$

12. Cat Island

$$y = 0.078912 - 0.002923x + 0.000172x^2 - 0.000002x^3$$

13. Chef Menteur

$$y = 0.469494 + 0.006533x - 0.000353x^2 + 0.000003x^3$$

14. Chen au Tigre

$$y = -0.040286 + 0.028275x - 0.000729x^2 + 0.000006x^3$$

15. Constance Bayou

$$y = 0.592469 + 0.006729x + 0.000054x^2 - 0.000004x^3$$

16. Covington

$$y = -0.002610 + 0.002390x + 0.000073x^2 - 0.000001x^3$$

17. Cut Off

$$y = 0.163992 + 0.008195x + 0.000231x^2 - 0.000006x^3$$

18. Derouen

$$y = 0.240392 + 0.000559x - 0.000068x^2 + 0.000001x^3$$

19. Dulac

$$y = 0.591729 - 0.069771x + 0.003847x^2 - 0.000040x^3$$

20. East Delta

$$y = 1.057749 + 0.004814x + 0.001264x^2 - 0.000030x^3$$

21. Empire

$$y = 0.650746 - 0.091879x + 0.004908x^2 - 0.000049x^3$$

22. Forked Island

$$y = -0.018616 + 0.004934x - 0.000025x^2 + 0x^3$$

23. Fort Livingston

$$y = 0.432106 - 0.028913x + 0.001629x^2 - 0.000019x^3$$

24. Gibson

$$y = -0.135247 + 0.054451x - 0.000969x^2 + 0.000002x^3$$

25. Grand Lake East

$$y = 0.521209 - 0.063253x + 0.003132x^2 - 0.000034x^3$$

26. Grand Lake West

$$y = 0.133981 - 0.054785x + 0.004785x^2 - 0.000067x^3$$

27. Hahnville

$$y = 0.137598 - 0.021982x + 0.002006x^2 - 0.000030x^3$$

28. Hog Bayou

$$y = 0.426081 + 0.024717x - 0.000837x^2 + 0.000037x^3$$

29. Houma

$$y = 0.098438 + 0.006921x + 0.000173x^2 + 0.000002x^3$$

30. Jeanerette

$$y = 0.072568 + 0.002339x - 0.000135x^2 + 0.000002x^3$$

31. Johnsons Bayou

$$y = -0.174601 - 0.023603x + 0.007476x^2 - 0.000129x^3$$

32. Lac des Allemands

$$y = 0.239989 - 0.060944x + 0.004041x^2 - 0.000061x^3$$

33. Lake Decade

$$y = 0.219343 - 0.023754x + 0.002914x^2 - 0.000045x^3$$

34. Lake Felicity

$$y = 0.161858 + 0.008642x + 0.000957x^2 - 0.000015x^3$$

35. Leeville

$$y = 0.363788 - 0.023126x + 0.001113x^2 - 0.000010x^3$$

36. Marsh Island

$$y = 0.221079 - 0.002392x + 0.000402x^2 - 0.000006x^3$$

37. Mitchell Key

$$y = 0.060067 - 0.002225x + 0.000082x^2 - 0.000001x^3$$

38. Morgan City

$$y = 0.285184 - 0.055971x + 0.004597x^2 - 0.000064x^3$$

39. Morgan Harbor

$$y = 0.215333 - 0.009648x + 0.000614x^2 - 0.000007x^3$$

40. Mount City

$$y = 0.043528 + 0.001357x - 0.000030x^2 + 0.000001x^3$$

41. New Orleans

$$y = 0.156477 + 0.000919x + 0.000138x^2 - 0.000003x^3$$

42. Oyster Bayou

$$y = 0.81802 - 0.006325x + 0.000486x^2 - 0.000007x^3$$

43. Pecan Island

$$y = -0.108125 + 0.031890x + 0.000277x^2 - 0.000012x^3$$

44. Point Chicot

$$y = 0.068818 + 0.003347x - 0.000189x^2 + 0.000003x^3$$

45. Point au Fer

$$y = 0.120715 - 0.004251x + 0.000285x^2 - 0.000004x^3$$

46. Pointe ala Hache

$$y = 0.312494 - 0.019210x + 0.001470x^2 - 0.000018x^3$$

47. Pontchatoula

$$y = 0.070410 - 0.000958x + 0.000118x^2 - 0.000018x^3$$

