CSDL Informal Technical Note No. 2

# PRINCIPAL COMPONENT DIRECTION CURRENT EVENT ANALYSIS: PROGRAM DOCUMENTATION

Silver Spring, Maryland
June 2002

U.S. DEPARTMENT OF COMMERCE
National Oceanic and Atmospheric Administration
National Ocean Service
Coast Survey Development Laboratory

NOTICE

CSDL Informal Technical Notes present work in progress or summaries of results that are not appropriate to be published as either formal NOAA Office of Coast Survey Technical Reports or the less formal Technical Memoranda. Results are intended primarily for use within CSDL. Scientific review of the material is minimal, and CSDL makes no warranty as to its validity or completeness.

CSDL Informal Technical Note No. 2

# PRINCIPAL COMPONENT DIRECTION CURRENT EVENT ANALYSIS: PROGRAM DOCUMENTATION

Philip H. Richardson
Richard A. Schmalz, Jr.

Silver Spring, Maryland
June 2002


U.S. DEPARTMENT OF COMMERCE
National Oceanic and Atmospheric Administration
National Ocean Service, Coast Survey Development Laboratory

## TABLE OF CONTENTS

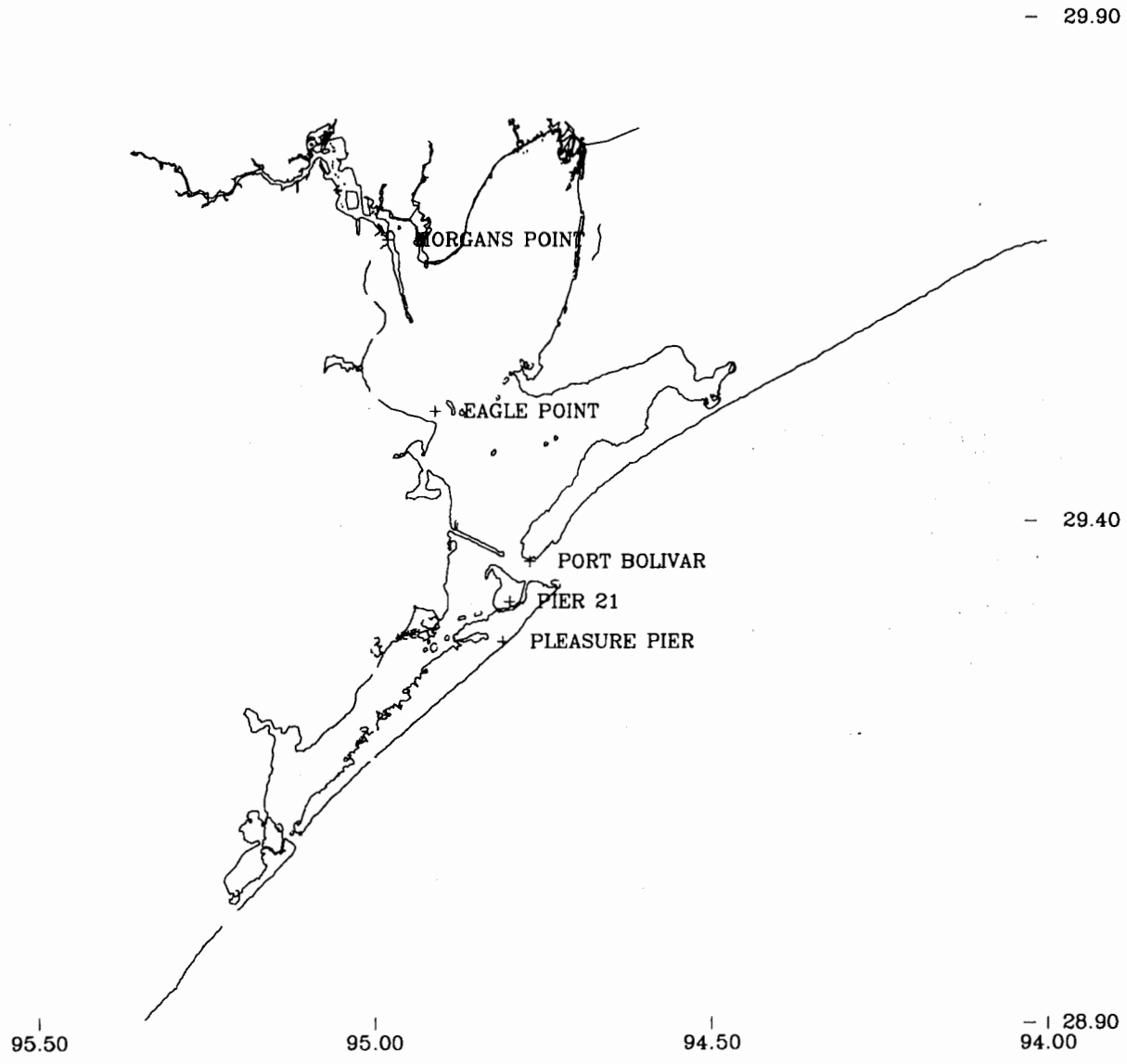
## LIST OF FIGURES



## LIST OF TABLES

## LIST OF COMPUTER PROGRAM LISTINGS

# GALVESTON BAY BASE MAP




Galveston Bay area base map showing locations mentioned in this report.

## ABSTRACT

The National Ocean Service (NOS), as part of its Houston/Galveston Physical Oceanographic Real Time System (PORTS), is developing a nowcast/forecast system to predict water level and currents within Galveston Bay and the Houston Ship Channel. Preliminary nowcast/forecast requirements are for a daily 36 hour forecast initiated from a continuous 24 hour nowcast as outlined by Schmalz and Richardson (1996). To assess the forecast of both flood and ebb current speeds along the principal component direction, a set of seven programs has been developed. The set of programs focuses on the ability to forecast an "event" situation. An "event" occurs when the observed current speed rises above a specified high level value on flood, or falls below a specified low level value on ebb. The programs are described and a sample application for September 2000 is presented. In Appendix A, the complete job control and program control files are provided.

## 1. INTRODUCTION

A Physical Oceanographic Real Time System (PORTS) has been installed in Galveston Bay to provide the navigation community with real time water level and current information (Frey, 1991; Bethem and Frey, 1991). The development of the PORTS is in response to the results of a mini-project conducted by NOS in 1988, in which NOS current predictions within the Bay were found to be outside the range of NOS standards (Williams et al., 1990). The present PORTS consists of five tide gauges and two permanent Acoustic Doppler Current Profilers (ADCP). Conductivity/temperature measurement systems have been installed at several tide gauges.

To complement the PORTS a nowcast/forecast system has been designed (Schmalz and Richardson, 1996) based on the National Ocean Service (NOS) Galveston Bay three-dimensional hydrodynamic model (Schmalz and Richardson, 1998a) and the National Weather Service (NWS) Aviation atmospheric model. To simulate currents within the Houston Ship Channel (HSC), a finer resolution three-dimensional HSC model has been developed. The Galveston Bay model is used to provide Bay wide water level and near entrance current forecasts as well as to directly provide water levels, density, and turbulence quantities to the HSC model for use in a one-way coupling. The combined model set forms the initial hydrodynamic component of the nowcast/forecast system (Schmalz and Richardson, 1998b).

The present nowcast/forecast system is used to provide experimental daily 24 hour nowcasts and 36 hour forecasts of water levels, currents, salinity and temperature at the locations of the PORTS instruments. The evaluation of forcast inputs has been considered by Richardson and Schmalz (1999). Here, a set of seven programs has been developed to evaluate nowcast/forecast principal component direction current speeds to assess the ability to forecast an "event" situation. An "event" occurs when the observed current speed rises above a specified high level value on flood, or falls below a specified low level value on ebb.

When forecasting events, there are three possible outcomes. A "success" occurs when the model successfully predicts an observed event. A "failure" occurs when the model fails to predict an observed event. A "false alarm" occurs when the model predicts an event not seen in the observed data. When the forecast model performs well, the number of successes should be high, and the number of failures and false alarms should be low.

The seven programs are described below.

1. Reform_ports.f was written to read PORTS (observed) current data, then to reformat into standard component format for analysis. PORTS current data is acquired from the PORTS-INFOHUB internet site (http://ports-infohub.nos.noaa.gov).

2. Regap.f was created to check for gaps in the PORTS current data and then to fill those gaps.

3. Reform2.f was created to select hourly values from six minute observed current data. The program will handle data with non-uniform time intervals.

4. Read_NF.curr.f was developed to read current speed and direction results from the NOS Galveston Bay/Houston Ship Channel model. The program reads current data from the 00z files in the form of U and V components. These curvilinear U and V components are converted to U and V with respect to the X,Y axis. The U, V components are then written to both nowcast and forecast output files for stations included in the analysis.

5. Match.event.f was originally written to perform a one year water level event comparison between the East Coast Ocean Forecast System (ECOFS) and the NWS Techniques Development Laboratory (TDL) surge model. The program was revised slightly to evaluate the Houston/Galveston nowcast/forecast water level model results and further revised herein to analyze current data, and renamed match.evnt_crnt.f. For observed and model data, the program reads the U and V components of current velocity. A call is made to subroutine prDirection to calculate the current speed along the principal direction (particular to each station). After the principal component speed has been determined, the program follows the same algorithm as match.event.f, the water level version.

6. Curr.prdir.pro is written in the IDL programming language, and will plot the observed and model current speed along the principal component direction. Also plotted are two lines which depict the critical high and low level values (Flood and Ebb, respectively). Curr.prdir.pro generates only one plot per page.

7. Curr.multcurv.pro, also an IDL program, was created to generate plots of observed versus nowcast and observed versus predicted current speeds on one page, then observed versus forecast and observed versus adjusted forecast current speeds on the second page.

Reform_ports.f, regap.f, reform2.f, read_NF.curr.f, and match.evnt_crnt.f are written in FORTRAN, and are run on the OPSEA system. Curr.prdir.pro and curr.multcurv.pro are written in IDL and are also run on the OPSEA system.

In Chapter 2, descriptions of each program are provided as well as program listings. In Chapter 3, a sample application for September 2000 is discussed in terms of output tables and plots. In Chapter 4, recommendations for the operational use of these assessment programs are presented. In addition, possible future enhancements to the programs are discussed. In Appendix A, the job control and control files for each of the seven programs are provided.

## 2. PROGRAM DESCRIPTIONS

### 2.1. Program Reform_ports.f

This program reads PORTS current data and reformats the data. The listing for reform_ports.f is given in Program Listing 2.1. The raw current data filename is read in from the control file. Line number 110 is the read statement. The program reads the month, day, year, hour, and minute, the time zone, then reads the current speed and direction. The calendar month and day are converted to Julian day (number of days elapsed since the beginning of the year) with a call to subroutine conctj, a standard routine. The speed and direction values are converted to U (East) and V (North) components by subroutine calc_uv. Statement 146 writes the U and V component values to output.

Reform_ports.f assumes the current data to be on a fixed six-minute interval. Data are decimated to form an hourly interval dataset for further analysis. Reform_ports.f was used to reformat PORTS data obtained from the PORTS-INFOHUB during September and October 2000. For data sets where there are gaps and the data interval is not on uniform six-minute intervals, two other programs, regap.f and reform2.f, are available. These programs are discussed in sections 2.2 and 2.3.

```
1       c Program Name : reform_ports.f
2       c
3       c Purpose : To read PORTS (observed) current data, then
4       c           to reformat in standard CFS U,V component (1x,3f9.4)
5       c           format.  Current speeds are recorded in knots
6       c           in PORTS data file.  Conversion factor (knts_mps)
7       c           converts current speed to m/s.
8       c           PORTS current data is acquired off the internet
9       c           from the ports-infohub site.
10      c
11      c Location : /usr/people/philr/galves/NF_eval/currents/ports/
12      c
13      c Subroutines Called : conctj, calc_uv
14      c
15      c Author : Phil Richardson
16      c
17      c Version Date : Ncvember 20, 2000
18
19      ******************************************************
20
21            character*3  timezn
22            character*15 filenm,fileout
23            character*40 line
24
25      ******************************************************
26
27      c Read from Control file:
28      c
29      c     strt_time - start time
30
31
32            read(5,*)idebug
33            read(5,*)strt_time
34            read(5,1)filenm
35            write(6,1)filenm
36            read(5,1)fileout
37            write(6,1)fileout
38
39      c     Conversion factor ! convert knots to m/s
40            rknts_mps = 0.5144
41
42
43          1 format(1x,a15)
44
45      ******************************************************
46
47      c Open PORTS data file to read current speed and direction,
48      c then open output file.
49
50            lun = 8
51            open(lun,file=filenm,form='formatted',
52           *     status='old')
53
54            lunout = 9
55            open(lunout,file=fileout,form='formatted')
56
```

Program Listing 2.1. Reform_ports.f

```
57      ******************************************************
58
59      c Position PORTS file to start time, read first value
60
61         50 continue
62            read(lun,11)imonth,iday,iyear,ihour,imin,timezn,
63           *            speed,idir
64
65            if(imonth.eq.0.and.iday.eq.0)goto 50
66            call conctj(jday,imonth,iday,iyear)
67            rjday = float(jday) + float(ihour)/24.0 +
68           *        float(imin)/1440.0
69            if(rjday.lt.strt_time)then
70               goto 50
71            else
72               write(6,14)strt_time,rjday,imonth,iday,iyear,
73           *              ihour,imin,speed,idir
74               if(imin.ne.58)then
75                  write(6,*)'start time not on hour'
76                  stop
77               endif
78               ihour = ihour + 1
79               imin = imin - 58
80               rjday = float(jday) + float(ihour)/24.0 +
81           *           float(imin)/1440.0
82               write(6,15)rjday
83
84               speed = speed * rknts_mps
85               call calc_uv(speed,idir,ucom,vcom)
86               write(lunout,21)rjday,ucom,vcom
87            endif
88
89
90         14 format(/,' start time =',f9.4,
91           *         ', time of start record is',f9.4,/,
92           *       1x,i3,'/',i2,'/',i4,2x,i2,':',i2,
93           *       /,' speed = ',f7.3,', dir = ',i4)
94         15 format(' Adjusted start time PORTS file ',f8.3)
95
96      !------------------------------------------------------
97
98      c Read remaining values
99
100
101           write(6,99)
102
103           ncnt = 0
104        100 continue
105     c     Account for 6 minute data by reading every 10th value
106           do 1=1,9
107              read(lun,101,end=120)line
108           enddo
109
110           read(lun,11)imonth,iday,iyear,ihour,imin,timezn,
111          *            speed,idir
112           if(idebug.eq.1)then
```

Program Listing 2.1. Reform_ports.f (continued)

```
113            write(6,102)imonth,iday,iyear,ihour,imin,
114           *              speed,idir
115            endif
116
117
118            if(imonth.eq.0.and.iday.eq.0)then
119               write(6,104)
120               read(lun,11)imonth,iday,iyear,ihour,imin,timezn,
121           *                speed,idir
122               write(6,102)imonth,iday,iyear,ihour,imin,speed,idir
123               backspace lun
124               imin = imin - 4
125               iflag = 1
126            else
127               if(imin.ne.58)then
128                  write(6,103)
129         110      continue
130                  read(lun,11)imonth,iday,iyear,ihour,imin,timezn,
131           *                  speed,idir
132                  write(6,102)imonth,iday,iyear,ihour,imin,speed,
133           *                  idir
134                  if(imin.ne.58)goto 110
135               endif
136               ihour = ihour + 1
137               imin = imin - 58
138            endif
139
140            call conctj(jday,imonth,iday,iyear)
141            rjday = float(jday) + float(ihour)/24.0
142            write(6,*)rjday
143
144            speed = speed * rknts_mps
145            call calc_uv(speed,idir,ucom,vcom)
146            write(lunout,21)rjday,ucom,vcom
147
148
149            ncnt = ncnt + 1
150            goto 100
151
152         120 continue
153
154
155          99 format(/)
156         101 format(a40)
157         102 format(1x,i2,'/',i2,'/',i4,2x,i2,':',i2,f8.3,i5)
158         103 format(' time of record not on hour, go to next',
159           *       ' record which is on hour')
160         104 format('Program has read a blank record, go to next',
161           *       ' record')
162
163       !-------------------------------------------------------
164
165          11 format(i2,1x,i2,1x,i4,1x,i2,1x,i2,1x,a3,f7.3,1x,i3)
166          21 format(1x,3f9.4)
167
168       *********************************************************
```

Program Listing 2.1. Reform_ports.f (continued)

```
169
170             stop
171             end
```

Program Listing 2.1. Reform_ports.f (continued)

```
      SUBROUTINE CONCTJ (IJD,IMON,IDAY,IYR)

C**** THIS SUBROUTINE CONVERTS CALENDER TO JULIAN DAY (IJD)
C
      DIMENSION   IDTBLE(12),ILTBLE(12)
C
      DATA (IDTBLE(I),I=1,12)/1,32,60,91,121,152,182,213,244,
     1                        274,305,335/
      DATA (ILTBLE(I),I=1,12)/1,32,61,92,122,153,183,214,245,
     1                        275,306,336/
C
C**** TEST FOR LEAP YEAR
C
      ISW = 1
      IF (MOD(IYR,4).EQ.0)   ISW = 2

      GO TO (9,10)   ISW
    9 IJD = IDTBLE(IMON) + IDAY - 1
      RETURN
   10 IJD = ILTBLE(IMON) + IDAY - 1
      RETURN
      END
```

```
      subroutine calc_uv(sp,idr,u,v)

c     Purpose : Given current speed and direction,
c               calculate U and V components.
c
c     Date : July 9, 2001


c     Input Arguments :
c
c     speed - current speed
c       idr - current direction

*******************************************************

      dir = float(idr)
      rwd = dir/57.2958
      v = cos(rwd) * sp
      u = sin(rwd) * sp

      return
      end
```

Program Listing 2.1. Reform_ports.f (continued)

## 2.2. Program regap.f

Regap.f was created to read PORTS current data, check for gaps and then to fill the gaps with a null value (9.99). This program assumes the data to be in GMT, and converts the times to local standard time. The listing of regap.f is given in Program Listing 2.2.

```
1       c     Program Name : regap.f
2       c
3       c     Author : Phil Richardson
4       c
5       c     Purpose : This program is run on the OPSEA to
6       c               check for gaps in Galveston observed (PORTS)
7       c               current data.  The program is written to
8       c               fill the gaps in the six minute data with
9       c               the null value (9.99).
10      c               This particular current data was acquired
11      c               from Karen Earwaker.  Karen's data is in
12      c               GMT and must be converted to local time.
13      c
14      c     Language : FORTRAN 77
15      c
16      c     Version date : July 6, 2001
17
18      ******************************************************
19
20            character*3  option
21            character*9  filefix
22            character*16 filenm
23
24      ******************************************************
25
26      c     Variables read from control :
27      c
28      c                   lun  - logical unit number
29      c              filenm() - filename (with path)
30      c              filefix  - name of file with null values
31      c                         inserted in gaps
32
33
34            read(5,*)lun
35            read(5,23)filenm
36            read(5,24)filefix
37            read(5,*)adjust
38
39
40         19 format(1x,a3)
41         23 format(a16)
42         24 format(a9)
43
44      ******************************************************
45
46      c     Open output (filefix) file.
47
48            lunfix = 9
49            open(lunfix,file=filefix,form='formatted')
50
51      c******************************************************
52
53      c Constant Values :
54      c
55      c        spdnull - null value for current speed
56      c       idirnull - null value for current direction
```

