Chapter 2

Mathematical Models and Numerical Methods

2.1 Application
Logistic Modeling of Population Data

This project deals with the problem of fitting a logistic model to given population data. Thus we want to determine the numerical constants a and b so that the solution $P(t)$ of the initial value problem

$$\frac{dP}{dt} = aP + bP^2, \qquad P(0) = P_0 \tag{1}$$

approximates the given values $P_0, P_1, P_2, \ldots, P_n$ of the population at the times $t_0 = 0, t_1, t_2, \ldots, t_n$. If we rewrite Eq. (1) — the logistic equation with $kM = a$ and $k = -b$ — in the form

$$\frac{1}{P}\frac{dP}{dt} = a + bP, \tag{2}$$

then we see that the points

$$\left(P(t_i), \frac{P'(t_i)}{P(t_i)} \right), \qquad i = 0, 1, 2, \ldots, n$$

should all lie on the straight line with y-intercept a and slope b that is defined by the linear function of P that appears on the right-hand side in Eq. (2).

This observation provides a way to find the values of a and b. If we can determine the approximate values of the derivatives $P_1', P_2', P_3', \cdots$ corresponding to the given population data, then we can proceed with the following agenda:

- First plot the points $(P_1, P_1'/P_1)$, $(P_2, P_2'/P_2)$, $(P_3, P_3'/P_3)$, . . . on a sheet of graph paper with horizontal P-axis.

- Then use a ruler to draw a straight line that appears to approximate these points well.

- Finally measure this straight line's y-intercept a and slope b.

But where are we to find the needed values of the derivative $P'(t)$ of the (as yet) unknown function $P(t)$? It is easiest to use the approximation

$$P_i' = \frac{P_{i+1} - P_{i-1}}{t_{i+1} - t_{i-1}} \tag{3}$$

that is suggested by Fig. 2.1.7 in the text. Note that, if the t-values for the given data points are equally distributed with successive differences $h = t_{i+1} - t_{i-1}$, then the quotient on the right-hand side in (3) is a "symmetric difference quotient" of the type

$$\frac{\Delta P}{\Delta t} = \frac{P(t_i + h) - P(t_i - h)}{2h}$$

that is used to define the derivative $P'(t)$, and is "centered" at the point t_i. For instance, if we take $i = 0$ as 1790, then the U.S. population data in Fig. 2.1.8 give

$$P_1' = \frac{P_2 - P_0}{t_2 - t_0} = \frac{7.240 - 3.929}{20} \approx 0.166$$

for the slope at the point (t_1, P_1) corresponding to the year 1800.

Investigation A
Use (3) to verify the slope figures shown in the final column of the table of Fig. 2.1.8 in the text, and then plot the points $(P_1, P_1'/P_1), (P_2, P_2'/P_2), \ldots, (P_{11}, P_{11}'/P_{11})$ indicated by the asterisks in Fig. 2.1.9. If an appropriate graphing calculator, spreadsheet, or computer program is available, use it find the straight line $y = a + bP$ as in (2) that best fits these points. If not, draw your own straight line approximating these points, and then measure its intercept a and slope b as accurately as you can. Next, solve the logistic equation in (1) with these numerical parameters, taking $t = 0$ in the year 1800. Finally, compare the predicted 20th century U.S. population figures with the actual data listed in Fig. 2.1.4 in the text.

Investigation B
Repeat Investigation A, but take $t = 0$ in 1900 and use only 20th century population data. Do you get a better approximation for the U.S. population during the final decades of the 20th century?

Investigation C
Model similarly the world population data shown in Fig. 2.1.10 in the text. The Population Division of the United Nations predicts a world population of 8.177 billion in the year 2025. What do you predict?

The Method of Least Squares

So you think there ought to be a better way than eyeballing the placement of a ruler on a sheet a paper? There is, and it dates back to Gauss. Writing Q for the left-hand side in Eq. (2), we seek the straight line

$$Q = a + bP \tag{4}$$

in the *PQ*-plane that "best fits" n given data points (P_1, Q_1), (P_2, Q_2), . . . , (P_n, Q_n). Gauss' idea was to choose this line so that it minimizes the *sum of the squares* of the vertical distances between the line and these points. The vertical distance — the difference in *Q*-coordinates above P_i — between the *i*th point (P_i, Q_i) is $d_i = Q_i - (a + bP_i)$, so the sum of the squares of these "errors" is the function

$$f(a,b) = \sum_{i=1}^{n} \left[Q_i - (a + bP_i) \right]^2 \tag{5}$$

of the (as yet) unknown parameters a and b.