48. Rigolets

$$y = 0.141356 - 0.011900x + 0.000772x^2 - 0.000010x^3$$

49. Slidell

$$y = 0.030559 + 0.003127x + 0.000075x^2 - 0.000003x^3$$

50. Southwest Pass

$$y = 0.091459 + 0.001269x + 0.000004x^2 - 0.000001x^3$$

51. Spanish Fort

$$y = 0.028101 + 0.001097x - 0.000091x^2 + 0.000001x^3$$

52. Springfield

$$y = 0.016064 - 0.001957x + 0.000114x^2 - 0.000002x^3$$

53. St. Bernard

$$y = 0.374544 - 0.048893x + 0.003841x^2 - 0.000054x^3$$

54. Sulphur

$$y = 0.163480 + 0.002793x + 0.003509x^2 - 0.000064x^3$$

55. Sweet Lake

$$y = -0.023669 - 0.008513x + 0.003718x^2 - 0.000062x^3$$

56. Terrebonne Bay

$$y = 0.008147 - 0.014921x + 0.000819x^2 - 0.000009x^3$$

57. Thibodaux

$$y = 0.201594 - 0.008501x + 0.000559x^2 - 0.000007x^3$$

58. Three Mile Bay

$$y = 0.069219 + 0.002578x - 0.000117x^2 + 0.000002x^3$$

59. Timbalier Bay

$$y = 0.242843 - 0.008458x + 0.000381x^2 - 0.000003x^3$$

60. Venice

$$y = 0.413814 + 0.030363x - 0.000284x^2 - 0x^3$$

61. West Delta

$$y = 1.267665 + 0.021683x - 0.000123x^2 - 0.000004x^3$$

62. Yscloskey

$$y = 0.204842 - 0.035825x + 0.002467x^2 + 0.000033x^3$$

K. Functions Descriptions:

`void screen(void)`

Passed to: None

Returned: None

Description: To setup the main layout.

`int flagg( int, int)`

Passed to: (x,y) Position of the left mouse button click

Returned: Index corresponding to each quadrangle

Description: To assign an index number for each quadrangle.

`double findmax(int, double *)`

Passed to: Size of data, Array of points

Returned: The largest point

Description: To find the largest element of a given data.

`void polymatrix(double [4][ ], double *, int, int)`

Passed to: Size of a data, degree of polynomial, and the data

Returned: None

Description: To calculate the elements of the matrix A and the vector D in equation(6).

`void ludecomp(double[5][5],int)`

Passed to: Size of matrix, elements of a matrix

Returned: None

Description: To perform LU Decomposition of a given matrix.

`void forward ( double[5][5], double *, int)`

Passed to: Size of a matrix, elements of a matrix, and elements of corresponding vector

Returned: None

Description: To perform forward elimination and back substitution to solve the system (6).

`void model ( int, int, int, char *)`

Passed to: Starting year, size of the data, degree of the polynomial, and the name of

quadrangle

Returned: None

Description: To find the coefficients of the polynomial model for a given quadrangle. This function opens a file corresponded to a quadrangle, and read the data. Then, the data is utilized for calculating the coefficients of the polynomial model. This function also plots the graph of the polynomial model.