Program Listing 2.2. Regap.f

```
c        time_dif - time difference (to be compared with)
c                   between new and old time values
c          amount - decimal value for portion of day
c                   equivalent to 6 minutes
c          adjust - 6 hrs, 0.25 of day
c
c Initialization -

      spdnull =  9.99
      idirnull = 999
      time_dif = 0.0042
      amount = 1.0/240.0


      ngap = 0

      open(lun,file=filenm,form='formatted',
     *     status='old')

c     Read first line of data file
      read(lun,*)time_ln1,sp,idir
      time_adj = time_ln1 + adjust
      write(lunfix,105)time_adj,sp,idir

      time_ln1jd = time_ln1

      time_old = time_ln1
      time_oldjd = time_old


   10 continue

         read(lun,*,end=20)time_read,sp,idir
         time_adjust = time_read + adjust
         time_new = time_read
         time_newjd = time_read
         diff = time_new - time_old
         if(diff.lt.0.001)then
            write(6,*)time_read
            write(6,*)time_old
         endif
         iyr = iyear_ref

         if(diff.lt.time_dif)then
            write(lunfix,105)time_adjust,sp,idir
         endif

         if(diff.ge.time_dif)then
            ngap = ngap + 1
            if(ngap.eq.1)write(6,111)filenm

            write(6,112)time_old,time_new
            rnline = diff * 240.0
            nline = nint(rnline) - 1
            write(6,115)nline
            time_hrsav = time_old
```

Program Listing 2.2. Regap.f (continued)

```
113                time_dysav = time_old
114                do 120 nl=1,nline
115                   nhr = mod(nl,10)
116                   ndy = mod(nl,240)
117                   write(6,*)nl,nhr
118                   if(ndy.eq.00)then
119                      timefix = time_dysav + 1.0
120                      time_hrsav = timefix
121                      time_dysav = timefix
122                      time_old = timefix
123                      timefix_adj = timefix + adjust
124                      write(lunfix,105)timefix_adj,spdnull,idirnull
125                      goto 120
126                   endif
127                   if(nhr.eq.0)then
128                      timefix = time_hrsav + 1.0/24.0
129                      time_hrsav = timefix
130                      time_old = timefix
131                      timefix_adj = timefix + adjust
132                   else
133                      timefix = time_old + amount
134                      time_old = timefix
135                      timefix_adj = timefix + adjust
136                   endif
137                   write(lunfix,105)timefix_adj,spdnull,idirnull
138          120      continue
139                   write(lunfix,105)time_adjust,sp,idir
140             endif
141
142             time_old = time_new
143             time_oldjd = time_newjd
144             goto 10
145
146           20  continue
147
148             write(6,123)time_read
149             write(6,124)ngap
150
151
152
153          102 format(1x,'file start time ',f8.3,2x,i2,'/',i2,'/',i2)
154          105 format(1x,f9.5,f5.2,1x,i3)
155          111 format(1x,'Gaps in file : ',/,1x,a16,/)
156          112 format(1x,'old time',f9.4,' new time',f10.4)
157          114 format(1x,'Gap ends ',i2,'/',i2,'/',i2)
158          115 format(1x,'Gap of ',i4,' lines')
159          123 format(/,1x,'file stop time ',f8.3)
160          124 format(1x,i3,' gaps')
161
162       ********************************************************
163
164
165             stop
166             end
```

Program Listing 2.2. Regap.f (continued)

## 2.3. Program reform2.f

Reform2.f was created to select hourly values from the output of regap.f. The current data output from regap.f is six minute data, but not necessarily on a uniform time interval. The listing for reform2.f is given in Program Listing 2.3.

```
1       c Program Name : reform2.f
2       c
3       c Author : Phil Richardson
4       c
5       c Purpose : This program is run to pick out hourly values
6       c           from 6 minute current data.
7       c
8       c Location : /usr/people/philr/galves/NF_eval/currents/ports/
9       c
10      c Version Date : July 11, 2001
11
12      ******************************************************
13
14            parameter (ndatpts=700,nstreams=50)
15
16            character*15 fileout
17            character*16 filenm
18
19            dimension rjtime(ndatpts),crnt_speed(ndatpts),
20           *          icrnt_dir(ndatpts)
21
22      ******************************************************
23
24      c Read from control file:
25
26            read(5,*)idebug
27            read(5,'(a16)')filenm
28            write(6,1)filenm
29            read(5,'(a15)')fileout
30            read(5,*)start_time
31            read(5,*)end_time
32
33      c Conversion factor
34            rknts_mps = 0.5144
35
36            t1hr = 0.04167
37            t1dy = 1.0
38            day_time = start_time
39            hour_time = start_time
40
41
42          1 format(' Input data filename : ',a16)
43
44      ******************************************************
45
46      c Open input data file, then open output file
47
48
49            lun = 8
50
51            open(lun,file=filenm,form='formatted',
52           *     status='old')
53
54
55            lunout = 9
56
```

Program Listing 2.3. Reform2.f

```
57            open(lunout,file=fileout,form='formatted')
58
59      ******************************************************
60
61      c Locate start time for input data file
62
63        100 continue
64            read(lun,*)rtime,speed,idir
65            if(rtime.lt.start_time)then
66               goto 100
67            endif
68
69            write(6,101)rtime
70
71      c     write(6,111)rtime,speed,idir
72            if(idebug.eq.1)write(lunout,111)rtime,speed,idir
73            if(idir.gt.900)then
74               ucom = 99.99
75               vcom = 99.99
76            else
77               speed = speed * rknts_mps
78
79               call calc_uv(speed,idir,ucom,vcom)
80            endif
81            write(lunout,112)rtime,ucom,vcom
82
83            day_time = day_time + t1dy
84            hour_time = hour_time + t1hr
85
86
87        110 continue
88            read(lun,*,end=125)rtime,speed,idir
89            if(rtime.gt.end_time)then
90               write(6,*)'End Time reached'
91               goto 125
92            endif
93            if(rtime.ge.day_time)then
94               write(6,111)rtime,speed,idir
95               backspace lun
96               backspace lun
97
98               read(lun,*)rtime1,speed1,idir1
99               read(lun,*)rtime2,speed2,idir2
100              write(6,112)rtime1,rtime2
101
102              diff1 = abs(rtime1 - day_time)
103              diff2 = abs(rtime2 - day_time)
104              write(6,112)diff1,diff2
105
106              if(diff1.lt.diff2)then
107                 rtime = rtime1
108                 speed = speed1 * rknts_mps
109                 idir = idir1
110                 write(6,111)rtime1,speed1,idir1
111              endif
112              if(diff2.lt.diff1)then
```

Program Listing 2.3. Reform2.f (continued)

```
113                rtime = rtime2
114                speed = speed2 * rknts_mps
115                idir = idir2
116             endif
117
118             if(idir.gt.900)then
119                ucom = 99.99
120                vcom = 99.99
121             else
122                call calc_uv(speed,idir,ucom,vcom)
123             endif
124
125             write(lunout,112)rtime,ucom,vcom
126             hour_time = day_time + t1hr
127             day_time = day_time + t1dy
128             goto 110
129          endif
130
131
132          if(rtime.lt.hour_time)then
133             goto 110
134          endif
135          if(rtime.ge.hour_time)then
136             hour_check = abs(rtime - day_time)
137             if(hour_check.lt.0.03)then
138                goto 110
139             endif
140             write(6,111)rtime,speed,idir
141             if(idir.gt.900)then
142                ucom = 99.99
143                vcom = 99.99
144             else
145                speed = speed * rknts_mps
146
147                call calc_uv(speed,idir,ucom,vcom)
148             endif
149             write(lunout,112)rtime,ucom,vcom
150             hour_time = hour_time + t1hr
151             goto 110
152          endif
153
154
155
156       125 continue
157
158    ******************************************************
159
160       101 format(' start time at ',f9.5)
161       111 format(1x,f10.5,2x,f5.2,5x,i3)
162       112 format(1x,3f9.4)
163       113 format(1x,3f10.5)
164
165          stop
166          end
```

Program Listing 2.3. Reform2.f (continued)

```
1             subroutine calc_uv(sp,idr,u,v)
2
3       c     Purpose : Given current speed and direction,
4       c               calculate U and V components.
5       c
6       c     Date : July 9, 2001
7
8
9
10      c     Input Arguments :
11      c
12      c     speed - current speed
13      c       idr - current direction
14
15      ******************************************************
16
17            dir = float(idr)
18            rwd = dir/57.2958
19            v = cos(rwd) * sp
20            u = sin(rwd) * sp
21
22            return
23            end
```

Program Listing 2.3. Reform2.f (continued)

## 2.4 Program Read_NF.curr.f

The computer listing for Program Read_NF.curr.f is given in Computer Listing 2.2. For each of the stations included in our analysis, the program first reads nowcast and forecast output filenames from the control file. Also read is mlevel, the model level used for analysis. Line 85 begins the loop in which the daily 00z files are read. Nfiledays is the number of daily files to be read (30 for the September 2000 sample application). The Galveston Bay model (GBM) 00z files are stored in filenm_gbm(nf), while the Houston Ship Channel model (HSCM) 00z files are stored in filename_hsc(nf).

Read_NF.curr.f accounts for days in which a 00z file is missing in either the GBM or the HSCM. Read from the control file are nmiss_gbm and nmiss_hsc, the number of missing 00z files for each. The program then dummy reads over the particular days that 00z files are missing for both the GBM and HSCM. For periods of time of missing data, the program substitutes a null value.

In order to convert U,V components of current velocity from the model's curvilinear coordinates to U and V with respect to the X-East axis and Y-North axis, respectively, model grid longitude and latitude data must be read for both the GBM grid and the HSCM grid. An angle is calculated for each grid cell which corresponds to a station location. The conversion calculation is carried out with a call to subroutine calc_uvnew.

The 500 loop, which begins on line 216, loops through the days of the month from nf equals one through nfiledays. Both the GBM and the HSCM 00z files are opened. The program first reads through 24 hours of nowcast current data. Lines 272 through 288 read the nowcast U,V component values from the GBM output file. Lines 320 through 328 read U,V component values from the HSCM output file. The program is set up to read the Port Bolivar results from the GBM, the Morgans Point results from the GBM, then read the Morgans Point results from the HSCM, and the Port Bolivar data from the HSCM.

The program will read the first 24 hours of forecast data. The 200 loop begins on line 381, where nhr ranges from 1 through 24. As with the nowcast data, U and V component values are read from the GBM output file and from the HSCM output file. A call is made to subroutine calc_uvnew to convert the U,V components (curvilinear) to U,V with respect to the X-East,Y-North system.

At the end of the 500 loop, the GBM 00z file and the HSCM 00z file are closed for that day.

```
1       c Program Name : read_NF.curr.f
2       c
3       c Purpoes : Read model output of u and v components of current
4       c           velocity from both the GBM and HSC forecast output.
5       c           Reformat into standard CFS format.
6       c
7       c Language : FORTRAN
8       c
9       c Subroutines Called : calc_uvnew
10      c
11      c Author : Phil Richardson
12      c
13      c Version Date : October 30, 2000
14
15      ******************************************************
16
17            parameter(numdays=31,nstat=4)
18            parameter(im=181,jm=101,kb=6,kbm1=kb-1)
19            parameter(imm=71,jmm=211)
20            parameter(nstatarr=34)
21
22            character*52 filegbm,filehsc
23            character*56 filenm_gbm(numdays),filenm_hsc(numdays)
24            character*60 line
25            character*10 filenow(nstat),fileforc(nstat)
26
27            dimension lunnow(nstat),lunforc(nstat)
28            dimension z(kb),zz(kb),dz(kb),dzz(kb)
29            dimension hz(kb),hzz(kb),hdz(kb),hdzz(kb)
30            dimension dx(im,jm),dy(im,jm),art(im,jm),aru(im,jm),
31           *          arv(im,jm),cor(im,jm),h(im,jm),fsm(im,jm),
32           *          dum(im,jm),dvm(im,jm)
33            dimension ang_gbm(im,jm),alon(im,jm),alat(im,jm)
34            dimension ang_hsc(imm,jmm),halon(imm,jmm),halat(imm,jmm)
35            dimension ndaymiss_gbm(numdays),ndaymiss_hsc(numdays)
36            dimension u3gbm(nstatarr),v3gbm(nstatarr),
37           *          u3hsc(nstatarr),v3hsc(nstatarr)
38            dimension beta_gbm(2),beta_hsc(2)
39
40      ******************************************************
41
42      c Read from control file :
43      c
44      c     idebug - debug switch;
45      c     idebug = 1, write raw data (time, U and V components)
46      c   lunnow() - logical unit number for nowcast files
47      c  filenow() - filenames for nowcast files
48      c  lunforc() - logical unit number for forecast files
49      c fileforc() - filenames for forecast files
50      c     mlevel - model level
51      c  nfiledays - number of daily 00z files to be opened/read
52      c filenm_gbm - GBM model current data
53      c filenm_hsc - HSC model current data
54
55
56            read(5,*)istrtday
```

Program Listing 2.4. Read_NF.curr.f

```
57            read(5,*)idebug
58
59            nstations = 4
60            do ns=1,nstations
61              read(5,*)lunnow(ns)
62              read(5,32)filenow(ns)
63              read(5,*)lunforc(ns)
64              read(5,32)fileforc(ns)
65            enddo
66
67            read(5,*)mlevel
68
69            read(5,*)nfiledays
70            read(5,*)nmiss_gbm
71            do nm=1,nmiss_gbm
72               read(5,*)ndaymiss_gbm(nm)
73               write(6,*)ndaymiss_gbm(nm)
74            enddo
75            read(5,*)nmiss_hsc
76            do nm=1,nmiss_hsc
77               read(5,*)ndaymiss_hsc(nm)
78               write(6,*)ndaymiss_hsc(nm)
79            enddo
80
81
82            nmgbm = 1
83            nmhsc = 1
84
85            do nf=1,nfiledays
86               if(nf.ne.ndaymiss_gbm(nmgbm))then
87                  read(5,31)filenm_gbm(nf)
88                  write(6,31)filenm_gbm(nf)
89               else
90                  nmgbm = nmgbm + 1
91               endif
92               if(nf.ne.ndaymiss_hsc(nmhsc))then
93                  read(5,31)filenm_hsc(nf)
94                  write(6,31)filenm_hsc(nf)
95               else
96                  nmhsc = nmhsc + 1
97               endif
98            enddo
99
100
101     c Initialization
102
103           lungbm = 7
104           lunhsc = 8
105
106           small = .001
107           valnull = 99.99
108
109
110        31 format(a56)
111        32 format(a10)
112
```

Program Listing 2.4. Read_NF.curr.f (continued)

```
113     ******************************************************
114
115     c Open GBM model grid data file, read grid data
116
117     c     Variables :
118     c     fsm - land/water indicator; fsm = 1.0, water
119     c                                 fsm = 0.0, land
120
121           pi = 3.141593
122           rad = pi/180.0
123
124           filegbm = '/usr/people/philr/galves/model_grid/grid.06M'
125           lungrd_gbm = 17
126           open(lungrd_gbm,file=filegbm,access='sequential',
127          *     form='unformatted',status='old')
128
129           read(17)z,zz,dz,dzz,alon,alat,dx,dy,art,aru,arv,cor
130           read(17)h,fsm,dum,dvm
131
132
133     c Calculate angle of curvigrid
134
135           do 50 i=1,im-1
136              do 55 j=1,jm
137                 dlon = (alon(i+1,j) - alon(i,j)) * cos(alat(i,j)*rad)
138                 dlat = alat(i+1,j) - alat(i,j)
139                 dlnt = (dlon**2 + dlat**2)**0.5
140                 ang_gbm(i,j) = asin(dlat/dlnt)
141        55    continue
142        50 continue
143           do j=1,jm
144              ang_gbm(im,j) = ang_gbm(im-1,j)
145           enddo
146
147     !-----------------------------------------------------
148
149     c Open HSC model grid data file, read grid data
150
151           filehsc =
152          *   '/usr/people/philr/galves/model_grid/grid.cn.ce.1.bin'
153           lungrd_hsc = 18
154           open(lungrd_hsc,file=filehsc,form='unformatted',
155          *     status='old')
156
157           read(18)hz,hzz,hdz,hdzz,halon,halat,hdx,hdy,hart,haru,
158          *        harv,hcor
159           read(18)hh,hfsm,hdum,hdvm
160
161
162     c Calculate angle of curvigrid
163
164           do 60 i=1,imm-1
165              do 65 j=1,jmm
166                 dlon = (halon(i+1,j) - halon(i,j)) * cos(halat(i,j)*rad)
167                 dlat = halat(i+1,j) - halat(i,j)
168                 dlnt = (dlon**2 + dlat**2)**0.5
```

Program Listing 2.4. Read_NF.curr.f (continued)

```
169                ang_hsc(i,j) = asin(dlat/dlnt)
170           65   continue
171           60 continue
172              do j=1,jmm
173                 ang_hsc(imm,j) = ang_hsc(imm-1,j)
174              enddo
175
176
177        c Assign value of ang(i,j) to variable beta in order to
178        c convert U and V components from curvilinear to U and V
179        c with respect to the X,Y axis.
180
181              beta_gbm(1) = ang_gbm(86,34)
182              beta_gbm(2) = ang_gbm(80,76)
183              write(6,198)beta_gbm(1)
184              write(6,198)beta_gbm(2)
185
186              beta_hsc(1) = ang_hsc(33,151)
187              beta_hsc(2) = ang_hsc(44,20)
188              write(6,199)beta_hsc(1)
189              write(6,199)beta_hsc(2)
190
191        ********************************************************
192
193        c Open nowcast and forecast output files
194
195              do ns=1,nstations
196                 open(lunnow(ns),file=filenow(ns),form='formatted')
197                 write(lunnow(ns),41)mlevel ! write model level
198
199                 open(lunforc(ns),file=fileforc(ns),form='formatted')
200                 write(lunforc(ns),41)mlevel ! write model level
201              enddo
202
203
204           41 format('Model Level',i2)
205
206        ********************************************************
207
208        c Read header information from GBM and HSC files
209
210
211              nmgbm = 1
212              nmhsc = 1
213
214              rday = float(istrtday) - 2.0 + 0.7083
215
216              do 500 nf=1,nfiledays
217                 rtime = rday + float(nf-1)
218                 if(nf.ne.ndaymiss_gbm(nmgbm))then
219                    open(lungbm,file=filenm_gbm(nf),form='formatted',
220             *           status='old')
221
222                    read(lungbm,20)line
223                    write(6,19)line
224                    read(lungbm,*)iyear
```