To minimize the value $f(a,b)$ as a function of a and b, we calculate the partial derivatives

$$\frac{\partial f}{\partial a} = \sum_{i=1}^{n} 2\left[Q_i - (a + bP_i) \right](-1) = 2a\sum_{i=1}^{n} 1 + 2b\sum_{i=1}^{n} P_i - 2\sum_{i=1}^{n} Q_i$$

and

$$\frac{\partial f}{\partial b} = \sum_{i=1}^{n} 2\left[Q_i - (a + bP_i) \right](-P_i) = 2a\sum_{i=1}^{n} P_i + 2b\sum_{i=1}^{n} P_i^2 - 2\sum_{i=1}^{n} P_i Q_i \, .$$

When we set both partial derivatives equal to zero, we get the pair

$$a\, n + b \sum_{i=1}^{n} P_i = \sum_{i=1}^{n} Q_i \tag{6}$$

$$a \sum_{i=1}^{n} P_i + b \sum_{i=1}^{n} P_i^2 = \sum_{i=1}^{n} P_i Q_i \tag{7}$$

of *linear* equations in the unknowns a and b, with coefficients that are simple combinations of the P- and Q-coordinates of the n given data points. It remains only to solve Eqs. (6)–(7) for the coefficients a and b in the desired straight line (4) that best fits these data points.

The sum of squares of errors in (5) is commonly denoted by SSE. Then the **average error** in the linear approximation (4) is defined by

$$\text{average error} = \sqrt{\frac{\text{SSE}}{n}}\,. \tag{8}$$

In the following paragraphs we illustrate this least squares approach using *Maple*, *Mathematica*, and MATLAB. In each we consider the $n = 6$ given data points

(1.1, 2.88), (1.9, 4.36), (3.2, 6.98), (4.6, 9.86), (5.3, 11.20), (6.5, 13.54).

You can automate similarly your calculations for Investigations A, B, and C.

Using *Maple*

To set up Eqs. (6) and (7) we first enter the lists

```
P := [1.1, 1.9, 3.2, 4.6, 5.3, 6.5]:
Q := [2.88, 4.36, 6.98, 9.86, 11.20, 13.54]:
```

of the P- and Q-coordinates of the

```
n := 6:
```

given data points. Next we use the *Maple* **sum** function to calculate the sums that appear as coefficients of a and b in Eqs. (6)-(7).

```
sumP := evalf(sum( P[i], i = 1..n)):
sumQ := evalf(sum( Q[i], i = 1..n)):
sumPsq := evalf(sum( P[i]^2, i = 1..n)):
sumPQ := evalf(sum( P[i]*Q[i], i = 1..n)):
```

Then the two equations we want to solve are defined by

```
eq6 := n*a + sumP*b = sumQ;
```

$$eq6 := 6\,a + 22.6\,b = 48.82$$

```
eq7 := sumP*a + sumPsq*b = sumPQ;
```

$$eq7 := 22.6\,a + 106.56\,b = 226.514$$

Now we simply solve these two equations for the coefficients a and b using the command

```
soln := fsolve({eq6,eq7}, {a,b});
```

$$\{a = .6457449574, b = 1.988740277\}$$

```
asoln := soln[1]:   bsoln := soln[2]:
a := rhs(asoln);    b := rhs(bsoln);
```

$$a := .6457449574$$
$$b := 1.988740277$$

Thus the least squares best fit in Eq. (4) takes the form

$$Q \;=\; 0.6457 + 1.9887\,P$$

(rounding the coefficients to four decimal places). Finally the differences between the actual values $\{Q_i\}$ and the predicted values $\{0.6457 + 1.9887\,P_i\}$ are given by

```
errors := seq(Q[i] - (a + b*P[i]), i = 1..n);
```

$$errors := .0466, -.0644, -.0297, .0660, .0139, -.0326$$

(again rounding to four decimal places). The sum of squares of errors, and then the average error, are given by

```
SSE := sum(errors[i]^2,i = 1..n);
AveError := sqrt(SSE/n);
```