L. Source Code

```
#include<stdio.h>
#include<dos.h>
#include<math.h>
#include<conio.h>
#include<string.h>
#include<graphics.h>
#include<process.h>
#include<stdlib.h>
double findmax(int, double *);
void polymatrix(double [4][],double *, int , int );
void ludcomp(double [5][5], int);
void forward(double [5][5], double *, int );
#define MOUSE 0x33
void screen(void);
int flagg(int, int);
void model(int,int, int,char * );

void main()
{
union REGS regs;
int driver, mode;      // graphics driver and mode
    driver = DETECT;    // auto detect
    // initialize graphics
    initgraph(&driver, &mode, "c:\\borlandc\\bgi");

    char NameBuf[40];
int rega=0;
int a,b,ch,exitt=0,k,j;
clrscr();
screen();
do{
regs.x.ax=0;
int86(MOUSE,&regs,&regs);
regs.x.ax=7; /
regs.x.cx=0;
regs.x.dx=640;
int86(MOUSE,&regs,&regs);
regs.x.ax=8;
regs.x.cx=0;
regs.x.dx=125;
int86(MOUSE,&regs,&regs);
regs.x.ax=1;
int86(MOUSE,&regs,&regs);
```

```

while(rega !=1)
{
    regs.x.ax=5;
    regs.x.bx=0;
    int86(MOUSE,&regs,&regs);
    rega=regs.x.bx;
    rega=(rega & 0x1);
}

    a=regs.x.cx;
    b=regs.x.dx;
regs.x.ax=2;           // hides mouse cursor //
int86(MOUSE,&regs,&regs);
rega=0; // reset rega flag //
ch=flagg(a,b);          // set index for the mouse position on the screen
switch(ch)
{
    case 1:
        strcpy(NameBuf,"abbevi.dat");
        model(5,56,3,NameBuf);
        break;

    case 2:
        strcpy(NameBuf,"Baratari");
        model(9,51,3,NameBuf);
        break;

    case 3:
        strcpy(NameBuf,"Dogris");
        model(2,59,3,NameBuf);
        break;

    case 4:
        strcpy(NameBuf,"DuLarge");
        model(2,59,3,NameBuf);
        break;

    case 5:
        strcpy(NameBuf,"Ba-Sale");
        model(7,53,3,NameBuf);
        break;

    case 6:
        strcpy(NameBuf,"BellIsle");
        model(10,51,3,NameBuf);
        break;
}

```

```
case 7:  
strcpy(NameBuf,"BlackBa");  
model(2,59,3,NameBuf);  
break;
```

```
case 8:  
strcpy(NameBuf,"Bonnet.");  
model(6,55,3,NameBuf);  
break;
```

```
case 9:  
strcpy(NameBuf,"Breton");  
model(2,59,3,NameBuf);  
break;
```

```
case 10:  
strcpy(NameBuf,"Caillou");  
model(2,59,3,NameBuf);  
break;
```

```
case 11:  
strcpy(NameBuf,"Cameron");  
model(3,58,3,NameBuf);  
break;
```

```
case 12:  
strcpy(NameBuf,"Cat-IsI");  
model(2,59,3,NameBuf);  
break;
```

```
case 13:  
strcpy(NameBuf,"Chef-Me");  
model(2,59,3,NameBuf);  
break;
```

```
case 14:  
strcpy(NameBuf,"Chen-Au");  
model(5,56,3,NameBuf);  
break;
```

```
case 15:  
strcpy(NameBuf,"Constan");  
model(2,59,3,NameBuf);  
break;
```

```
case 16:
strcpy(NameBuf,"Covingt");
model(2,59,3,NameBuf);
break;
```

```
case 17:
strcpy(NameBuf,"Cutoff");
model(9,52,3,NameBuf);
break;
```

```
case 18:
strcpy(NameBuf,"Derouen");
model(2,59,3,NameBuf);
break;
```

```
case 19:
strcpy(NameBuf,"Dulac");
model(2,59,3,NameBuf);
break;
```

```
case 20:
strcpy(NameBuf,"E-Delta");
model(2,59,3,NameBuf);
break;
```

```
case 21:
strcpy(NameBuf,"Empire");
model(2,59,3,NameBuf);
break;
```

```
case 22:
strcpy(NameBuf,"Forked");
model(5,55,3,NameBuf);
break;
```

```
case 23:
strcpy(NameBuf,"Ft-Livin");
model(2,59,3,NameBuf);
break;
```

```
case 24:
strcpy(NameBuf,"Gibson");
model(9,52,3,NameBuf);
break;
```

```
case 25:
strcpy(NameBuf,"Grand-E");
model(2,59,3,NameBuf);
break;
```

```
case 26:
strcpy(NameBuf,"Grand-W");
model(3,58,3,NameBuf);
break;
```

```
case 27:
strcpy(NameBuf,"Hahnvill");
model(5,56,3,NameBuf);
break;
```

```
case 28:
strcpy(NameBuf,"Hog-Bay");
model(2,59,3,NameBuf);
break;
```

```
case 29:
strcpy(NameBuf,"Houma");
model(9,52,3,NameBuf);
break;
```

```
case 30:
strcpy(NameBuf,"Jeaneret");
model(7,54,3,NameBuf);
break;
```

```
case 31:
strcpy(NameBuf,"Johnson");
model(3,58,3,NameBuf);
break;
```

```
case 32:
strcpy(NameBuf,"Lac-Des");
model(15,46,3,NameBuf);
break;
```

```
case 33:
strcpy(NameBuf,"L-Decad");
model(1,60,3,NameBuf);
break;
```