Program Listing 2.4. Read_NF.curr.f (continued)

```
225               write(6,*)iyear
226
227               read(lungbm,21)nsta
228               read(lungbm,22)l1,l2
229               write(6,*)l1,l2
230               do l=1,2
231                  read(lungbm,20)line
232                  write(6,*)line
233               enddo
234            endif
235            if(nf.ne.ndaymiss_hsc(nmhsc))then
236               open(lunhsc,file=filenm_hsc(nf),form='formatted',
237         *          status='old')
238
239               read(lunhsc,20)line
240               write(6,*)line
241               read(lunhsc,*)iyear
242               write(6,*)iyear
243               read(lunhsc,21)nsta
244               read(lunhsc,20)line
245               read(lunhsc,25)l1,l2
246               write(6,*)l1,l2
247               do l=1,2
248                  read(lunhsc,20)line
249                  write(6,*)line
250               enddo
251            endif
252
253
254
255         19 format(/,a60)
256         20 format(a60)
257         21 format(3x,i2,2x,i3,2x,i3)
258         22 format(35x,2i5)
259
260      !-------------------------------------------------------
261
262      c Read first 24 hours, nowcast data.  Call subroutine
263      c calc_uvnew to convert U,V components with respect to
264      c curvilinear grid to U,V with respect to X,Y plane.
265
266
267               do 100 nhr=1,24 ! Loop thru hours 1 - 24.
268               rtimen11 = rtime + float(nhr)/24.0
269               if(nf.ne.ndaymiss_gbm(nmgbm))then
270                  read(lungbm,*,end=125)rtimeGBM
271
272                  do ns=1,24
273                     if(mlevel.eq.3)then
274                        read(lungbm,23)u3gbm(ns)
275                     endif
276                     if(mlevel.eq.1)then
277                        read(lungbm,26)u3gbm(ns)
278                     endif
279                  enddo
280
```

Program Listing 2.4. Read_NF.curr.f (continued)

```
281                do ns=1,24
282                   if(mlevel.eq.3)then
283                      read(lungbm,23)v3gbm(ns)
284                   endif
285                   if(mlevel.eq.1)then
286                      read(lungbm,26)v3gbm(ns)
287                   endif
288                enddo
289
290
291                betagbm = beta_gbm(1)
292                call calc_uvnew(betagbm,u3gbm(12),v3gbm(12),u3new,v3new)
293
294                write(lunnow(1),24)rtimeGBM,u3new,v3new
295
296
297                betagbm = beta_gbm(2)
298                call calc_uvnew(betagbm,u3gbm(19),v3gbm(19),u3new,v3new)
299
300                write(lunnow(2),24)rtimeGBM,u3new,v3new
301
302
303                do nt=1,9
304                   read(lungbm,*)rtimesp
305                   do l=1,48
306                      read(lungbm,20)line
307                   enddo
308                enddo
309
310             else
311                write(lunnow(1),24)rtimenll,valnull,valnull
312                write(lunnow(2),24)rtimenll,valnull,valnull
313             endif
314
315
316
317             if(nf.ne.ndaymiss_hsc(nmhsc))then
318                read(lunhsc,*,end=125)rtimeHSC
319
320                do ns=1,34
321                   if(mlevel.eq.3)read(lunhsc,23)u3hsc(ns)
322                   if(mlevel.eq.1)read(lunhsc,26)u3hsc(ns)
323                enddo
324
325                do ns=1,34
326                   if(mlevel.eq.3)read(lunhsc,23)v3hsc(ns)
327                   if(mlevel.eq.1)read(lunhsc,26)v3hsc(ns)
328                enddo
329
330
331                betahsc = beta_hsc(1)
332                call calc_uvnew(betahsc,u3hsc(19),v3hsc(19),u3new,v3new)
333
334                unewhsc = u3new
335                vnewhsc = v3new
336
```

Program Listing 2.4. Read_NF.curr.f (continued)

```
337                write(lunnow(3),24)rtimeHSC,unewhsc,vnewhsc
338
339
340                betahsc = beta_hsc(2)
341                call calc_uvnew(betahsc,u3hsc(3),v3hsc(3),u3new,v3new)
342
343                unewhsc = u3new
344                vnewhsc = v3new
345
346             write(lunnow(4),24)rtimeHSC,unewhsc,vnewhsc
347
348
349             do nt=1,9
350                read(lunhsc,*)rtimesp
351                do l=1,68
352                   read(lunhsc,20)line
353                enddo
354             enddo
355             else
356                write(lunnow(3),24)rtimenll,valnull,valnull
357                write(lunnow(4),24)rtimenll,valnull,valnull
358             endif
359
360     c       timediff = rtimeGBM - rtimeHSC
361     c       if(timediff.gt.small)then
362     c          write(6,101)
363     c          stop
364     c       endif
365
366
367       100 continue
368
369       125 continue
370
371
372       101 format('Program stopped, times not in agreement')
373
374     !-------------------------------------------------
375
376     c Read 2nd 24 hours, forecast data
377
378
379          rtime = rtime + 1.0
380
381          do 200 nhr=1,24 ! Loop thru hours 1 - 24.
382             rtimenll = rtime + float(nhr)/24.0
383             if(nf.ne.ndaymiss_gbm(nmgbm))then
384                read(lungbm,*,end=125)rtimeGBM
385
386                do ns=1,24
387                   if(mlevel.eq.3)then
388                      read(lungbm,23)u3gbm(ns)
389                   endif
390                   if(mlevel.eq.1)read(lungbm,26)u3gbm(ns)
391                enddo
392
```

Program Listing 2.4. Read_NF.curr.f (continued)

```
393
394               do ns=1,24
395                  if(mlevel.eq.3)read(lungbm,23)v3gbm(ns)
396                  if(mlevel.eq.1)read(lungbm,26)v3gbm(ns)
397               enddo
398
399
400               if(idebug.eq.1)write(6,201)rtimeGBM,u3gbm(12),
401           *                      v3gbm(12)
402
403               betagbm = beta_gbm(1)
404               call calc_uvnew(betagbm,u3gbm(12),v3gbm(12),u3new,v3new)
405
406               write(lunforc(1),24)rtimeGBM,u3new,v3new
407
408
409               betagbm = beta_gbm(2)
410               call calc_uvnew(betagbm,u3gbm(19),v3gbm(19),u3new,v3new)
411
412               write(lunforc(2),24)rtimeGBM,u3new,v3new
413
414
415               do nt=1,9
416                  read(lungbm,*)rtimesp
417                  do l=1,48
418                     read(lungbm,20)line
419                  enddo
420               enddo
421
422            else
423               write(lunforc(1),24)rtimenll,valnull,valnull
424               write(lunforc(2),24)rtimenll,valnull,valnull
425               if(nhr.eq.24)then
426                  nmgbm = nmgbm + 1
427               endif
428            endif
429
430
431            if(nf.ne.ndaymiss_hsc(nmhsc))then
432               read(lunhsc,*,end=125)rtimeHSC
433
434               do ns=1,34
435                  if(mlevel.eq.3)then
436                  read(lunhsc,23)u3hsc(ns)
437                  endif
438                  if(mlevel.eq.1)read(lunhsc,26)u3hsc(ns)
439               enddo
440
441
442               do ns=1,34
443                  if(mlevel.eq.3)read(lunhsc,23)v3hsc(ns)
444                  if(mlevel.eq.1)read(lunhsc,26)v3hsc(ns)
445               enddo
446
447
448               if(idebug.eq.1)write(6,202)rtimeHSC,u3hsc(19),
```

Program Listing 2.4. Read_NF.curr.f (continued)

```
449         *                       v3hsc(19)
450
451               betahsc = beta_hsc(1)
452               call calc_uvnew(betahsc,u3hsc(19),v3hsc(19),u3new,v3new)
453
454               write(lunforc(3),24)rtimeHSC,u3new,v3new
455
456
457               betahsc = beta_hsc(2)
458               call calc_uvnew(betahsc,u3hsc(3),v3hsc(3),u3new,v3new)
459
460               write(lunforc(4),24)rtimeHSC,u3new,v3new
461
462
463               do nt=1,9
464                  read(lunhsc,*)rtimesp
465                  do l=1,68
466                     read(lunhsc,20)line
467                  enddo
468               enddo
469
470            else
471               write(lunforc(3),24)rtimenll,valnull,valnull
472               write(lunforc(4),24)rtimenll,valnull,valnull
473               if(nhr.eq.24)then
474                  nmhsc = nmhsc + 1
475               endif
476            endif
477
478       200 continue
479
480           close (lungbm)
481           close (lunhsc)
482
483       500 continue
484
485
486       198 format(/,'angle of curvigrid angle (GBM) = ',f8.5,
487          *         ' radians')
488       199 format('angle of curvigrid angle (HSC) = ',f8.5,
489          *       ' radians',/)
490       201 format(1x,3f9.4,'  (GBM)')
491       202 format(1x,3f9.4,'  (HSC)')
492
493     ******************************************************
494
495        23 format(20x,f10.4)
496        24 format(1x,3f9.4)
497        25 format(25x,2i5)
498        26 format(f10.4)
499
500     ******************************************************
501
502           stop
503           end
```

Program Listing 2.4. Read_NF.curr.f (continued)

```
1             subroutine calc_uvnew(beta,u3,v3,u3new,v3new)
2
3       c     Purpose : To convert U,V components with respect to
4       c     curvilinear grid to U,V components with respect to
5       c     X,Y plane.
6       c
7       c     Version Date : Dec 20, 2000
8
9       ******************************************************
10
11            u3new = u3 * cos(beta) - v3 * sin(beta)
12
13            v3new = u3 * sin(beta) + v3 * cos(beta)
14
15
16            return
17            end
```

Program Listing 2.4. Read_NF.curr.f (continued)

## 2.5. Program Match.evnt_crnt.f

The listing for Program Match.evnt_crnt.f is given in Program Listing 2.5. After the parameter and dimension statements and after the character variables are declared, match.evnt_crnt.f will read necessary information from the control file. Variables read from the control file include idebug, monyr, headr1, headr2, endjd. Idebug can be set from 0 to 5 for various debug options. Monyr is the month and year for output files. Headr1 and headr2 are the headers for output files. Endjd is the end time of the analysis in Julian days (days elapsed from the beginning of the year). The next variable read is option. Option designates whether the type of data to be compared with the observed is nowcast, forecast, or astronomic tidal current along the principal component. Next is nsta, the number of stations included in the analysis. For each station, a station name is read, a start time is read in Julian days (see above), then the observed (PORTS) and model filenames are read. The next variables read are crlevel and prdir. Crlevel designates the lower (ebb) and upper (flood) critical current speed values and can be different for each station. Prdir designates the principal component direction by station. For the forecast adjustment option, the gain and bias are read separately for flood and ebb.

The 100 loop is the station loop, where ns is equal to 1 through nsta (nsta being the number of stations). First, the observed and model current data files are opened for each station and both files are positioned to the correct start time.

The model data and the observed data are read simultaneously. If either the model data point, or the observed data point, are greater than the specified critical value for flood, or less than the specified critical value for ebb, then nc2sig is incremented and the current speed values, model and observed, are stored in speedm and speedo, respectively. The corresponding times are stored in time_m and time_o, although the model time and the observed time should be very close to equal. Each point is flagged as to whether it is part of a flood event or an ebb event.

The 190 loop, which begins on line 426, loops through all extreme points, from 1 through nc2sig. All extreme points are grouped into events. These events are identified as being either flood events or ebb events. The number of points in the event, nevent, is incremented. Iflg is set to 1 for a flood event, or set to 0 for an ebb event. For a flood event, nevent_flood is incremented. For an ebb event, nevent_ebb is incremented.

The 200 loop, which begins on line 519, loops through events from 1 through nevent, where nevent is the number of events for that station. The 210 loop, loops through the points of each event, from 1 through nh(ns,np), where ns is the station number and np is the event number. The 210 loop determines the peak speed of each flood and ebb event. Also, the initial extreme point of each event is identified.

Finally, the 440 loop writes the output to the monthly table file, table2.out. Match.event_crnt.f does not produce a table.out file as does match.event.f, the version used for water levels. Table2.out includes information on each forecast of an event. The events are grouped by the outcome of the

forecast: success, failure, or false alarm. The information for each "success" includes the success number, the event number, the difference in start time between model and observed, and the difference in the duration of the event between model and observed. For failure and false alarm, the difference in start time and event duration are not applicable. Dpeak time is the time difference of the model peak current speed and the observed peak current speed in hours. Also included are the model peak current speed, the observed peak current speed, and the time (Julian date) of the observed peak. Dspeed is the absolute value of the difference between the model and the observed peak current speed in cm/s. The mean dspeed for each outcome (success, failure, false alarm) is calculated and presented in the table. Also included, in table2, is a summary, by station, of the total number of successes, failures, and false alarms for that month. The program generates two other tables. Table_flood provides information on the flood events, while table_ebb provides information on the ebb events.

```
1       c     Program Name : Match.evnt_crnt.f
2       c
3       c     Author : Phil Richardson
4       c
5       c     Version Date : Nov 14, 2000
6       c
7       c     Purpose : To search for current values > crlevel,
8       c     where values are current speed with respect to a
9       c     specified direction (by station), and where
10      c     crlevel is a critical value read in.  The program
11      c     will read through model and observed data simultan-
12      c     eously.  Match.event.f was originally written
13      c     to assist Eddie Shih with his COFS vs. TDL water
14      c     level analysis.  Later, it was revised to evaluate
15      c     Houston/Galveston forecast and nowcast water level
16      c     data.  In particular, this program will evaluate the
17      c     system with regard to current speed event situations.
18      c     Output includes table2 with success, failure, and
19      c     false alarm statistics by station for each month.
20      c
21      c     Revision : Dec 8, 2000
22      c                Creates plot data files for both forecast
23      c                and PORTS; plots of current speed along
24      c                principal direction.
25      c
26      c     Subroutines called : conctj, timehi, jdgreg, prDirection
27      c
28      c     Location : OPSEA -
29      c                /usr/people/philr/galves/NF_eval/currents/sa.current/
30
31      ******************************************************
32
33            parameter(npts=745,npeaks=80,nstat=6)
34
35            character*1  formfd
36            character*3  plt_option
37            character*5  istation_id
38            character*8  option
39            character*10 filetab2,file23
40            character*11 file22
41            character*14 stanam(nstat),monyr
42            character*17 fileplotM(nstat),fileplotO(nstat)
43            character*50 filegrid
44            character*60 headr1,file2sig
45            character*25 headr2
46            character*71 fileports(nstat),filesumm
47            character*72 filemod(nstat)
48
49            dimension nh(nstat,npeaks),npt_obspk(nstat,npeaks),
50           *          npt_modpk(nstat,npeaks),iflg(npeaks),
51           *          iflag(npts),iflag_sta(nstat,npeaks),
52           *          i_flag(3,npeaks)
53            dimension nevent_sta(nstat),crlevel(nstat),
54           *          gain_flood(nstat),gain_ebb(nstat),
55           *          bias_flood(nstat),bias_ebb(nstat)
56            dimension time_o(npts),time_m(npts),speedo(npts),
```