$$AveError := .0462$$

Using *Mathematica*

To set up Eqs. (6) and (7) we first enter the lists

```
P = {1.1, 1.9, 3.2, 4.6, 5.3, 6.5};
Q = {2.88, 4.36, 6.98, 9.86, 11.20, 13.54};
```

of the P- and Q-coordinates of the

```
n = 6;
```

given data points. Next we use the *Mathematica* **Sum** function to calculate the sums that appear as coefficients of a and b in Eqs. (6)-(7).

```
sumP = Sum[ P[[i]], {i, 1,n} ];
sumQ = Sum[ Q[[i]], {i, 1,n} ];
sumPsq = Sum[ P[[i]]^2, {i, 1,n} ];
sumPQ = Sum[ P[[i]]*Q[[i]], {i, 1,n} ];
```

Then the two equations we want to solve are defined by

```
eq6 = n*a + sumP*b == sumQ
```

$6a + 22.6b == 48.82$

```
eq7 = sumP*a + sumPsq*b == sumPQ
```

$22.6\ a + 106.56\ b == 226.514$

Now we simply solve these two equations for the coefficients a and b using the command

```
soln = NSolve[{eq6,eq7}, {a,b}]
```

$\{\{a \to 0.645745, b \to 1.98874\}\}$

```
a = First[a /. soln];
b = First[b /. soln];
```

Thus the least squares best fit in Eq. (4) takes the form

$$Q = 0.6457 + 1.9887\ P$$

(rounding the coefficients to four decimal places). Finally the differences between the actual values $\{Q_i\}$ and the predicted values $\{0.6457 + 1.9887\ P_i\}$ are given by

```
errors = Q - (a + b P)
```

$\{0.0466, 0.0644, -0.0297, 0.0660, 0.0139, -0.0326\}$

(again rounding to four decimal places). The sum of squares of errors, and then the average error, are given by

```
SSE = Sum[errors[[i]]^2, {i,1,n}];
AveError = Sqrt[SSE/n]
```

0.0462169

Using MATLAB

To set up Eqs. (6) and (7) we first enter the lists

```
P = [1.1, 1.9, 3.2, 4.6, 5.3, 6.5];
Q = [2.88, 4.36, 6.98, 9.86, 11.20, 13.54];
```

of the *P*- and *Q*-coordinates of the

```
n = 6;
```

given data points. Next we use the MATLAB **sum** function to calculate the sums that appear as coefficients of a and b in Eqs. (6)-(7).

```
sumP = sum(P);
sumQ = sum(Q);
sumPsq = sum(P.*P);
sumPQ = sum(P.*Q);
```

Then the two symbolic equations **eq6 = 0** and **eq7 = 0** that we want to solve are defined by

```
eq6 = n*a + sumP*b - sumQ

6*a+113/5*b-2441/50

eq7 = sumP*a + sumPsq*b - sumPQ

113/5*a+2664/25*b-7969752859329691/35184372088832
```

Apparently MATLAB is rationalizing the numerical coefficients. For instance, checking the "big fraction" we see here:

```
sumPQ

sumPQ =
   226.5140

7969752859329691/35184372088832

ans =
   226.5140
```

We proceed to solve symbolically for a and b.

```
soln = solve(eq6,eq7)

soln =
   a: [1x1 sym]
   b: [1x1 sym]
```

This means that **`soln`** is a structure with two symbolic fields **`a`** and **`b`**. The numerical values of these coefficients are given by

```
a = numeric(soln.a)
a =
    0.6457

b = numeric(soln.b)
b =
    1.9887
```

Thus the least squares best fit in Eq. (4) takes the form

$$Q = 0.6457 + 1.9887\,P$$

(with the coefficients rounded to four decimal places). Finally the differences between the actual values $\{Q_i\}$ and the predicted values $\{0.6457 + 1.9887\,P_i\}$ are given by

```
errors = Q - (a + b*P)

errors =
  0.0466  -0.0644  -0.0297   0.0660   0.0139  -0.0326
```

The sum of squares of errors, and then the average error, are given by

```
SSE = sum(errors.^2);
AveError = sqrt(SSE/n)

AveError =
   0.0462
```