case 34:
strcpy(NameBuf,"L-Felic");
model(2,59,3,NameBuf);
break;

case 35:
strcpy(NameBuf,"Leevill");
model(2,59,3,NameBuf);
break;

case 36:
strcpy(NameBuf,"Marsh");
model(2,59,3,NameBuf);
break;

case 37:
strcpy(NameBuf,"Mitchel");
model(2,59,3,NameBuf);
break;

case 38:
strcpy(NameBuf,"Morg_ci");
model(1,60,3,NameBuf);
break;

case 39:
strcpy(NameBuf,"Morg_ha");
model(2,59,3,NameBuf);
break;

case 40:
strcpy(NameBuf,"Mount");
model(9,52,3,NameBuf);
break;

case 41:
strcpy(NameBuf,"N-Orlea");
model(5,56,3,NameBuf);
break;

case 42:
strcpy(NameBuf,"Oyster");
model(1,60,3,NameBuf);
break;

```
case 43:  
strcpy(NameBuf,"Pecan");  
model(5,56.3,NameBuf);  
break;
```

```
case 44:  
strcpy(NameBuf,"Poi-Chi");  
model(2,59,3,NameBuf);  
break;
```

```
case 45:  
strcpy(NameBuf,"Poi-Au");  
model(1,60,3,NameBuf);  
break;
```

```
case 46:  
strcpy(NameBuf,"Poi-Ala");  
model(2,59,3,NameBuf);  
break;
```

```
case 47:  
strcpy(NameBuf,"Pontcha");  
model(9,52,3,NameBuf);  
break;
```

```
case 48:  
strcpy(NameBuf,"Rigolet");  
model(2,59,3,NameBuf);  
break;
```

```
case 49:  
strcpy(NameBuf,"Slidell");  
model(9,52.3,NameBuf);  
break;
```

```
case 50:  
strcpy(NameBuf,"SW-Pass");  
model(2,59.3,NameBuf);  
break;
```

```
case 51:  
strcpy(NameBuf,"Spanish");  
model(6,55.3,NameBuf);  
break;
```



```
case 52:
strcpy(NameBuf,"Spring");
model(9,52,3,NameBuf);
break;

case 53:
strcpy(NameBuf,"St-Bern");
model(2,59,3,NameBuf);
break;

case 54:
strcpy(NameBuf,"Sulphur");
model(3,58,3,NameBuf);
break;

case 55:
strcpy(NameBuf,"Sweet-L");
model(3,58,3,NameBuf);
break;

case 56:
strcpy(NameBuf,"Terrebo");
model(2,59,3,NameBuf);
break;

case 57:
strcpy(NameBuf,"Thiboda");
model(19,42,3,NameBuf);
break;

case 58:
strcpy(NameBuf,"Three-M");
model(2,59,3,NameBuf);
break;

case 59:
strcpy(NameBuf,"Timbali");
model(2,59,3,NameBuf);
break;

case 60:
strcpy(NameBuf,"Venice");
model(2,59,3,NameBuf);
break;
```

```

case 61:
strcpy(NameBuf,"W-Delta");
model(2,59,3,NameBuf);
break;

case 62:
strcpy(NameBuf,"Ysclosk");
model(2,59,3,NameBuf);
break;

case 63:
exitt=1;
break;
}

}
while(exitt!=1);
closegraph();
exit(1);

}

int flagg(int x, int y)
{

if(y<25){
    if(x<43)
        return(1);

    if(x<86 && x>43)
        return(2);

    if(x<129 && x>86)
        return(3);

    if (x<172 && x>129)
        return(4);

    if (x<215 && x>172)
        return(5);

    if (x<258 && x>215)
        return(6);