Program Listing 2.5. Match.evnt_crnt.f

```
57           *           speedm(npts)
58            dimension time_peako(nstat,npeaks,npts),
59           *           time_peakm(nstat,npeaks,npts),
60           *           value_evnto(nstat,npeaks,npts),
61           *           value_evntm(nstat,npeaks,npts)
62            dimension tm_pkval(nstat,2,npeaks),
63           *           tm_pkvalWr(nstat,2,npeaks),
64           *           wl_pkval(nstat,2,npeaks),
65           *           wl_pkvalWr(nstat,2,npeaks),
66           *           cald_pkval(nstat,2,npeaks)
67            dimension tm_1stval(nstat,2,npeaks)
68            dimension idel_strtm(3,npeaks),idel_peaktm(3,npeaks),
69           *           ievent(3,npeaks),idel_duration(npeaks),
70           *           obs_wl(3,npeaks),delta_wl(3,npeaks),
71           *           forc_wl(3,npeaks),tm_obspeak(3,npeaks)
72            dimension startjd(nstat),mlevel(nstat)
73
74            common/headrs/headr1,headr2,monyr,option,startjd,
75           *              endjd,crlevel
76            common/debug/idebug
77            common/prindir/prdir(nstat)
78
79      ******************************************************
80
81      c  Read from input :
82      c
83      c     idebug - debug switch
84      c     idebug = 1, write input files to 6
85      c     idebug = 2, events
86      c            = 3, points above critical current speed value
87      c            = 4, check for tdiff (time difference)
88      c            = 5, calculations of current speed with respect
89      c                 to principal direction
90      c       monyr - month and year
91      c       headr - header for output files
92      c    startjd() - Julian start time by station
93      c       endjd - ending time
94      c      option - forecast, nowcast, or prediction
95      c        nsta - number of stations
96      c     stanam() - station name
97      c fileports() - file containing observed water level data
98      c    filemod() - file containing model water level data
99      c    crlevel() - critical value for current speed by station
100     c      prdir() - principal direction (by station)
101     c  gain_flood - gain applied to flood values
102     c  gain_ebb   - gain applied to ebb values
103     c  plt_option - option to write plot data (time and current
104     c               speed with respect to principal direction)
105     c  fileplotM  - time and speed data for plot file (model)
106     c  fileplotO  - time and speed data for plot file (obs)
107
108           read(5,*)idebug
109           read(5,1039)monyr
110           read(5,31)headr1,headr2
111           read(5,*)endjd
112           write(6,33)endjd
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
113
114           read(5,'(a8)')option
115
116           read(5,*)nsta
117           do ls=1,nsta
118              read(5,1039)stanam(ls)
119              if(idebug.eq.1)write(6,1040)stanam(ls)
120              read(5,*)startjd(ls)
121              read(5,34)fileports(ls)
122              if(idebug.eq.1)write(6,34)fileports(ls)
123              read(5,35)filemod(ls)
124              if(idebug.eq.1)write(6,35)filemod(ls)
125              read(5,*)crlevel(ls)
126              read(5,*)prdir(ls)
127              if(option.eq.'forecast')then
128                 read(5,*)gain_flood(ls),gain_ebb(ls)
129                 read(5,*)bias_flood(ls),bias_ebb(ls)
130              endif
131           enddo
132
133           read(5,'(a3)')plt_option
134           if(plt_option.eq.'yes')then
135              do ls=1,nsta
136                 read(5,39)fileplotM(ls)
137                 read(5,39)fileplotO(ls)
138              enddo
139           endif
140
141           dayspl = endjd - startjd(1)
142     c     initialization
143           formfd = CHAR(12)
144           small = 0.01
145           tlhour = 0.0417
146
147
148     c     Set time check interval for peak values
149
150           nh_ch = 1
151           peak_intervl = float(nh_ch) * tlhour + .002
152           write(6,36)peak_intrvl
153           big_value = 99999.9999
154           write(6,37)
155
156
157           lunplotF = 9
158           lunplotP = 10
159
160
161        31 format(a60,/,a25)
162        33 format('Julian stop time from control file ',f8.2,/)
163        34 format(a71)
164        35 format(a72)
165        36 format(/,'Peak time interval = ',f8.4)
166        37 format(//)
167        38 format(a8)
168        39 format(a17)
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
169
170     ******************************************************
171
172     c  Open output files
173
174     c  Filenames :
175     c
176     c   filetab2 - table of event information;
177     c              includes success, failure, false alarm
178     c              summary
179
180
181           filetab2 = 'table2.out'
182           file22 = 'table_flood'
183           file23 = 'table_ebb'
184
185
186           open(21,file=filetab2,form='formatted')
187           open(22,file=file22,form='formatted')
188           open(23,file=file23,form='formatted')
189
190
191        42 format(//,' station       crlevel')
192
193     ******************************************************
194
195     c     Begin station loop (100).  Open observed and
196     c     model water level files.
197
198
199           lunobs = 7
200           lunmod = 8
201
202           do 100 ns=1,nsta
203              write(6,1040)stanam(ns)
204              open(lunobs,file=fileports(ns),form='formatted',
205          *        status='old')
206              open(lunmod,file=filemod(ns),form='formatted',
207          *        status='old')
208
209              read(lunmod,101)mlevel(ns)
210
211
212              if(plt_option.eq.'yes')then
213                 open(lunplotF,file=fileplotM(ns),form='formatted')
214                 open(lunplotP,file=fileplotO(ns),form='formatted')
215              endif
216
217
218       101 format(11x,i2)
219
220     !-------------------------------------------------------
221
222     c        Position obs file to start time
223
224       610    continue
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
225             read(lunobs,*,end=615)t,u,v
226             if(t.lt.startjd(ns))goto 610
227       615   continue
228             write(6,121)t
229             backspace lunobs
230
231
232       c     Position mod file to start time
233
234       620   continue
235
236             if(option.eq.'forecast'.or.option.eq.'nowcast*')then
237                read(lunmod,*,end=625)t,u,v
238             endif
239             if(option.eq.'astronom')then
240                read(lunmod,125,end=625)istation_id,iyear,imonth,
241          *          iday,ihr,imin
242                write(6,125)istation_id,iyear,imonth,iday,ihr,imin
243                call conctj(jday,imonth,iday,iyear)
244                rday = float(jday) + float(ihr)/24.0
245                t = rday
246             endif
247
248             if(t.lt.startjd(ns))goto 620
249       625   continue
250             write(6,122)t
251             backspace lunmod
252
253
254       121 format('Start time observed (PORTS) current data ',
255          *        f7.2)
256       122 format('Start time model current data',f10.2)
257
258       !-------------------------------------------------------
259
260       c Read forecast U,V values and PORTS U,V values.  Values
261       c are read in units of m/s (Subroutine prDirection later
262       c converts values to cm/s).  Then determine which points
263       c are above crlevel.
264       c
265       c       Variables :
266       c         crlevel - current speed critical value
267       c          nc2sig - counter for number of values above
268       c                   or below specified critical value
269       c           tdiff - time check, time difference between
270       c                   observed and model
271       c          speedo - observed water level (extreme)
272       c          time_o - time which corresponds to the observed
273       c                   extreme value
274       c          speedm - model water level (extreme)
275       c          time_m - time which corresponds to the model
276       c                   extreme value
277       c         iflag() - identifies each point as to flood/ebb
278
279
280
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
281             nc2sig = 0
282             tm_old = 0.0
283
284             write(6,1001)
285
286
287       175   continue
288             if(option.eq.'forecast'.or.option.eq.'nowcast*')then
289                read(lunmod,*,end=180)t_mod,umod,vmod
290                if(umod.lt.99.90)then
291                   if(idebug.eq.5)then
292                      write(6,132)t_mod,umod,vmod
293                   endif
294                   call prDirection(ns,umod,vmod,value_mod)
295                   if(idebug.eq.5)then
296                      write(6,134)t_mod,value_mod
297                   endif
298                   if(option.eq.'forecast')then
299                      if(value_mod.gt.0.0)then
300                         value_mod = value_mod * gain_flood(ns) +
301          *                          bias_flood(ns)
302                      endif
303                      if(value_mod.lt.0.0)then
304                         value_mod = value_mod * gain_ebb(ns) +
305          *                          bias_ebb(ns)
306                      endif
307                   endif
308                   if(plt_option.eq.'yes')then
309                      write(lunplotF,138)t_mod,value_mod
310                   endif
311                endif
312             endif
313             if(option.eq.'astronom')then
314                read(lunmod,125,end=180)istation_id,iyear,imonth,
315          *          iday,ihr,imin,value_mod
316                if(idebug.eq.5)then
317                write(6,*)iyear,imonth,iday,ihr,imin,value_mod
318                endif
319                call conctj(jday,imonth,iday,iyear)
320                rday = float(jday) + float(ihr)/24.0
321                t_mod = rday
322                value_mod = value_mod * 100.0
323                if(plt_option.eq.'yes')then
324                   write(lunplotF,138)t_mod,value_mod
325                endif
326             endif
327
328
329       176   read(lunobs,*,end=180)t_obs,uobs,vobs
330
331
332             if(idebug.eq.5)then
333                write(6,133)t_obs,uobs,vobs
334             endif
335             if(uobs.lt.99.90)then
336                call prDirection(ns,uobs,vobs,value_obs)
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
337               if(idebug.eq.5)then
338                  write(6,135)t_obs,value_obs
339               endif
340               if(plt_option.eq.'yes')then
341                  write(lunplotP,138)t_obs,value_obs
342               endif
343            endif
344
345
346            if(umod.gt.99.90.or.uobs.gt.99.90)then
347               goto 175
348            else
349               if(idebug.eq.5)then
350                  write(6,136)value_mod,value_obs
351               endif
352            endif
353
354            if(t_obs.gt.endjd)goto 185
355            tdiff = abs(t_obs-t_mod)
356
357            if(tdiff.gt.small)then
358               write(6,131)t_obs,t_mod,stanam(ns)
359               stop
360            endif
361
362          if(value_obs.ge.crlevel(ns).or.
363         *   value_obs.le.-crlevel(ns).or.
364         *   value_mod.ge.crlevel(ns).or.
365         *   value_mod.le.-crlevel(ns))then
366            nc2sig = nc2sig + 1
367            if(value_obs.ge.crlevel(ns).or.value_mod.ge.
368         *      crlevel(ns))then
369               iflag(nc2sig) = 1
370            endif
371            if(value_obs.le.-crlevel(ns).or.value_mod.le.
372         *      -crlevel(ns))then
373               iflag(nc2sig) = 0
374            endif
375            time_o(nc2sig) = t_obs
376            speedo(nc2sig) = value_obs
377            time_m(nc2sig) = t_mod
378            speedm(nc2sig) = value_mod
379          endif
380          goto 175
381       180 continue
382          write(6,137)t_mod,t_obs
383       185 continue
384
385
386
387       125 format(1x,a5,3x,i4,4i3,f10.3)
388       131 format(' Program stopped due to time discrepancy',
389         *       ' between obs ',f9.4,/,' and model ',f9.4,
390         *       ' for station ',a14)
391       132 format(/,1x,3f9.4)
392       133 format(1x,3f9.4)
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
      134 format(1x,f9.4,f9.3,' forecast')
      135 format(1x,f9.4,f9.3,' PORTS')
      136 format(1x,2f9.4)
      137 format('End of File reached',/,'forecast time =',
     *        f9.4,' observed time = ',f9.4)
      138 format(1x,f9.4,f10.4)

!-----------------------------------------------------

c     Loop through extreme points from 1 through nc2sig,
c     group extreme points together in events.  Identify
c     events as being either flood (positive) or ebb
c     (negative) with iflg.

c     Varialbles :
c         nevent - counter for number of events (by station)
c  nh(ns,nevent) - counts number of points in each event
c     time_peako - time which corresponds to observed extreme
c                  value grouped by station, event no., and
c                  point number of event
c    value_evnto - observed current speed value grouped by
c                  station, event no., and point number
c     time_peakm - time which corresponds to model extreme
c                  value grouped by station, event no., and
c                  point number of event
c    value_evntm - model current speed value grouped by station,
c                  event no., and point number


      nevent = 0
      nevent_flood = 0
      nevent_ebb = 0

      do 190 nt=1,nc2sig
c     Look at first 2sigma point
         if(nt.eq.1)then
            nevent = 1
            if(iflag(nt).eq.1)then
               iflg(nevent) = 1
               iflag_sta(ns,nevent) = 1
               if(idebug.eq.2)write(6,192)nevent
               nevent_flood = 1
            endif
            if(iflag(nt).eq.0)then
               iflg(nevent) = 0
               iflag_sta(ns,nevent) = 0
               if(idebug.eq.2)write(6,193)nevent
               nevent_ebb = 1
            endif
            nh(ns,nevent) = 1
         endif

c        Look at remainder of extreme points
         if(nt.gt.1)then
         tmdiff = time_o(nt) - tm_old
         if(tmdiff.gt.peak_intervl)then
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
c        if(tmdiff.gt.peak_intrvl.or.iflag(nt).ne.
c    *                iflag(nt-1))then
               nevent = nevent + 1
               if(iflag(nt).eq.1)then
                  iflg(nevent) = 1
                  iflag_sta(ns,nevent) = 1
                  if(idebug.eq.2)write(6,192)nevent
                  nevent_flood = nevent_flood + 1
               endif
               if(iflag(nt).eq.0)then
                  iflg(nevent) = 0
                  iflag_sta(ns,nevent) = 0
                  if(idebug.eq.2)write(6,193)nevent
                  nevent_ebb = nevent_ebb + 1
               endif
               nh(ns,nevent) = 1
            else
               nh(ns,nevent) = nh(ns,nevent) + 1
            endif
            endif

            time_peako(ns,nevent,nh(ns,nevent)) = time_o(nt)
            value_evnto(ns,nevent,nh(ns,nevent)) = speedo(nt)
            time_peakm(ns,nevent,nh(ns,nevent)) = time_m(nt)
            value_evntm(ns,nevent,nh(ns,nevent)) = speedm(nt)
            tm_old = time_o(nt)
  190 continue


            if(idebug.eq.3)then
               write(6,191)stanam(ns),nevent
            endif

            nevent_sta(ns) = nevent


  191 format(/,' for station ',a14,/,1x,i4,
     *         ' events; ')
  192 format(' Event',i3,' is a flood event')
  193 format(' Event',i3,' is an ebb event')

!-------------------------------------------------------

c     Loop (200) thru events (1 thru nevent) for each
c     station; iflg = 1 denotes flood, iflg = 0 denotes
c     ebb.  Loop (210) through points (1 thru nh(ns,np))
c     for each event to determine the extreme point of each
c     flood and ebb.  Also, determine initial extreme
c     point, model and observed, for flood and ebb.


      if(idebug.eq.3)write(6,199)

c     npt_obspk(ns,np) - number of points in observed event
c     npt_modpk(ns,np) - number of points in model event
c            time_chk - check for peak value occurring at
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
c                          start of next month
c   tm_1stval(ns,2,np) - time of 1st wl value during
c                          observed event
c   tm_1stval(ns,1,np) - time of 1st wl value during
c                          model event
c       currnt_higho   - peak observed water level value
c           tm_higho   - time of peak obs water level
c       currnt_lowo    - low observed water level value
c           tm_lowo    - time of low obs water level
c       currnt_highm   - peak model water level value
c           tm_highm   - time of peak model water level
c       currnt_lowm    - low model water level value
c           tm_lowm    - time of low model water level

        do 200 np=1,nevent
           npt_obspk(ns,np) = 0
           npt_modpk(ns,np) = 0
           if(idebug.eq.3)write(6,201)np
           currnt_higho = -999.99
           currnt_lowo = 999.99
           currnt_highm = -999.99
           currnt_lowm = 999.99
           do 210 n=1,nh(ns,np)
           if(idebug.eq.3)write(6,1006)time_peako(ns,np,n),
     *        value_evnto(ns,np,n),value_evntm(ns,np,n)
              if(iflg(np).eq.1)then
                 if(value_evnto(ns,np,n).ge.crlevel(ns))then
                    if(npt_obspk(ns,np).eq.0)then
                       tm_1stval(ns,2,np) = time_peako(ns,np,n)
                    endif
                    npt_obspk(ns,np) = npt_obspk(ns,np) + 1
                 endif
                 if(value_evnto(ns,np,n).gt.currnt_higho)then
                    currnt_higho = value_evnto(ns,np,n)
                    tm_higho = time_peako(ns,np,n)
                 endif
                 if(value_evntm(ns,np,n).ge.crlevel(ns))then
                 if(npt_modpk(ns,np).eq.0)then
                    tm_1stval(ns,1,np) = time_peakm(ns,np,n)
                 endif
                 npt_modpk(ns,np) = npt_modpk(ns,np) + 1
                 endif
                 if(value_evntm(ns,np,n).gt.currnt_highm)then
                 currnt_highm = value_evntm(ns,np,n)
                 tm_highm = time_peakm(ns,np,n)
                 endif
              endif
              if(iflg(np).eq.0)then
                 if(value_evnto(ns,np,n).le.-crlevel(ns))then
                    if(npt_obspk(ns,np).eq.0)then
                       tm_1stval(ns,2,np) = time_peako(ns,np,n)
                    endif
                    npt_obspk(ns,np) = npt_obspk(ns,np) + 1
                 endif
                 if(value_evnto(ns,np,n).lt.currnt_lowo)then
                    currnt_lowo = value_evnto(ns,np,n)
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
                     tm_lowo = time_peako(ns,np,n)
                  endif
                  if(value_evntm(ns,np,n).le.-crlevel(ns))then
                     if(npt_modpk(ns,np).eq.0)then
                        tm_1stval(ns,1,np) = time_peakm(ns,np,n)
                     endif
                     npt_modpk(ns,np) = npt_modpk(ns,np) + 1
                  endif
                  if(value_evntm(ns,np,n).lt.currnt_lowm)then
                     currnt_lowm = value_evntm(ns,np,n)
                     tm_lowm = time_peakm(ns,np,n)
                  endif
               endif
  210       continue


c        Variables :
c        wl_pkval(ns,1,np) - model peak water level value
c                            stored in array by event
c        tm_pkval(ns,1,np) - time of model peak value stored
c                            in array by event
c        wl_pkval(ns,2,np) - observed peak water level value
c                            stored in array by event
c        tm_pkval(ns,2,np) - time of observed peak value
c                            stored in array by event

            if(iflg(np).eq.1)then
               if(idebug.eq.2)write(6,202)currnt_higho,tm_higho
               call timehi(caldayo,tm_higho,idebug)
               time_chk = caldayo - 1.000
               if(abs(time_chk).lt.small.and.tm_higho.gt.
     *            startjd(ns)+5.0)then
                  caldayo = caldayo + dayspl
                  write(6,204)stanam(ns),np,tm_higho,caldayo
               endif
               if(idebug.eq.2)write(6,203)currnt_highm,tm_highm
               call timehi(caldaym,tm_highm,idebug)
               time_chk = caldaym - 1.000
               if(abs(time_chk).lt.small.and.tm_highm.gt.
     *                              startjd(ns)+5.0)then
                  caldaym = caldaym + dayspl
                  write(6,204)stanam(ns),np,tm_highm,caldaym
               endif
               if(currnt_highm.ge.crlevel(ns))then
                  tm_pkval(ns,1,np) = tm_highm
                  tm_pkvalWr(ns,1,np) = tm_highm
                  cald_pkval(ns,1,np) = caldaym
                  wl_pkval(ns,1,np) = currnt_highm
                  wl_pkvalWr(ns,1,np) = currnt_highm
               else
                  tm_1stval(ns,1,np) = big_value
                  tm_pkval(ns,1,np) = tm_highm
                  tm_pkvalWr(ns,1,np) = big_value
                  cald_pkval(ns,1,np) = big_value
                  wl_pkval(ns,1,np) = currnt_highm
                  wl_pkvalWr(ns,1,np) = big_value
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
617               endif
618               if(currnt_higho.ge.crlevel(ns))then
619                  tm_pkval(ns,2,np) = tm_higho
620                  tm_pkvalWr(ns,2,np) = tm_higho
621                  cald_pkval(ns,2,np) = caldayo
622                  wl_pkval(ns,2,np) = currnt_higho
623                  wl_pkvalWr(ns,2,np) = currnt_higho
624               else
625                  tm_1stval(ns,2,np) = big_value
626                  tm_pkval(ns,2,np) = tm_higho
627                  tm_pkvalWr(ns,2,np) = big_value
628                  cald_pkval(ns,2,np) = big_value
629                  wl_pkval(ns,2,np) = currnt_higho
630                  wl_pkvalWr(ns,2,np) = big_value
631               endif
632            endif
633            if(iflg(np).eq.0)then
634               if(idebug.eq.2)write(6,205)currnt_lowo,tm_lowo
635               call timehi(caldayo,tm_lowo,idebug)
636               time_chk = caldayo - 1.000
637               if(abs(time_chk).lt.small.and.tm_lowo.gt.
638         *                           startjd(ns)+5.0)then
639                  write(6,204)stanam(ns),np,tm_lowo,caldayo
640                  caldayo = caldayo + dayspl
641               endif
642               if(idebug.eq.2)write(6,206)currnt_lowm,tm_lowm
643               call timehi(caldaym,tm_lowm,idebug)
644               time_chk = caldaym - 1.000
645               if(abs(time_chk).lt.small.and.tm_lowm.gt.
646         *                           startjd(ns)+5.0)then
647                  write(6,204)stanam(ns),np,tm_lowm,caldaym
648               endif
649               if(currnt_lowm.le.-crlevel(ns))then
650                  tm_pkval(ns,1,np) = tm_lowm
651                  tm_pkvalWr(ns,1,np) = tm_lowm
652                  cald_pkval(ns,1,np) = caldaym
653                  wl_pkval(ns,1,np) = currnt_lowm
654                  wl_pkvalWr(ns,1,np) = currnt_lowm
655               else
656                  tm_1stval(ns,1,np) = big_value
657                  tm_pkval(ns,1,np) = tm_lowm
658                  tm_pkvalWr(ns,1,np) = big_value
659                  cald_pkval(ns,1,np) = big_value
660                  wl_pkval(ns,1,np) = currnt_lowm
661                  wl_pkvalWr(ns,1,np) = big_value
662               endif
663               if(currnt_lowo.le.-crlevel(ns))then
664                  tm_pkval(ns,2,np) = tm_lowo
665                  tm_pkvalWr(ns,2,np) = tm_lowo
666                  cald_pkval(ns,2,np) = caldayo
667                  wl_pkval(ns,2,np) = currnt_lowo
668                  wl_pkvalWr(ns,2,np) = currnt_lowo
669               else
670                  tm_1stval(ns,2,np) = big_value
671                  tm_pkval(ns,2,np) = tm_lowo
672                  tm_pkvalWr(ns,2,np) = big_value
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
673                  cald_pkval(ns,2,np) = big_value
674                  wl_pkval(ns,2,np) = currnt_lowo
675                  wl_pkvalWr(ns,2,np) = big_value
676               endif
677            endif
678      200 continue
679
680
681
682      199 format('    Time          obs       model')
683      201 format(' spike #',i3)
684      202 format(' Largest pos current vel (obs) is',f10.4,
685         *       ' occurring at',f10.4)
686      203 format(' Largest pos current vel (mod) is',f10.4,
687         *       ' occurring at',f10.4)
688      204 format(1x,a14,'Peak value event ',i3,2x,'occurs at',
689         *       2f9.3)
690      205 format(' Largest neg current vel (obs) is',f10.4,
691         *       ' occurring at',f10.4)
692      206 format(' Largest neg current vel (mod) is',f10.4,
693         *       ' occurring at',f10.4)
694
695   !-----------------------------------------------------
696
697            close (lunobs)
698            close (lunmod)
699
700            if(plt_option.eq.'yes')then
701               close (lunplotF)
702               close (lunplotP)
703            endif
704
705      100 continue
706
707
708   c End of station loop.
709
710   ******************************************************
711
712   c     Group events as to success, failure, false.
713
714   c     Variables :
715   c
716   c  tm_pkval(ns,1,np) - time of model extreme value
717   c                     stored in array by event
718   c  tm_pkval(ns,2,np) - time of observed extreme value
719   c                     stored in array by event
720   c
721   c        int_big - integer dummy variable for ihr_diff
722   c      big_value - 99999.9999 (?)
723   c       nsuccess - number of times model successfully
724   c                  predicts observed event.
725   c      nfailure  - number of times model fails to predict
726   c                  an observed event.
727   c         nfalse - number of times model predicts event
728   c                  is forecast but not observed
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
c nevent_sta(ns) - number of events for each station
c          dtime - time difference (real) between
c                  observed and model event peaks
c   idel_strtm() - time difference in hours between
c                  observed and model event start times
c       ihr_diff - time difference in hours between
c                  observed and model event peaks
c            dwl - difference in water level between
c                  observed and model event peaks


c     Initialization :
      int_big = 999999
      small = 0.01


c     Write header information to each output file

      write(21,420)headr1,headr2
      write(21,1040)monyr
      write(22,421)headr1
      write(22,1040)monyr
      write(23,422)headr1
      write(23,1040)monyr


  420 format(a60,/,a15)
  421 format(a60,/,'Event Analysis (Flood)')
  422 format(a60,/,'Event Analysis (Ebb)')

!-----------------------------------------------------

c     Loop 440 is the station loop.  450 loops through
c     events by station.


      do 440 ns=1,nsta

c        Initialization :
         nsuccess = 0
         nsuccess_flood = 0
         nsuccess_ebb = 0
         nfailure = 0
         nfail_flood = 0
         nfail_ebb = 0
         nfalse = 0
         nfalse_flood = 0
         nfalse_ebb = 0
         dwlS_total = 0.0
         dwlS_totalFl = 0.0
         dwlS_totalEbb = 0.0

         dwlFail_total = 0.0
         dwlFail_totFl = 0.0
         dwlFail_totEbb = 0.0
         dwlFalse_total = 0.0
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
785             dwlFalse_totFl = 0.0
786             dwlFalse_totEbb = 0.0
787
788     c       Write header information to tables 21, 22, 23
789             write(21,411)stanam(ns),mlevel(ns)
790             write(21,416)prdir(ns)
791             write(21,412)crlevel(ns)
792             if(option.eq.'forecast')then
793                write(21,413)gain_flood(ns),bias_flood(ns),
794          *                    gain_ebb(ns),bias_ebb(ns)
795             endif
796             write(21,419)startjd(ns)
797
798             write(22,411)stanam(ns)
799             write(22,412)crlevel(ns)
800             write(23,411)stanam(ns)
801             write(23,412)crlevel(ns)
802             if(option.eq.'forecast')then
803                write(22,414)gain_flood(ns),bias_flood(ns)
804                write(23,415)gain_ebb(ns),bias_ebb(ns)
805             endif
806             write(22,419)startjd(ns)
807             write(23,419)startjd(ns)
808
809
810             do 450 np=1,nevent_sta(ns)
811                dtime = tm_pkval(ns,1,np) - tm_pkval(ns,2,np)
812                hr_diff = dtime * 24.0
813                ihr_diff = nint(hr_diff)
814                del_strttm = (tm_1stval(ns,1,np)-
815          *                   tm_1stval(ns,2,np)) * 24.0
816
817     c     Case 1 : Success
818                if(tm_pkvalWr(ns,2,np).lt.90000.0.and.tm_pkvalWr
819          *        (ns,1,np).lt.90000.0)then
820                   nsuccess = nsuccess + 1
821                   dwlS = wl_pkval(ns,1,np) - wl_pkval(ns,2,np)
822                   ievent(1,nsuccess) = np
823                   idel_peaktm(1,nsuccess) = ihr_diff
824                   idel_strtm(1,nsuccess) = nint(del_strttm)
825                   id_duration = npt_modpk(ns,np) -
826          *                      npt_obspk(ns,np)
827                   idel_duration(nsuccess) = id_duration
828                   forc_wl(1,nsuccess) = wl_pkval(ns,1,np)
829                   obs_wl(1,nsuccess) = wl_pkval(ns,2,np)
830                   tm_obspeak(1,nsuccess) = tm_pkval(ns,2,np)
831                   delta_wl(1,nsuccess) = dwlS
832                   dwlS_total = dwlS_total + abs(dwlS)
833                   if(iflag_sta(ns,np).eq.1)then
834                      i_flag(1,nsuccess) = 1
835                      nsuccess_flood = nsuccess_flood + 1
836                      dwlS_totalFl = dwlS_totalFl + abs(dwlS)
837                   endif
838                   if(iflag_sta(ns,np).eq.0)then
839                      i_flag(1,nsuccess) = 0
840                      nsuccess_ebb = nsuccess_ebb + 1
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
841                     dwlS_totalEbb = dwlS_totalEbb + abs(dwlS)
842                  endif
843                  dwl = dwlS
844               endif
845
846     c     Case 2 : Failure
847               if(tm_pkvalWr(ns,1,np).gt.90000.0)then
848                  nfailure = nfailure + 1
849                  dwlFail = wl_pkval(ns,1,np) - wl_pkval(ns,2,np)
850                  ievent(2,nfailure) = np
851                  idel_peaktm(2,nfailure) = ihr_diff
852                  ihr_diff = int_big
853                  forc_wl(2,nfailure) = wl_pkval(ns,1,np)
854                  obs_wl(2,nfailure) = wl_pkval(ns,2,np)
855                  tm_obspeak(2,nfailure) = tm_pkval(ns,2,np)
856                  delta_wl(2,nfailure) = dwlFail
857                  dwlFail_total = dwlFail_total + abs(dwlFail)
858                  if(iflag_sta(ns,np).eq.1)then
859                     i_flag(2,nfailure) = 1
860                     nfail_flood = nfail_flood + 1
861                     dwlFail_totFl = dwlFail_totFl + abs(dwlFail)
862                  endif
863                  if(iflag_sta(ns,np).eq.0)then
864                     i_flag(2,nfailure) = 0
865                     nfail_ebb = nfail_ebb + 1
866                     dwlFail_totEbb = dwlFail_totEbb + abs(dwlFail)
867                  endif
868                  dwl = big_value
869               endif
870
871     c     Case 3 : false alarm
872               if(tm_pkvalWr(ns,2,np).gt.90000.0)then
873                  nfalse = nfalse + 1
874                  dwlFalse = wl_pkval(ns,1,np) - wl_pkval(ns,2,np)
875                  ievent(3,nfalse) = np
876                  idel_peaktm(3,nfalse) = ihr_diff
877                  ihr_diff = int_big
878                  forc_wl(3,nfalse) = wl_pkval(ns,1,np)
879                  obs_wl(3,nfalse) = wl_pkval(ns,2,np)
880                  tm_obspeak(3,nfalse) = tm_pkval(ns,2,np)
881                  delta_wl(3,nfalse) = dwlFalse
882                  dwlFalse_total = dwlFalse_total + abs(dwlFalse)
883                  if(iflag_sta(ns,np).eq.1)then
884                     i_flag(3,nfalse) = 1
885                     nfalse_flood = nfalse_flood + 1
886                     dwlFalse_totFl = dwlFalse_totFl + abs(dwlFalse)
887                  endif
888                  if(iflag_sta(ns,np).eq.0)then
889                     i_flag(3,nfalse) = 0
890                     nfalse_ebb = nfalse_ebb + 1
891                     dwlFalse_totEbb = dwlFalse_totEbb + abs(dwlFalse)
892                  endif
893                  dwl = big_value
894               endif
895
896
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
897       402 format(/,20x,'Model data',15x,'Observed data',
898          *       /,6x,'event#   jul day  cal day   wl',
899          *      4x,'jul day  cal day   wl  Dt(hour) Dwl(m)')
900       411 format(//,1x,a14,1x,'; Model Level',i2)
901       412 format(' critical value (current speed along principal',
902          *       ' direction)',/,' = ',f7.2,' cm/s')
903       413 format(' Flood : gain =',f7.2,', bias =',f7.2,
904          *     /,'   Ebb : gain =',f7.2,', bias =',f7.2)
905       414 format(' Flood : gain =',f7.2,', bias =',f7.2)
906       415 format(' Ebb : gain =',f7.2,', bias =',f7.2)
907       416 format(' Principal Direction :',f6.1,' degr')
908       419 format(' Julian start time from control file',
909          *         f8.2)
910
911     !-------------------------------------------------
912
913       450 continue
914
915     ********************************************************
916
917     c Write output to table2.out.
918     c Calculate mean water level differences for three cases.
919     c
920     c Variables :
921     c
922     c    dwlS_mean - mean water level difference (success)
923     c    dwlFail_mean - mean water level difference (failure)
924     c    dwlFalse_mean - mean water level difference (false)
925
926
927       437 format('High water critical level,',f10.3)
928
929     !-------------------------------------------------
930
931     c    Case 1 : Success
932
933
934             if(nsuccess.gt.0)write(21,431)
935             if(nsuccess_flood.gt.0)write(22,431)
936             if(nsuccess_ebb.gt.0)write(23,431)
937
938             do 455 n=1,nsuccess
939                write(21,447)n,ievent(1,n),idel_strtm(1,n),
940          *           idel_duration(n),idel_peaktm(1,n),
941          *           forc_wl(1,n),obs_wl(1,n),delta_wl(1,n),
942          *           tm_obspeak(1,n)
943                if(i_flag(1,n).eq.1)then
944                   write(22,447)n,ievent(1,n),idel_strtm(1,n),
945          *              idel_duration(n),idel_peaktm(1,n),
946          *              forc_wl(1,n),obs_wl(1,n),delta_wl(1,n),
947          *              tm_obspeak(1,n)
948                endif
949                if(i_flag(1,n).eq.0)then
950                   write(23,447)n,ievent(1,n),idel_strtm(1,n),
951          *              idel_duration(n),idel_peaktm(1,n),
952          *              forc_wl(1,n),obs_wl(1,n),delta_wl(1,n),
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
953         *                tm_obspeak(1,n)
954                endif
955       455    continue
956
957              if(nsuccess.gt.0)then
958                 dwlS_mean = dwlS_total/float(nsuccess)
959                 write(21,461)dwlS_mean
960              endif
961
962              if(nsuccess_flood.gt.0)then
963                 dwlS_meanFl = dwlS_totalFl/float(nsuccess_flood)
964                 write(22,464)nsuccess_flood,dwlS_meanFl
965              endif
966              if(nsuccess_ebb.gt.0)then
967                 dwlS_meanEbb = dwlS_totalEbb/float(nsuccess_ebb)
968                 write(23,465)nsuccess_ebb,dwlS_meanEbb
969              endif
970
971
972       c 431 format(/,'success  event  dstart  delta    dpeak',
973       c     *    ' forecast   observed  dspeed  obs peak',/,
974       c     *         ' number  number  time  duration  time ',
975       c     *    ' spd(cm/s)  spd(cm/s) (cm/s)  time(jd)')
976         431 format(/,'success  event  dstart  delta    dpeak',
977            *    '   model    observed  dspeed  obs peak',/,
978            *         ' number  number  time  duration  time ',
979            *    ' spd(cm/s)  spd(cm/s) (cm/s)  time(jd)')
980         447 format(1x,i3,4(3x,i5),2(1x,f10.4),f9.3,f9.3)
981         461 format(' mean difference of peak current speeds for',
982            *     /,' "success" forecasts is',f8.3,' cm/s')
983         464 format(' mean difference of peak current speeds (',i3,
984            *       ' flood events)',/,
985            *       ' for "success" forecasts is', f8.3,'cm/s')
986         465 format(' mean difference of peak current speeds (',i3,
987            *       ' ebb events)',/,
988            *       ' for "failure" forecasts is',f8.3,'cm/s')
989
990       !-------------------------------------------------
991
992       c    Case 2 : Failure
993
994
995              if(nfailure.gt.0)write(21,432)
996              if(nfail_flood.gt.0)write(22,432)
997              if(nfail_ebb.gt.0)write(23,432)
998
999              do 456 n=1,nfailure
1000                write(21,448)n,ievent(2,n),idel_peaktm(2,n),
1001        *            forc_wl(2,n),obs_wl(2,n),delta_wl(2,n),
1002        *            tm_obspeak(2,n)
1003                if(i_flag(2,n).eq.1)then
1004                   write(22,448)n,ievent(2,n),idel_peaktm(2,n),
1005        *               forc_wl(2,n),obs_wl(2,n),delta_wl(2,n),
1006        *               tm_obspeak(2,n)
1007                endif
1008                if(i_flag(2,n).eq.0)then
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
1009                  write(23,448)n,ievent(2,n),idel_peaktm(2,n),
1010         *                forc_wl(2,n),obs_wl(2,n),delta_wl(2,n),
1011         *                tm_obspeak(2,n)
1012                  endif
1013        456     continue
1014
1015              if(nfailure.gt.0)then
1016                 dwlFail_mean = dwlFail_total/float(nfailure)
1017                 write(21,462)dwlFail_mean
1018              endif
1019              if(nfail_flood.gt.0)then
1020                 dwlFail_meanFl = dwlFail_totFl/float(nfail_flood)
1021                 write(22,466)nfail_flood,dwlFail_meanFl
1022              endif
1023              if(nfail_ebb.gt.0)then
1024                 dwlFail_meanEbb = dwlFail_totEbb/float(nfail_ebb)
1025                 write(23,467)nfail_ebb,dwlFail_meanebb
1026              endif
1027
1028
1029        c 432 format(/,
1030        c     *  'failure  event  dpeak  forecast     observed  ',
1031        c     *  ' dspeed  obs peak',/,
1032        c     *  ' number  number time  speed(cm/s)  speed(cm/s)',
1033        c     *  ' (cm/s)  time(jd)')
1034          432 format(/,
1035             *  'failure  event  dpeak   model       observed  ',
1036             *  ' dspeed  obs peak',/,
1037             *  ' number  number time  speed(cm/s)  speed(cm/s)',
1038             *  ' (cm/s)  time(jd)')
1039          462 format(' mean difference of peak current speeds for',
1040             *     /,' "failure" forecasts is',f8.3,' cm/s')
1041          466 format(' mean difference of peak current speeds (',i3,
1042             *       ' flood events)',/,
1043             *       ' for failure forecasts is',f8.3,' cm/s')
1044          467 format(' mean difference of peak current speeds (',i3,
1045             *       ' ebb events)',/,
1046             *       ' for failure forecasts is',f8.3,' cm/s')
1047
1048        !-------------------------------------------------------
1049
1050        c    Case 3 : False
1051
1052
1053              if(nfalse.gt.0)write(21,433)
1054              if(nfalse_flood.gt.0)write(22,433)
1055              if(nfalse_ebb.gt.0)write(23,433)
1056
1057
1058              do 457 n=1,nfalse
1059                 write(21,448)n,ievent(3,n),idel_peaktm(3,n),
1060         *            forc_wl(3,n),obs_wl(3,n),delta_wl(3,n),
1061         *            tm_obspeak(3,n)
1062                 if(i_flag(3,n).eq.1)then
1063                    write(22,448)n,ievent(3,n),idel_peaktm(3,n),
1064         *               forc_wl(3,n),obs_wl(3,n),delta_wl(3,n),
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
1065          *                  tm_obspeak(3,n)
1066                endif
1067                if(i_flag(3,n).eq.0)then
1068                   write(23,448)n,ievent(3,n),idel_peaktm(3,n),
1069          *                  forc_wl(3,n),obs_wl(3,n),delta_wl(3,n),
1070          *                  tm_obspeak(3,n)
1071                endif
1072       457    continue
1073
1074             if(nfalse.gt.0)then
1075                dwlFalse_mean = dwlFalse_total/float(nfalse)
1076                write(21,463)dwlFalse_mean
1077             endif
1078             if(nfalse_flood.gt.0)then
1079                dwlFalse_meanFl = dwlFalse_totFl/float(nfalse_flood)
1080                write(22,468)nfalse_flood,dwlFalse_meanFl
1081             endif
1082             if(nfalse_ebb.gt.0)then
1083                dwlFalse_meanEbb = dwlFalse_totEbb/float(nfalse_ebb)
1084                write(23,469)nfalse_ebb,dwlFalse_meanEbb
1085             endif
1086
1087             write(21,471)nsuccess,nfailure,nfalse
1088       440 continue
1089
1090
1091       c 433 format(/,
1092       c     *  ' false   event  dpeak  forecast     observed  ',
1093       c     *  ' dspeed  obs peak',/,
1094       c     *  ' number  number time  speed(cm/s)  speed(cm/s)',
1095       c     *  ' (cm/s)  time(jd)')
1096         433 format(/,
1097            *  ' false   event  dpeak   model       observed  ',
1098            *  ' dspeed  obs peak',/,
1099            *  ' number  number time  speed(cm/s)  speed(cm/s)',
1100            *  ' (cm/s)  time(jd)')
1101         463 format(' mean difference of peak current speeds for',
1102            *     /,' "false" forecasts is',f8.3,' cm/s')
1103         468 format(' mean difference of peak current speeds (',i3,
1104            *       ' flood events)',/,
1105            *       ' for "false" forecasts is',f8.3)
1106         469 format(' mean difference of peak current speeds (',i3,
1107            *       ' ebb events)',/,
1108            *       ' for "false" forecasts is',f8.3)
1109
1110       !-------------------------------------------------------
1111
1112         446 format(7x,i3,' total points in event;',
1113            *          i3,' model points,',i3,' observed')
1114         448 format(1x,i3,3x,i5,3x,i4,1x,f11.4,f13.4,1x,f8.3,f9.3)
1115         471 format(/,'         Success  Failure  false',/,
1116            *       8x,i5,2(4x,i4)/)
1117         451 format(' for ',i3,' high water events, mean difference',
1118            *     /,' of the peak water levels is ',f7.4,' meters')
1119         452 format(' for ',i3,' low water events, mean difference',
1120            *     /,' of the peak water levels is ',f7.4,' meters')
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
1121
1122    ********************************************************
1123
1124     1001 format(/)
1125     1003 format(a28)
1126     1039 format(a14)
1127     1040 format(/,1x,a14)
1128     1006 format(1x,3f10.4)
1129
1130
1131          stop
1132          end
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
      subroutine prDirection(n,u,v,val)

c     Purpose : Given U and V components of current,
c               to calculate current speed with respect to
c               the specified principal direction.  Current
c               speed values (val) are converted from
c               m/s to cm/s.


c     Input Arguments :
c
c     n - station number
c     u - U component
c     v - V component

*********************************************************

      common/debug/idbug
      common/prindir/prdir(5)

*********************************************************

      rad = 57.29578

      xmag = sqrt(u**2 + v**2)
      if(idbug.eq.5)write(6,1)xmag
      angle = atan2(v,u)
      angle_deg = angle * rad
      if(idbug.eq.5)then
         write(6,2)angle,angle_deg
      endif
      dir = 90.0 - angle_deg
      if(idbug.eq.5)write(6,3)dir
      kdir = dir/360.0
      if(kdir.gt.0)then
         write(6,*)kdir
         dir = dir - kdir*360.0
         write(6,*)dir
      endif

      val = xmag * (cos(dir/rad)*cos(prdir(n)/rad)
     *      + sin(dir/rad)*sin(prdir(n)/rad))
      val = val * 100.0

*********************************************************

    1 format(' magnitude of current velocity = ',f7.4)
    2 format(' angle = ',f7.4,' radians, ',f8.4,' degrees')
    3 format(' direction (rel to 0 deg North) = ',f9.4)

      return
      end
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
1             SUBROUTINE CONCTJ (IJD,IMON,IDAY,IYR)
2       C
3       C**** THIS SUBROUTINE CONVERTS CALENDER TO JULIAN DAY (IJD)
4       C
5             DIMENSION   IDTBLE(12),ILTBLE(12)
6       C
7             DATA (IDTBLE(I),I=1,12)/1,32,60,91,121,152,182,213,244,
8            1                        274,305,335/
9             DATA (ILTBLE(I),I=1,12)/1,32,61,92,122,153,183,214,245,
10           1                        275,306,336/
11      C
12      C**** TEST FOR LEAP YEAR
13      C
14            ISW = 1
15            IF (MOD(IYR,4).EQ.0)   ISW = 2
16
17            GO TO (9,10)   ISW
18          9 IJD = IDTBLE(IMON) + IDAY - 1
19            RETURN
20         10 IJD = ILTBLE(IMON) + IDAY - 1
21            RETURN
22            END
```

```
1             subroutine timehi(caldy,tmmax,ibug)
2
3       c     Purpose : To take the real value julian day
4       c     and convert to calendar day with fraction.
5       c     Version date : November 1, 1997
6
7       ************************************************
8
9             itmhi = tmmax
10            rtmhi = float(itmhi)
11            tmdiff = tmmax - rtmhi
12
13
14            call jdgreg(rtmhi,imonth,iday,iyear)
15
16            caldy = float(iday) + tmdiff
17            if(ibug.eq.2)then
18               write(6,101)imonth,iday,iyear,caldy
19            endif
20
21
22          101 format(1x,i2,'/',i2,'/',i4,'  Calendar Day ',f9.4,/)
23
24            return
25            end
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
1       ! SUBROUTINE NAME : JDGREG
2       !
3       ! PURPOSE : This subroutine converts the Julian date
4       !           to Gregorian date.
5       !
6       !
7       !    VARIABLE NAMES :
8       !    DOUBLE PRECISION RJD  - - - - - - Julian date
9       !    DIMENSION JDAY(13)  - - - - - - - Non-leap year
10      !    DIMENSION JDAYL(13) - - - - - - - Leap year
11      !
12      !-------------------------------------------------
13
14            SUBROUTINE JDGREG(rjdy,imon,ida,iyr)
15
16
17      !:      DOUBLE PRECISION RJD
18            DIMENSION JDAY(13), JDAYL(13)
19
20            DATA JDAY/0,31,59,90,120,151,181,212,243,273,304,334,365/
21            DATA JDAYL/0,31,60,91,121,152,182,213,244,274,305,335,366/
22
23            iyr = 2000
24
25            rjd = rjdy
26            IDA = INT(RJD)
27
28
29            IF(MOD(IYR,4).EQ.0.AND.MOD(IYR,100).NE.0
30           &    .OR. MOD(IYR,400) .EQ. 0)THEN
31
32      ! Find the month for IDAY --- leap year calender
33      !
34            ILEAP = 1
35            DO I = 1,12
36               IF (IDA.GT.JDAYL(I)) IMO = I
37            END DO
38
39      ! Day of the month
40
41               IDY = IDA - JDAYL(IMO)
42            ELSE
43
44      ! Find the month for IDAY --- non-leap year
45
46               ILEAP = 0
47               DO I=1,12
48                  IF(IDA.GT.JDAY(I)) IMO = I
49               ENDDO
50
51      ! Day of the month
52
53               IDY = IDA-JDAY(IMO)
54            ENDIF
55
56            IMON = IMO
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