```

```

if (x<301 && x>258)
return(7);

if (x<344 && x>301)
return(8);

if (x<387 && x>344)
return(9);

if (x<430 && x>387)
return(10);

if (x<473 && x>430)
return(11);

if (x<516 && x>473)
return(12);

if (x<559 && x>516)
return(13);

if (x<602 && x>559)
return(14);

if (x<640 && x>602)
return(15);

}
if(y>25 && y<50){
    if(x<43)
        return(16);
    else
        if(x<86 && x>43)
            return(17);
        else
            if(x<129 && x>86)
                return(18);
            else
                if (x<172 && x>129)
                    return(19);
                else
                    if (x<215 && x>172)
                        return(20);
                    else
                        if (x<258 && x>215)

```

```

return(21);
else if (x<301 && x>258)
    return(22);
else if (x<344 && x>301)
    return(23);
else if (x<387 && x>344)
    return(24);
else
    if (x<430 && x>387)
        return(25);
    else
        if (x<473 && x>430)
            return(26);
        else
            if (x<516 && x>473)
                return(27);
            else
                if (x<559 && x>516)
                    return(28);
                else
                    if (x<602 && x>559)
                        return(29);
                    else
                        if (x<640 && x>602)
                            return(30);

}
if(y>50 && y<75){
    if(x<43)
        return(31);

    if(x<86 && x>43)
        return(32);

    if(x<129 && x>86)
        return(33);

    if (x<172 && x>129)
        return(34);

    if (x<215 && x>172)
        return(35);

    if (x<258 && x>215)
        return(36);

```

```

    if (x<301 && x>258)
    return(37);

    if (x<344 && x>301)
    return(38);

    if (x<387 && x>344)
    return(39);

    if (x<430 && x>387)
    return(40);

    if (x<473 && x>430)
    return(41);

    if (x<516 && x>473)
    return(42);

    if (x<559 && x>516)
    return(43);

    if (x<602 && x>559)
    return(44);

    if (x<640 && x>602)
    return(45);

    }
    if(y>75 && y<100){
        if(x<43)
        return(46);

        if(x<86 && x>43)
        return(47);

        if(x<129 && x>86)
        return(48);

        if (x<172 && x>129)
        return(49);

        if (x<215 && x>172)
        return(50);

```

```

    if (x<258 && x>215)
    return(51);

    if (x<301 && x>258)
    return(52);

    if (x<344 && x>301)
    return(53);

    if (x<387 && x>344)
    return(54);

    if (x<430 && x>387)
    return(55);

    if (x<473 && x>430)
    return(56);

    if (x<516 && x>473)
    return(57);

    if (x<559 && x>516)
    return(58);

    if (x<602 && x>559)
    return(59);

    if (x<640 && x>602)
    return(60);

    }
    if(x<43 && y<125 && y>100)
    return(61);

    if(x>602 && y<125 && y>100)
    return(62);

    if(x>43 && x<82 && y<125 && y>100)

    return(63);
    }
void screen(void)
{
    int i;
    int top_rect[]={0,0, 639,0, 639,100, 0,100,0,0};

```

```

int background[]={0,0,640,0,640,640,0,640,0,0};
int rect1[]={0,100,86,100,86,125,0,125,0,100};
int rect2[]={602,100,639,100,639,125,602,125,602,100};

```

```

    // set fill for rectangle
    setfillstyle(1,9);
    fillpoly(5, background);
    setfillstyle(1,RED);
    fillpoly(5, top_rect);      // draw front rectangle
    fillpoly(5, rect1);
    fillpoly(5, rect2);
    for(i=25;i<76;i+=25)
    line(0,i,639,i);
    for (i=0;i<630;i+=43)
    line (i,0,i,100);
    line(43,100,43,125);
    setcolor(BLUE);

    settextstyle(6,0,0);
    moveto (2,12);
    outtext(" Abbe Bara B_Do B_Du B_Sa Bell Blac Bonn Bert Cail Came Cat Chef Chen Cons
");
    moveto (2,37);
    outtext(" Covi Cut Dero Dula E_De Empi Fork Fort Gibs G_Ea G_We Hahv Hog Houm
Jean ");
    moveto (2,62);
    outtext(" John Lac L_De L_Fe Leev Mars Mitc M_Ci M_Ai Moun N_Or Oyst Peca P_CH
P_Au ");
    moveto (2,87);
    outtext(" P_Al Pont Rigo Slid Sout Span Spri S_Be Sulp Swet Terr Thib Thre Timb Veni");
    moveto (2,112);
    outtext(" West Exit                                Yscl");
    moveto(140,120);

    setcolor(BLACK);
    settextstyle(7,0,USER_CHAR_SIZE);
    setusercharsize(4,1,7,1);
    outtext("The Coefficients of the Polynomial Model for");
}