```
57              IDA = IDY
58
59              RETURN
60              END
```

Program Listing 2.5. Match.evnt_crnt.f (continued)

## 2.6. Program Curr.prdir.pro

The listing for Curr.prdir.pro is given in Program Listing 2.6. This is an plot program used to plot a month of observed, forecast, and nowcast current speed data. In addition to plotting the current speed, the program will plot two lines which designate the low critical current speed and the high critical current speed. A value more negative than the low critical current speed is part of an ebb event, while a value greater than the high critical current speed is part of a flood event.

The plots are annotated with a title, station name, and a legend. More than one current speed signal can be put on one plot, if desired. The program generates only one plot per page.

```
1       ; Program : curr.prdir.pro
2       ;
3       ; Purpose : This program, written in the IDL programming
4       ; language, was developed to help evaluate Houston/Galveston
5       ; nowcast/forecast current data.  The program will plot
6       ; the observed (PORTS) and model current speeds with respect
7       ; to a principal direction.  This is a revision of the program
8       ; wl.sigma.pro.  This version of the program accounts for
9       ; gaps in the data stream.
10      ;
11      ;
12      ; Language : IDL
13      ;
14      ; Location : /usr/people/philr/galves/nowforc_eval/currents/plot
15      ;
16      ; Version date : January 18, 2001
17      ;
18      ; Author : Phil Richardson
19
20      ;******************************************************
21
22        im = 1500
23        nsig = 2
24        nline = 12
25
26
27      ;Initialize character strings
28        line = ' '
29        wltitle = ' '
30        filedata = ' '
31        legend = ' '
32        cntrl_file = ' '
33        time_axis = ' '
34        stat_name = ' '
35        ptype=' '
36        plottype = ' '
37
38      ;Initialize integer variables
39        idebug = 0
40        iyear = 0
41        ncurve = 0
42        lun = 0
43
44      ;Dimension arrays
45        ilun=intarr(nsig)
46        numb_pts = intarr(nsig)
47        ncg = intarr(nline)
48        ncntg = intarr(nline)
49
50        filedat = strarr(2)
51        legnd = strarr(2)
52        ravg = strarr(2)
53
54        t=fltarr(im,nline)
55        wlplt = fltarr(im,nline)
56        xl=fltarr(2,2)
```

Program Listing 2.6. Curr.prdir.pro

```
57        y1=fltarr(2)
58        xpos = fltarr(2)
59        time_strt = fltarr(2)
60        xline = fltarr(2)
61        crlevel_high = fltarr(2)
62        crlevel_low = fltarr(2)
63
64
65      ; Initialize Real variables
66
67        small = 0.001
68        crlevel = 0.0
69        hr_intrvl = 0.09
70
71      ;************************************************
72
73      ; Open control file, read from control file
74
75      ;       ptype - x, ps, or tek
76      ;   idebug = 1, times (Julian dates)
77      ;          = 2, EOF result
78      ;   stat_name - station name
79      ;        tmin - start point for time (x) axis
80      ;    strttime - start time (Julian date)
81      ;        tmax - end point for time (x) axis
82      ;     endtime - end time (Julian date)
83      ;       iyear - year of plot
84      ;      ncurve - number of curves to plot
85      ;   legnd(nc) - character string, for legend
86      ; filedat(nc) - data filenames
87      ;   time_axis - time axis name
88      ;     crlevel - critical value
89
90
91        print,'Enter name of control file '
92        read,cntrl_file
93      ; cntrl_file = 'cntrl.bolvr_forc'
94        openr,1,cntrl_file
95
96
97        readf,1,ptype
98        if(ptype eq 'ps')then begin
99           readf,1,plottype
100       endif
101       readf,1,idebug
102       print,idebug,format='(2x,"idebug = ",i3)'
103
104       readf,1,stat_name
105       readf,1,tmin
106       readf,1,strttime
107       readf,1,tmax
108       readf,1,endtime
109       readf,1,iyear
110       readf,1,ncurve
111       print,ncurve,format='(1x,i2," curves to be plotted",/)
112       ncurvm1 = ncurve - 1
```

Program Listing 2.6. Curr.prdir.pro (continued)

```
113
114       readf,1,wltitle
115
116       for nc=0,ncurvml do begin
117           get_lun,lun
118           ilun(nc) = lun
119           readf,1,legend
120           legnd(nc) = legend
121           readf,1,filedata
122           filedat(nc) = filedata
123       endfor
124
125       readf,1,ymin,ymax,ytcks
126       readf,1,time_axis
127
128       readf,1,crlevel
129
130
131       close,1
132
133     ;-------------------------------------------------
134
135     ;set plot type : x, ps, or tek
136      set_plot,ptype
137
138     ;set the plot scaling
139      aspect=1.5
140      isize=1024.
141      jsize=isize*aspect
142
143      if (ptype eq 'x') then window,0,xsize=isize,ysize=jsize
144      xs=8.0
145      ys=8.0*aspect
146
147      if(ptype eq 'ps')then begin
148         if(plottype eq 'portrait')then begin
149            device, xsize=xs,$
150                    ysize=ys,/inch,xoffs=0.25,yoffs=0.
151         endif
152         if(plottype eq 'landscape')then begin
153            device, ysize=10.0, /landscape,$
154                   /inches, xoffs=-2.0
155         endif
156      endif
157
158     ;***********************************************
159
160     ; Open observed wl data file and model wl data file
161
162     ; variables :
163     ;   ndatpts - number of data points
164
165
166      for nc=0,ncurvml do begin
167          openr,ilun(nc),filedat(nc),error=err
168          if(err ne 0) then print, !err_string
```

Program Listing 2.6. Curr.prdir.pro (continued)

```
169          print,nc,filedat(nc),    $
170                format='(1x,"Curve ",i2," ; file",a71)'
171       endfor
172
173      ;-------------------------------------------------------
174
175      ; Read data from files
176
177       if(idebug eq 1)then openw,4,'time.out'
178
179
180       for nc=0,ncurvm1 do begin
181           ncpl1 = nc + 1
182           ncount = 0
183           nlin = 1
184           nlinm1 = nlin - 1
185           if(idebug eq 1)then begin
186              printf,4,filedat(nc),format='(1x,a73)'
187           endif
188
189           readf,ilun(nc),time
190           print,ncpl1,time,                $
191           format='(/,1x,"file (",i1,") starts at time =",f8.3)'
192           print,time,strttime
193           point_lun,ilun(nc),0
194
195           READDATA: readf,ilun(nc),time,wlevel
196           result = EOF(ilun(nc))
197           if(idebug eq 2)then print,result
198           if(time lt strttime)then goto, READDATA
199           if(time gt endtime)then begin
200              ncount = ncount - 1
201              ndatpts = ncount + 1
202              goto, ENDLOOP
203           endif
204           if(ncount eq 0)then begin
205              time_old = time
206              print,ncpl1,time,        $
207              format='(1x,"start time file (",i1,") = ",f8.3)'
208              time_strt(nc) = time
209           endif
210
211
212           if(time lt 366.0)then begin
213              jd_offset = 0.0
214           endif
215           time = time - jd_offset
216
217           if(result lt 1)then begin
218              if(idebug eq 1)then printf,4,ncount,time
219              ncount = ncount + 1
220              time_dif = time - time_old
221              if(time_dif gt hr_intrvl)then begin
222                 print,format='("gap in data file")'
223                 if(idebug eq 1)then begin
224                 printf,4,time_old,time,time_dif,    $
```

Program Listing 2.6. Curr.prdir.pro (continued)

```
                    format='(1x,3f9.3,", gap in data file")'
               endif
               nlin = nlin + 1
               nlinm1 = nlin - 1
               ncg(nlinm1) = 0
               t(ncg(nlinm1),nlinm1) = time
               wlplt(ncg(nlinm1),nlinm1) = wlevel
               ncg(nlinm1) = ncg(nlinm1) + 1
            endif
            if(time_dif lt hr_intrvl)then begin
               t(ncg(nlinm1),nlinm1) = time
               wlplt(ncg(nlinm1),nlinm1) = wlevel
               ncg(nlinm1) = ncg(nlinm1) + 1
            endif
            time_old = time
            goto, READDATA
         endif

;        End of File
         if(result gt 0)then begin
            if(idebug eq 1)then printf,4,ncount,time
            print,ncpl1,              $
                 format='(" End of file (",i1,") reached")'
;           t(ncg(nlinm1),nlinm1) = time
;           wlplt(ncg(nlinm1),nlinm1) = wlevel
         endif
       close,ilun(nc)
       ndatpts = ncount + 1
       ENDLOOP: print,ndatpts,       $
                format='(i4," data points, End of loop")'
       numb_pts(nc) = ncount
       endfor


       print,ncount,format='(/,1x,i4)'

       print,ndatpts,format='(1x,i4)'

      !p.multi=[0,0,1]

      ;-----------------------------------------------

      ; make the plot

      !P.CHARSIZE=1.0


        ncnt = numb_pts(0)
        ncntl = ncg(0) - 1
        print,ncnt1
        nticks = 4
      @plot01
        plot,t[0:ncnt1],wlplt[0:ncnt1,0],              $
           title=wltitle,              $
           yrange=[ymin,ymax],         $
           xtitle=time_axis,           $
```

Program Listing 2.6. Curr.prdir.pro (continued)

```
281          ytitle='cm/s',              $
282          xmargin=[0,0],              $
283          ymargin=[0,0],              $
284          xstyle=1,ystyle=1,          $
285          linestyle=0,                $
286          xrange=[tmin,tmax],         $
287          xticks = nticks,            $
288          yticks = ytcks,             $
289          position=[0.10,0.52,0.90,0.87]
290        for nl=2,nlin do begin
291            nlm1 = nl - 1
292            ncntg(nlm1) = ncg(nlm1) - 1
293            oplot,t[0:ncntg(nlm1),nlm1],wlplt[0:ncntg(nlm1),nlm1]
294        endfor
295
296
297        xyouts,0.50,0.55,stat_name,size=1.5,/normal,alignment=0.5
298
299      ;*******************************************************
300
301      ; Draw Legend
302
303      ; Establish x,y coordinates for legend
304
305
306        x1(0,0) = 0.36
307        x1(0,1) = 0.44
308        y1(0) = 0.825
309        x1(1,0) = 0.62
310        x1(1,1) = 0.67
311        y1(1) = 0.825
312
313        xpos(0) = 0.25
314        xpos(1) = 0.49
315        ypos = 0.83
316
317        for nc=0,ncurvm1 do begin
318            xyouts,xpos(nc),ypos,legnd(nc),size=1.4,/NORMAL
319            if(nc eq 0)then linest = 0
320            if(nc eq 1)then linest = 1
321        endfor
322            plots,[x1(0,0),x1(0,1)],y1,linestyle=0,  $
323                  /normal
324      ;     plots,[x1(1,0),x1(1,1)],y1,psym=2,/normal
325
326      ;-------------------------------------------------------
327
328      ; Draw solid lines representing low and high critical
329      ; values for events.
330
331        xline(0) = tmin
332        xline(1) = tmax
333
334        crlevel_high(0) = crlevel
335        crlevel_high(1) = crlevel
336        crlevel_low(0) = -crlevel
```

Program Listing 2.6. Curr.prdir.pro (continued)

```
337       crlevel_low(1) = -crlevel
338
339
340       plots,xline,crlevel_high
341       plots,xline,crlevel_low
342
343     ;-------------------------------------------------------
344
345       if(ptype eq 'ps') then device,/close
346
347       ENDPROG:
348
349       end
```

Program Listing 2.6. Curr.prdir.pro (continued)

## 2.7. Program Curr.multcurv.pro

Curr.multcurv.pro, also an IDL program, is an improved version of curr.prdir.pro. The program will generate plots of observed versus nowcast and observed versus predicted current speeds on one page, then observed versus forecast and observed versus adjusted forecast current speeds on the second page.

```
1       ; Program : curr.multcurv.pro
2       ;
3       ; Purpose : This program, written in the IDL programming
4       ; language, was developed to help evaluate Houston/Galveston
5       ; nowcast/forecast current data.  The program will
6       ; produce four separate plots of principal component current
7       ; speed: OBS vs.Nowcast, OBS vs. Predicted, OBS vs. Forecast,
8       ; and OBS vs. adjusted Forecast.  On each plot, the high
9       ; and low critical values, which determine events, are depicted.
10      ;
11      ; Language : IDL
12      ;
13      ; Location : /usr/people/philr/galves/NF_eval/wlevel/plot
14      ;
15      ; Version date : August 6, 2001
16      ;
17      ; Author : Phil Richardson
18
19      ;******************************************************
20
21        im = 1500
22        nline = 12
23      ; nline = 120
24        ncurves = 5
25
26
27      ;Initialize character strings
28        line = ' '
29        crnt_title = ' '
30        filedata = ' '
31        legend = ' '
32        cntrl_file = ' '
33        time_axis = ' '
34        stat_name = ' '
35        ptype=' '
36        plottype = ' '
37        typedata = ' '
38
39      ;Initialize integer variables
40        idebug = 0
41        ncurve = 0
42        lun = 0
43        ncleg = 0
44
45      ;Dimension arrays
46      ; ncg = intarr(nline)
47
48        ilun = intarr(ncurves)
49        numb_pts = intarr(ncurves)
50
51        filedat = strarr(ncurves)
52        typedat = strarr(ncurves)
53        legnd = strarr(ncurves)
54        ravg = strarr(2)
55
56        t = fltarr(im,nline)
```

Program Listing 2.7. Curr.multcurv.pro

```
57        wlplt = fltarr(im,nline)
58        x1=fltarr(ncurves,2)
59        y1=fltarr(2)
60        y11=fltarr(2)
61        xpos = fltarr(ncurves)
62        time_strt = fltarr(ncurves)
63        xline = fltarr(2)
64        crlevel_high = fltarr(2)
65        crlevel_low = fltarr(2)
66
67
68        small = 0.001
69        crlevel = 0.0
70        hr_intrvl = 0.09
71
72      ;*****************************************************
73
74      ; Open control file, read from control file
75
76      ;       ptype - x, ps, or tek
77      ;   idebug = 1, times (Julian dates)
78      ;          = 2, EOF result
79      ;   stat_name - station name
80      ;        tmin - start point for time (x) axis
81      ;    strttime - start time (Julian date)
82      ;        tmax - end point for time (x) axis
83      ;     endtime - end time (Julian date)
84      ;      ncurve - number of curves to plot
85      ;   legnd(nc) - character string, for legend
86      ; filedat(nc) - data filenames
87      ;  ymin, ymax - Y axis
88      ;   time_axis - time axis name
89      ;     crlevel - critical value
90
91
92        print,'Enter name of control file '
93        read,cntrl_file
94        openr,1,cntrl_file
95
96        readf,1,ptype
97        if(ptype eq 'ps')then begin
98           readf,1,plottype
99        endif
100       readf,1,idebug
101       print,idebug,format='(2x,"idebug = ",i3)'
102
103       readf,1,stat_name
104       readf,1,tmin
105       readf,1,strttime
106       readf,1,tmax
107       readf,1,endtime
108       readf,1,ncurve
109       print,ncurve,format='(1x,i2," curves to be plotted",/)
110       ncurvm1 = ncurve - 1
111
112       readf,1,crnt_title
```

Program Listing 2.7. Curr.multcurv.pro (continued)

```
113
114       for nc=0,ncurvm1 do begin
115           get_lun,lun
116           ilun(nc) = lun
117           readf,1,legend
118           legnd(nc) = legend
119           readf,1,typedata
120           readf,1,filedata
121           typedat(nc) = typedata
122           filedat(nc) = filedata
123       endfor
124
125       readf,1,ymin,ymax,ytcks
126       readf,1,time_axis
127
128       readf,1,crlevel
129
130
131     ; close,1
132
133     ;-----------------------------------------------
134
135     ;set plot type : x, ps, or tek
136      set_plot,ptype
137
138     ;set the plot scaling
139      aspect=1.5
140      isize=1024.
141      jsize=isize*aspect
142
143      if (ptype eq 'x') then window,0,xsize=isize,ysize=jsize
144      xs=8.0
145      ys=8.0*aspect
146
147      if(ptype eq 'ps')then begin
148         if(plottype eq 'portrait')then begin
149            device, xsize=xs,$
150                    ysize=ys,/inch,xoffs=0.25,yoffs=0.
151         endif
152         if(plottype eq 'landscape')then begin
153            device, ysize=10.0, /landscape,$
154                   /inches, xoffs=-2.0
155         endif
156      endif
157
158     ;*****************************************************
159
160     ; Open observed wl data file and model wl data file
161
162     ; variables :
163     ;   ndatpts - number of data points
164
165
166      for nc=0,ncurvm1 do begin
167          openr,ilun(nc),filedat(nc),error=err
168          if(err ne 0) then print, !err_string
```

Program Listing 2.7. Curr.multcurv.pro (continued)

```
169          print,nc,filedat(nc),    $
170                format='(1x,"Curve ",i2," ; file",a67)'
171       endfor
172
173      ;-------------------------------------------------------
174
175      ; Read data from files
176
177       if(idebug eq 1)then openw,4,'time.out'
178
179
180       for nc=0,ncurvm1 do begin
181           ncpl1 = nc + 1
182           nlin = 1
183           nlinm1 = nlin - 1
184           ncount = 0
185           if(idebug eq 1)then begin
186              printf,4,filedat(nc),format='(1x,a77)'
187           endif
188
189           readf,ilun(nc),time
190           print,ncpl1,time,              $
191           format='(/,1x,"file (",i1,") starts at time =",f8.3)'
192           print,time,strttime
193           point_lun,ilun(nc),0
194
195           READDATA: readf,ilun(nc),time,wlevel
196           result = EOF(ilun(nc))
197           if(idebug eq 2)then print,result
198           if(time lt strttime)then goto, READDATA
199           if(time gt endtime)then begin
200              ncount = ncount - 1
201              ndatpts = ncount + 1
202              goto, ENDLOOP
203           endif
204           if(ncount eq 0)then begin
205              time_old = time
206              print,ncpl1,time,        $
207              format='(1x,"start time file (",i1,") = ",f8.3)'
208              time_strt(nc) = time
209           endif
210
211
212           if(time lt 366.0)then begin
213              jd_offset = 0.0
214           endif
215           time = time - jd_offset
216
217           if(result lt 1)then begin
218              if(idebug eq 1)then printf,4,ncount,time
219              t(ncount,nc) = time
220              wlplt(ncount,nc) = wlevel
221              ncount = ncount + 1
222              time_dif = time - time_old
223      ;       if(time_dif gt hr_intrvl)then begin
224      ;          print,format='("gap in data file")'
```

Program Listing 2.7. Curr.multcurv.pro (continued)

```
225     ;          if(idebug eq 1)then begin
226     ;             printf,4,time_old,time,time_dif,    $
227     ;                  format='(1x,3f9.3,", gap in data file")'
228     ;          endif
229     ;          nlin = nlin + 1
230     ;          nlinm1 = nlin - 1
231     ;          ncg(nlinm1) = 0
232     ;          t(ncg(nlinm1),nlinm1) = time
233     ;          wlplt(ncg(nlinm1),nlinm1) = wlevel
234     ;          ncg(nlinm1) = ncg(nlinm1) + 1
235     ;       endif
236     ;       if(time_dif lt hr_intrvl)then begin
237     ;          t(ncg(nlinm1),nlinm1) = time
238     ;          wlplt(ncg(nlinm1),nlinm1) = wlevel
239     ;          wlplt(ncount,nc) = wlevel
240     ;          ncg(nlinm1) = ncg(nlinm1) + 1
241     ;       endif
242             time_old = time
243             goto, READDATA
244          endif
245          if(result gt 0)then begin
246             if(idebug eq 1)then printf,4,ncount,time
247             print,ilun(nc),        $
248                   format='(" End of file (",i1,") reached")'
249             t(ncount,nc) = time
250             wlplt(ncount,nc) = wlevel
251          endif
252       close,ilun(nc)
253       ndatpts = ncount + 1
254       ENDLOOP: print,ndatpts,       $
255                format='(i4," data points, End of loop")'
256       numb_pts(nc) = ndatpts - 1
257       endfor
258
259
260
261     !p.multi=[0,0,1]
262
263     ;*****************************************************
264
265     ; make the plot
266
267     !P.CHARSIZE=1.0
268
269     ; Variables :
270     ;
271     ;    ytl - Y coordinate (window) of top plot
272     ;    ybl - Y coordinate (window) of bottom of top plot
273     ; ystnmT - Y coordinate of station name (top plot)
274     ; ystnmB - Y coordinate of station name (bottom plot)
275     ;  yposT - Y coordinate of legend (top plot)
276     ;  yposB - Y coordinate of legend (bottom plot)
277     ;     yl - Y coordinate of legend (line), top plot
278     ;    yll - Y coordinate of legend (line), bottom plot
279
280     ; Establish x,y coordinates for legend
```

Program Listing 2.7. Curr.multcurv.pro (continued)

```
281
282       x1(0,0) = 0.36
283       x1(0,1) = 0.44
284       y1(0) = 0.91
285       y1(0) = 0.71
286       x1(1,0) = 0.62
287       x1(1,1) = 0.70
288       y1(1) = 0.71
289       y11(0) = 0.33
290       y11(1) = 0.33
291
292       xpos(0) = 0.25
293       xpos(1) = 0.49
294       yposT = 0.71
295       yposB = 0.33
296
297
298       xline(0) = tmin
299       xline(1) = tmax
300
301       crlevel_high(0) = crlevel
302       crlevel_high(1) = crlevel
303       crlevel_low(0) = -crlevel
304       crlevel_low(1) = -crlevel
305
306
307       nticks = 5
308       ncnt1 = numb_pts(0)
309       ncnt2 = numb_pts(1)
310       ncnt3 = numb_pts(2)
311       ncnt4 = numb_pts(3)
312       ncnt5 = numb_pts(4)
313       print,ncnt1
314       print,ncnt2
315       print,ncnt3
316       print,ncnt4
317       print,ncnt5
318
319     ;-------------------------------------------------------
320
321     ; 2 plots per page -
322
323     !P.Multi = [0,1,2,0,0]
324
325
326     ; OBS vs. Nowcast
327
328     @plot01
329
330       yt1 = 0.95
331       yb1 = 0.68
332       ystnmT = 0.70
333
334       plot,t[0:ncnt1,0],wlplt[0:ncnt1,0],          $
335           title=crnt_title,          $
336           yrange=[ymin,ymax],        $
```

Program Listing 2.7. Curr.multcurv.pro (continued)

```
337           xtitle=time_axis,          $
338           ytitle='cm/s',             $
339           xmargin=[0,0],             $
340           ymargin=[0,0],             $
341           xstyle=1,ystyle=1,         $
342           linestyle=0,               $
343           xrange=[tmin,tmax],        $
344           xticks = nticks,           $
345           yticks = ytcks,            $
346           position=[0.10,yb1,0.90,yt1]
347           oplot,t[0:ncnt2,1],wlplt[0:ncnt2,1],       $
348                 linestyle=2
349
350
351     ; xyouts,0.50,ystnmT,stat_name,size=1.5,/normal,alignment=0.5
352
353       for nc=0,1 do begin
354           xyouts,xpos(nc),yposT,legnd(nc),size=1.4,/NORMAL
355           if(nc eq 0)then linest = 0
356           if(nc eq 1)then linest = 1
357           plots,[x1(nc,0),x1(nc,1)],y1,linestyle=linest, $
358                 /NORMAL
359       endfor
360
361       plots,xline,crlevel_high
362       plots,xline,crlevel_low
363
364     ;-------------------------------------------------
365
366     ; OBS vs. Predicted
367
368       yt2 = 0.57
369       yb2 = 0.30
370       ystnmB = 0.32
371
372       plot,t[0:ncnt1,0],wlplt[0:ncnt1,0],                $
373           title=crnt_title,             $
374           yrange=[ymin,ymax],           $
375           xtitle=time_axis,             $
376           ytitle='cm/s',                $
377           xmargin=[0,0],                $
378           ymargin=[0,0],                $
379           xstyle=1,ystyle=1,            $
380           linestyle=0,                  $
381           xrange=[tmin,tmax],           $
382           xticks = nticks,              $
383           yticks = ytcks,               $
384           position=[0.10,yb2,0.90,yt2]
385           oplot,t[0:ncnt3,2],wlplt[0:ncnt3,2],           $
386                 linestyle=2
387
388     ; xyouts,0.50,ystnmB,stat_name,size=1.5,/normal,alignment=0.5
389
390
391     ; Draw Legend
392
```

Program Listing 2.7. Curr.multcurv.pro (continued)

```
393       for nc=0,1 do begin
394           ncleg = nc * 2
395           xyouts,xpos(nc),yposB,legnd(ncleg),size=1.4,/NORMAL
396           if(nc eq 0)then linest = 0
397           if(nc eq 1)then linest = 1
398           plots,[x1(nc,0),x1(nc,1)],y11,linestyle=linest,  $
399                 /normal
400       endfor
401
402
403     ; Draw solid lines representing low and high critical
404     ; values for events.
405
406
407       plots,xline,crlevel_high
408       plots,xline,crlevel_low
409
410     ;-------------------------------------------------------
411
412     ; OBS vs. Forecast
413
414       plot,t[0:ncnt1,0],wlplt[0:ncnt1,0],                  $
415           title=crnt_title,              $
416           yrange=[ymin,ymax],            $
417           xtitle=time_axis,              $
418           ytitle='cm/s',                 $
419           xmargin=[0,0],                 $
420           ymargin=[0,0],                 $
421           xstyle=1,ystyle=1,             $
422           linestyle=0,                   $
423           xrange=[tmin,tmax],            $
424           xticks=nticks,                 $
425           yticks=ytcks,                  $
426           position=[0.10,yb1,0.90,yt1]
427           oplot,t[0:ncnt4,3],wlplt[0:ncnt4,3],             $
428                 linestyle=2
429
430     ; xyouts,0.50,ystnmT,stat_name,size=1.5,/normal,alignment=0.5
431
432       for nc=0,1 do begin
433           ncleg = nc * 3
434           xyouts,xpos(nc),yposT,legnd(ncleg),size=1.4,/NORMAL
435           if(nc eq 0)then linest = 0
436           if(nc eq 1)then linest = 1
437           plots,[x1(nc,0),x1(nc,1)],y1,linestyle=linest,  $
438                 /normal
439       endfor
440
441       plots,xline,crlevel_high
442       plots,xline,crlevel_low
443
444     ;-------------------------------------------------------
445
446     ; OBS vs. adjusted Forecast
447
448       plot,t[0:ncnt1,0],wlplt[0:ncnt1,0],                  $
```

Program Listing 2.7. Curr.multcurv.pro (continued)