```

```

void ludecomp(double a[5][5], int n)
{
    int i,j,k;
    float sum;

```

```

for(j=1;j<n;j++){
a[0][j]/=a[0][0];
}

for(j=1;j<n-1;j++){
    for(i=j;i<n;i++){
        sum=0;
        for(k=0;k<j;k++)
            sum+=a[i][k]*a[k][j];
        a[i][j]-=sum;

    }

    for(k=j+1;k<n;k++){
        sum=0;
        for(i=0;i<j;i++)
            sum+=a[j][i]*a[i][k];

        a[j][k]=(a[j][k]-sum)/a[j][j];
    }
}

sum=0;
for(k=0;k<n-1;k++)
    sum+=a[n-1][k]*a[k][n-1];
a[n-1][n-1]-=sum;

}

```

```

void forward(double a[5][5], double *d, int n)
{
    int i,j;
    float sum;
    d[0]=d[0]/a[0][0];

    for(i=1;i<n;i++){
        sum=0;
        for(j=0;j<i;j++)
            sum+=a[i][j]*d[j];
        d[i]=(d[i]-sum)/a[i][i];
    }
    i=n-2;

    while(i>=0){
        sum=0;
        for(j=i+1;j<n;j++)

```



```

sum+=a[i][j]*d[j];
d[i]-=sum;
i--;
}

}

void polymatrix(double a[5][5],double *d, int n, int m)
{
int i,j,k,l;
double sum,y[50];
for(i=0;i<m+1;i++){
    for (j=0;j<i+1;j++){
        k=i+j;
        sum=0;
        for(l=1;l<n;l++){
            sum+= pow(l,k);
            a[i][j]=sum;
            a[j][i]=sum;
        }
        sum=0;
        for(l=0;l<n;l++){
            if(i<1)
                sum+=d[l];
            else
                sum+=d[l]*pow(l,i);
        }
        y[i]=sum;
    }
for(i=0;i<m+1;i++){
    d[i]=y[i];
}
a[0][0]=n;
}

double findmax(int n, double *d)
{
int i;
double large=0;
for (i=0;i<n;i++){
    if(large< d[i])
        large=d[i];
}
return(large);
}

```

```

void model(int start,int n,int m,char *name)
{
FILE *ptr;
int i,size,j,graphwindow[]={200,250,450,250,450,450,200,450,200,250};
int textwindow[]={270,130,380,130,380,160,270,160,270,130};
double a[5][5],*d,data,y1,max;
for(i=0;i<m+1;i++)
for(j=0;j<m+1;j++)
a[i][j]=0;
for(i=0;i<m+1;i++)
d[i]=0;
setfillstyle(1,9);
fillpoly(5,textwindow);
setfillstyle(1,14);
fillpoly(5,graphwindow);
setlinestyle(0,0,3);
setcolor(3);
rectangle(200,250,450,450);
ptr=fopen(name,"r");
moveto(280, 150);
outtext(name);
for(i=0;i<n;i++) {
fscanf(ptr,"%lf",&data);
d[i]=data;
}
max=findmax(n,d);
polymatrix(a,d,n,m);
size=m+1;
ludecomp(a,size);
forward(a,d,size);
gotoxy(10,12);
for(i=0;i<size;i++)
printf("a(%d)=%f\t",i,d[i]);
setlinestyle(0,0,2);
setcolor(0);
y1=400-d[0]*100/max;
for(i=1;i<n;i++){
y=400-(d[0]+d[1]*i+d[2]*i*i+d[3]*pow(i,3))*100/max;
line(start*4+200+4*(i-1),y1,start*4+200+4*i,y);
y1=y;
}
moveto(196,455);
outtext("30 40 50 60 70 80 90");
fclose(ptr);
}

```

Reference

Press, Teukolsky, Vetterling, and Flannery, "Numerical Recipes in C", Second Edition, Cambridge, 1992

Dunbar, Britsch, and Kemp III, Land Lost Rate; Louisiana Coastal Plains, Technical Report GL-90-2, US Army Engineer District, New Orleans, 1992

Chapra, "Numerical Methods For Engineering", 2nd Edition, McGraw Hill, 1988