```
449             title=crnt_title,              $
450             yrange=[ymin,ymax],            $
451             xtitle=time_axis,              $
452             ytitle='cm/s',                 $
453             xmargin=[0,0],                 $
454             ymargin=[0,0],                 $
455             xstyle=1,ystyle=1,             $
456             linestyle=0,                   $
457             xrange=[tmin,tmax],            $
458             xticks=nticks,                 $
459             yticks=ytcks,                  $
460             position=[0.10,yb2,0.90,yt2]
461             oplot,t[0:ncnt5,4],wlplt[0:ncnt5,4],          $
462                   linestyle=2
463
464     ; xyouts,0.50,ystnmB,stat_name,size=1.5,/normal,alignment=0.5
465
466       for nc=0,1 do begin
467           ncleg = nc * 4
468           xyouts,xpos(nc),yposB,legnd(ncleg),size=1.4,/NORMAL
469           if(nc eq 0)then linest = 0
470           if(nc eq 1)then linest = 1
471           plots,[xl(nc,0),xl(nc,1)],yll,linestyle=linest, $
472                 /normal
473       endfor
474
475       plots,xline,crlevel_high
476       plots,xline,crlevel_low
477
478     ;*****************************************************
479
480       spawn, 'lp -dqms2 idl.ps'
481
482     ;*****************************************************
483
484       if(ptype eq 'ps') then device,/close
485
486       ENDPROG:
487
488       end
```

Program Listing 2.7. Curr.multcurv.pro (continued)

## 3. SEPTEMBER 2000 SAMPLE APPLICATION

To perform the principal component direction current event analysis, each program is executed in the order shown in Table 3.1. For each program a separate directory is recommended as shown below, where ~ designates the users home area, galves the project directory, and NF_eval/currents the nowcast/forecast current event evaluation directory.

~/galves/NF_eval/currents/ports/reform.ports/reform_ports.f
~/galves/NF_eval/currents/ports/regap.f
~/galves/NF_eval/currents/ports/reform2.f
~/galves/NF_eval/currents/read_NF.curr.f
~/galves/NF_eval/currents/sa.current/match.evnt_crnt.f
~/galves/NF_eval/currents/plot/curr.prdir.pro
~/galves/NF_eval/currents/plot/curr.multcurv.pro

Table 3.1. Job Control, Source File, and Control File Inventory

| Job Control | Source File | Control File |
|---|---|---|
| reform.jcl | reform_ports.f | reform.(station).n |
| regap.jcl | regap.f | regap.n |
| reform2.jcl | reform2.f | reform2.n |
| readcurr.jcl | read_NF.curr.f | readcrr_sep00.n |
| match.jcl | match.evnt_crnt.f | match_(station).sep00.n |
| | curr.prdir.pro | cntrl.(station)_sep00 |
| | curr.multcurv.pro | c.(station)_sep00 |

Listings for jcl files and control files are provided in Appendix A. The two IDL plot programs do not have job control files. To run the IDL programs, type idl <return>, then type .r filename.pro <return>.

Program output files for the evaluation of the forecast current speeds at Port Bolivar in lower Galveston Bay and at Morgans Point in upper Galveston Bay are given. For this application, current speeds of 65 cm/s and 45 cm/s or greater at Bolivar Roads and at Morgans Point, respectively, are considered to be flood events. Ebb current events consists of current speeds of opposite sign with the same magnitudes. For September 2000, table2.out, table_flood, and table_ebb are provided for Port Bolivar (GBM), Morgans Point (HSC), and Morgans Point (GBM), for the unadjusted forecasts. To adjust the forecasts, a separate linear regression (requiring specification of gain and bias) is used on flood (positive current values) and on ebb (negative current values). Table2.out, table_flood, and

table_ebb are provided for Morgans Point (HSCM) and Morgans Point (GBM), both adjusted forecasts. The adjusted forecast has a gain of 1.0 for flood values and a gain of 2.0 for ebb values.

Plots of the observed (PORTS) and forecast data from Port Bolivar are presented in Figure 3.1. The forecast at Port Bolivar was of high quality and our attempts to adjust the forecast data did not seem to signficantly improve the forecast. Plots of the observed (PORTS), unadjusted forecast (HSCM), and the adjusted forecast (HSCM) data at Morgans Point are presented in Figure 3.2. Plots of the observed (PORTS), unadjusted forecast (GBM), and the adjusted forecast (GBM) data at Morgans Point are presented in Figure 3.3.

Table 3.2a. Table2.out for Port Bolivar (GBM) forecast, unadjusted

September 2000

Bolivar Roads ; Model Level 3
Principal Direction : 322.0 degr
critical value (current speed along principal direction)
= 65.00 cm/s
Flood : gain = 1.00, bias = 0.00
Ebb : gain = 1.00, bias = 0.00
Julian start time from control file 245.00

| success number | event number | dstart time | delta duration | dpeak time | model spd(cm/s) | observed spd(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 1 | -2 | -1 | 68.5779 | 73.8338 | -5.256 | 248.917 |
| 2 | 7 | 0 | -1 | -1 | 68.1463 | 79.7989 | -11.653 | 250.000 |
| 3 | 8 | 0 | 0 | 0 | 72.4272 | 89.0561 | -16.629 | 251.000 |
| 4 | 9 | -2 | 1 | -2 | 72.4322 | 68.2427 | 4.189 | 252.083 |
| 5 | 10 | 0 | 0 | -1 | 81.8255 | 81.3558 | 0.470 | 253.042 |
| 6 | 11 | 0 | 0 | 0 | 65.2592 | 79.1005 | -13.841 | 253.125 |
| 7 | 12 | -2 | 1 | -1 | 71.0408 | 78.4170 | -7.376 | 254.083 |
| 8 | 14 | -3 | 1 | -3 | 74.2619 | 78.1822 | -3.920 | 255.167 |
| 9 | 15 | 0 | 0 | -1 | 69.4340 | 79.5555 | -10.121 | 256.167 |
| 10 | 22 | 1 | -1 | 1 | 65.2547 | 75.2603 | -10.006 | 261.750 |
| 11 | 23 | -1 | 1 | 0 | -69.8492 | -65.8917 | -3.958 | 262.542 |
| 12 | 24 | 0 | -1 | 0 | 70.7803 | 76.8547 | -6.074 | 262.792 |
| 13 | 25 | 0 | 2 | 1 | -71.7524 | -75.3568 | 3.604 | 263.500 |
| 14 | 26 | -1 | 2 | 0 | 75.6857 | 72.7120 | 2.974 | 263.875 |
| 15 | 27 | 0 | 0 | 0 | -70.5719 | -69.2458 | -1.326 | 264.500 |
| 16 | 28 | 0 | 1 | 0 | -72.0235 | -75.7092 | 3.686 | 264.583 |
| 17 | 29 | 0 | 1 | -1 | 73.4379 | 86.7661 | -13.328 | 264.958 |
| 18 | 30 | 0 | 0 | 0 | -74.9596 | -67.9850 | -6.975 | 265.583 |
| 19 | 31 | -1 | -1 | -1 | 78.6891 | 95.0360 | -16.347 | 266.000 |
| 20 | 32 | 0 | -2 | -1 | -68.4796 | -80.0462 | 11.567 | 266.708 |
| 21 | 33 | -1 | 0 | -1 | 78.2737 | 80.7464 | -2.473 | 267.042 |
| 22 | 34 | 0 | -1 | -1 | -74.3544 | -77.2811 | 2.927 | 267.708 |
| 23 | 35 | 0 | -1 | 0 | 69.6401 | 76.7522 | -7.112 | 273.708 |

mean difference of peak current speeds for
"success" forecasts is 7.209 cm/s

Table 3.2a. Table2.out for Port Bolivar (GBM) forecast, unadjusted (continued)

| failure number | event number | dpeak time | model speed(cm/s) | observed speed(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | -38.2003 | -65.6662 | 27.466 | 245.042 |
| 2 | 2 | 0 | -55.9332 | -78.1056 | 22.172 | 246.542 |
| 3 | 3 | 0 | 58.4141 | 67.9969 | -9.583 | 246.792 |
| 4 | 4 | -2 | 57.4917 | 71.3969 | -13.905 | 247.917 |
| 5 | 6 | 0 | -56.2490 | -78.0616 | 21.813 | 249.625 |
| 6 | 13 | -2 | 15.9248 | -81.7844 | 97.709 | 254.833 |
| 7 | 16 | 0 | -60.8915 | -69.6052 | 8.714 | 256.875 |
| 8 | 18 | 0 | 54.9804 | 69.1343 | -14.154 | 258.208 |
| 9 | 19 | 0 | -24.3489 | -69.2458 | 44.897 | 258.917 |
| 10 | 20 | 0 | 50.2201 | 69.7272 | -19.507 | 259.708 |
| 11 | 36 | 0 | -53.8136 | -69.1284 | 15.315 | 274.458 |
| 12 | 37 | 0 | 55.2503 | 81.9747 | -26.724 | 274.708 |

mean difference of peak current speeds for "failure" forecasts is 26.830 cm/s

| false number | event number | dpeak time | model speed(cm/s) | observed speed(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|
| 1 | 17 | -1 | 68.3763 | 59.5663 | 8.810 | 257.167 |
| 2 | 21 | 0 | -65.9947 | -53.7106 | -12.284 | 261.500 |

mean difference of peak current speeds for "false" forecasts is 10.547 cm/s

| Success | Failure | false |
|---|---|---|
| 23 | 12 | 2 |

Table 3.2b. Table2.out for Morgans Point (HSC) forecast, unadjusted

Morgans Point ; Model Level 3
Principal Direction : 341.0 degr
critical value (current speed along principal direction)
= 40.00 cm/s
Flood : gain = 1.00, bias = 0.00
Ebb : gain = 1.00, bias = 0.00
Julian start time from control file 245.37

| failure number | event number | dpeak time | model speed(cm/s) | observed speed(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | -18.4786 | -41.1084 | 22.630 | 246.583 |
| 2 | 2 | 0 | -0.5357 | -44.7812 | 44.245 | 249.667 |
| 3 | 3 | -1 | 33.8400 | 44.6761 | -10.836 | 250.083 |
| 4 | 4 | 1 | -13.4174 | -42.6960 | 29.279 | 259.000 |
| 5 | 7 | 0 | -26.7903 | -44.1311 | 17.341 | 264.583 |
| 6 | 8 | 0 | -18.8879 | -49.0746 | 30.187 | 264.667 |
| 7 | 9 | 0 | 16.7848 | 43.3144 | -26.530 | 265.042 |
| 8 | 10 | 0 | -20.1586 | -42.1773 | 22.019 | 266.667 |
| 9 | 13 | 0 | 26.2696 | 41.4093 | -15.140 | 270.333 |

mean difference of peak current speeds for
"failure" forecasts is 24.245 cm/s

| false number | event number | dpeak time | model speed(cm/s) | observed speed(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|
| 1 | 5 | 0 | 40.9380 | 24.2013 | 16.737 | 260.750 |
| 2 | 6 | 0 | 48.1970 | 28.1868 | 20.010 | 263.917 |
| 3 | 11 | 0 | 45.2810 | 3.5180 | 41.763 | 270.125 |
| 4 | 12 | 0 | 42.1037 | 24.8829 | 17.221 | 270.208 |

mean difference of peak current speeds for
"false" forecasts is 23.933 cm/s

| Success | Failure | false |
|---|---|---|
| 0 | 9 | 4 |

Table 3.2c. Table2.out for Morgans Point (GBM) forecast, unadjusted

Table 3.2c. Table2.out for Morgans Point (GBM) forecast, unadjusted

Morgans (GBM) ; Model Level 3
Principal Direction : 341.0 degr
critical value (current speed along principal direction)
= 40.00 cm/s
Flood : gain = 1.00, bias = 0.00
Ebb : gain = 1.00, bias = 0.00
Julian start time from control file 245.37

| failure number | event number | dpeak time | model speed(cm/s) | observed speed(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | -10.1934 | -41.1084 | 30.915 | 246.583 |
| 2 | 2 | 0 | -3.9130 | -44.7812 | 40.868 | 249.667 |
| 3 | 3 | -1 | 11.7809 | 44.6761 | -32.895 | 250.083 |
| 4 | 4 | 1 | -10.6474 | -42.6960 | 32.049 | 259.000 |
| 5 | 5 | 0 | -14.7962 | -44.1311 | 29.335 | 264.583 |
| 6 | 6 | 0 | -11.7411 | -49.0746 | 37.334 | 264.667 |
| 7 | 7 | 0 | 4.6701 | 43.3144 | -38.644 | 265.042 |
| 8 | 8 | 0 | -11.3975 | -42.1773 | 30.780 | 266.667 |
| 9 | 9 | 0 | 5.7682 | 41.4093 | -35.641 | 270.333 |

mean difference of peak current speeds for
"failure" forecasts is 34.273 cm/s

| Success | Failure | false |
|---|---|---|
| 0 | 9 | 0 |

Table 3.3a. Table_flood for Port Bolivar (GBM) forecast, unadjusted

September 2000

Bolivar Roads ; Model Level
critical value (current speed along principal direction)
= 65.00 cm/s
Flood : gain = 1.00, bias = 0.00
Julian start time from control file 245.00

| success number | event number | dstart time | delta duration | dpeak time | model spd(cm/s) | observed spd(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 1 | -2 | -1 | 68.5779 | 73.8338 | -5.256 | 248.917 |
| 2 | 7 | 0 | -1 | -1 | 68.1463 | 79.7989 | -11.653 | 250.000 |
| 3 | 8 | 0 | 0 | 0 | 72.4272 | 89.0561 | -16.629 | 251.000 |
| 4 | 9 | -2 | 1 | -2 | 72.4322 | 68.2427 | 4.189 | 252.083 |
| 5 | 10 | 0 | 0 | -1 | 81.8255 | 81.3558 | 0.470 | 253.042 |
| 6 | 11 | 0 | 0 | 0 | 65.2592 | 79.1005 | -13.841 | 253.125 |
| 7 | 12 | -2 | 1 | -1 | 71.0408 | 78.4170 | -7.376 | 254.083 |
| 8 | 14 | -3 | 1 | -3 | 74.2619 | 78.1822 | -3.920 | 255.167 |
| 9 | 15 | 0 | 0 | -1 | 69.4340 | 79.5555 | -10.121 | 256.167 |
| 10 | 22 | 1 | -1 | 1 | 65.2547 | 75.2603 | -10.006 | 261.750 |
| 12 | 24 | 0 | -1 | 0 | 70.7803 | 76.8547 | -6.074 | 262.792 |
| 14 | 26 | -1 | 2 | 0 | 75.6857 | 72.7120 | 2.974 | 263.875 |
| 17 | 29 | 0 | 1 | -1 | 73.4379 | 86.7661 | -13.328 | 264.958 |
| 19 | 31 | -1 | -1 | -1 | 78.6891 | 95.0360 | -16.347 | 266.000 |
| 21 | 33 | -1 | 0 | -1 | 78.2737 | 80.7464 | -2.473 | 267.042 |
| 23 | 35 | 0 | -1 | 0 | 69.6401 | 76.7522 | -7.112 | 273.708 |

mean difference of peak current speeds ( 16 flood events)
for "success" forecasts is 8.236cm/s

| failure number | event number | dpeak time | model speed(cm/s) | observed speed(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|
| 3 | 3 | 0 | 58.4141 | 67.9969 | -9.583 | 246.792 |
| 4 | 4 | -2 | 57.4917 | 71.3969 | -13.905 | 247.917 |
| 8 | 18 | 0 | 54.9804 | 69.1343 | -14.154 | 258.208 |
| 10 | 20 | 0 | 50.2201 | 69.7272 | -19.507 | 259.708 |
| 12 | 37 | 0 | 55.2503 | 81.9747 | -26.724 | 274.708 |

mean difference of peak current speeds ( 5 flood events)
for failure forecasts is 16.775 cm/s

| false number | event number | dpeak time | model speed(cm/s) | observed speed(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|
| 1 | 17 | -1 | 68.3763 | 59.5663 | 8.810 | 257.167 |

mean difference of peak current speeds ( 1 flood events)
for "false" forecasts is 8.810

Table 3.3b. Table_flood for Morgans Point (HSC) forecast, unadjusted

```
Morgans Point  ; Model Level 3
critical value (current speed along principal direction)
=   40.00 cm/s
Flood : gain =   1.00, bias =   0.00
Julian start time from control file  245.37

failure  event  dpeak   model       observed   dspeed  obs peak
number  number time  speed(cm/s)  speed(cm/s) (cm/s)  time(jd)
  3       4     -1     33.8400      44.6761  -10.836  250.083
  5      10      0     16.7848      43.3144  -26.530  265.042
  6      14      0     26.2696      41.4093  -15.140  270.333
mean difference of peak current speeds (  3 flood events)
for failure forecasts is  17.502 cm/s

false   event  dpeak   model       observed   dspeed  obs peak
number  number time  speed(cm/s)  speed(cm/s) (cm/s)  time(jd)
  2       6      0     40.9380      24.2013   16.737  260.750
  3       7      0     48.1970      28.1868   20.010  263.917
  5      12      0     45.2810       3.5180   41.763  270.125
  6      13      0     42.1037      24.8829   17.221  270.208
mean difference of peak current speeds (  4 flood events)
for "false" forecasts is  23.933
```

Table 3.3c. Table_flood for Morgans Point (GBM) forecast, unadjusted

```
Morgans (GBM)  ; Model Level 3
critical value (current speed along principal direction)
=   40.00 cm/s
Flood : gain =   1.00, bias =   0.00
Julian start time from control file  245.37

failure  event  dpeak   model       observed   dspeed  obs peak
number  number time  speed(cm/s)  speed(cm/s) (cm/s)  time(jd)
  3       3     -1     11.7809      44.6761  -32.895  250.083
  7       7      0      4.6701      43.3144  -38.644  265.042
  9       9      0      5.7682      41.4093  -35.641  270.333
mean difference of peak current speeds (  3 flood events)
for failure forecasts is  35.727 cm/s
```

Table 3.4a. Table_ebb for Port Bolivar (GBM) forecast, unadjusted

Observed (PORTS) current data vs. forecast current data,
Event Analysis (Ebb)

September 2000

Bolivar Roads ; Model Level
critical value (current speed along principal direction)
= 65.00 cm/s
Ebb : gain = 1.00, bias = 0.00
Julian start time from control file 245.00

| success number | event number | dstart time | delta duration | dpeak time | model spd(cm/s) | observed spd(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|---|---|
| 11 | 23 | -1 | 1 | 0 | -69.8492 | -65.8917 | -3.958 | 262.542 |
| 13 | 25 | 0 | 2 | 1 | -71.7524 | -75.3568 | 3.604 | 263.500 |
| 15 | 27 | 0 | 0 | 0 | -70.5719 | -69.2458 | -1.326 | 264.500 |
| 16 | 28 | 0 | 1 | 0 | -72.0235 | -75.7092 | 3.686 | 264.583 |
| 18 | 30 | 0 | 0 | 0 | -74.9596 | -67.9850 | -6.975 | 265.583 |
| 20 | 32 | 0 | -2 | -1 | -68.4796 | -80.0462 | 11.567 | 266.708 |
| 22 | 34 | 0 | -1 | -1 | -74.3544 | -77.2811 | 2.927 | 267.708 |

mean difference of peak current speeds ( 7 ebb events)
for "failure" forecasts is 4.863cm/s

| failure number | event number | dpeak time | model speed(cm/s) | observed speed(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | -38.2003 | -65.6662 | 27.466 | 245.042 |
| 2 | 2 | 0 | -55.9332 | -78.1056 | 22.172 | 246.542 |
| 5 | 6 | 0 | -56.2490 | -78.0616 | 21.813 | 249.625 |
| 6 | 13 | -2 | 15.9248 | -81.7844 | 97.709 | 254.833 |
| 7 | 16 | 0 | -60.8915 | -69.6052 | 8.714 | 256.875 |
| 9 | 19 | 0 | -24.3489 | -69.2458 | 44.897 | 258.917 |
| 11 | 36 | 0 | -53.8136 | -69.1284 | 15.315 | 274.458 |

mean difference of peak current speeds ( 7 ebb events)
for failure forecasts is 34.012 cm/s

| false number | event number | dpeak time | model speed(cm/s) | observed speed(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|
| 2 | 21 | 0 | -65.9947 | -53.7106 | -12.284 | 261.500 |

mean difference of peak current speeds ( 1 ebb events)
for "false" forecasts is 12.284

Table 3.4b. Table_ebb for Morgans Point (HSC) forecast, unadjusted

September 2000

Morgans Point ; Model Level
critical value (current speed along principal direction)
= 40.00 cm/s
Ebb : gain = 1.00, bias = 0.00
Julian start time from control file 245.37

| failure number | event number | dpeak time | model speed(cm/s) | observed speed(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | -18.4786 | -41.1084 | 22.630 | 246.583 |
| 2 | 2 | 0 | -0.5357 | -44.7812 | 44.245 | 249.667 |
| 4 | 4 | 1 | -13.4174 | -42.6960 | 29.279 | 259.000 |
| 5 | 7 | 0 | -26.7903 | -44.1311 | 17.341 | 264.583 |
| 6 | 8 | 0 | -18.8879 | -49.0746 | 30.187 | 264.667 |
| 8 | 10 | 0 | -20.1586 | -42.1773 | 22.019 | 266.667 |

mean difference of peak current speeds ( 6 ebb events)
for failure forecasts is 27.617 cm/s

Table 3.4c. Table_ebb for Morgans Point (GBM) forecast, unadjusted

Morgans (GBM) ; Model Level
critical value (current speed along principal direction)
= 40.00 cm/s
Ebb : gain = 1.00, bias = 0.00
Julian start time from control file 245.37

| failure number | event number | dpeak time | model speed(cm/s) | observed speed(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | -10.1934 | -41.1084 | 30.915 | 246.583 |
| 2 | 2 | 0 | -3.9130 | -44.7812 | 40.868 | 249.667 |
| 4 | 4 | 1 | -10.6474 | -42.6960 | 32.049 | 259.000 |
| 5 | 5 | 0 | -14.7962 | -44.1311 | 29.335 | 264.583 |
| 6 | 6 | 0 | -11.7411 | -49.0746 | 37.334 | 264.667 |
| 8 | 8 | 0 | -11.3975 | -42.1773 | 30.780 | 266.667 |

mean difference of peak current speeds ( 6 ebb events)
for failure forecasts is 33.547 cm/s

Table 3.5b. Table2.out for Morgans Point (GBM) forecast, adjusted

Morgans (GBM) ; Model Level 3
Principal Direction : 341.0 degr
critical value (current speed along principal direction)
= 40.00 cm/s
Flood : gain = 1.00, bias = 0.00
Ebb : gain = 2.00, bias = 0.00
Julian start time from control file 245.37

| failure number | event number | dpeak time | model speed(cm/s) | observed speed(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | -20.3867 | -41.1084 | 20.722 | 246.583 |
| 2 | 2 | 0 | -7.8259 | -44.7812 | 36.955 | 249.667 |
| 3 | 3 | -1 | 11.7809 | 44.6761 | -32.895 | 250.083 |
| 4 | 4 | 1 | -21.2948 | -42.6960 | 21.401 | 259.000 |
| 5 | 5 | 0 | -29.5925 | -44.1311 | 14.539 | 264.583 |
| 6 | 6 | 0 | -23.4823 | -49.0746 | 25.592 | 264.667 |
| 7 | 7 | 0 | 4.6701 | 43.3144 | -38.644 | 265.042 |
| 8 | 8 | 0 | -22.7950 | -42.1773 | 19.382 | 266.667 |
| 9 | 9 | 0 | 5.7682 | 41.4093 | -35.641 | 270.333 |

mean difference of peak current speeds for
"failure" forecasts is 27.308 cm/s

| Success | Failure | false |
|---|---|---|
| 0 | 9 | 0 |

Table 3.6a. Table_flood for Morgans Point (HSC) forecast, adjusted

Morgans Point ; Model Level 3
critical value (current speed along principal direction)
= 40.00 cm/s
Flood : gain = 1.00, bias = 0.00
Julian start time from control file 245.37

| failure number | event number | dpeak time | model speed(cm/s) | observed speed(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|
| 3 | 4 | -1 | 33.8400 | 44.6761 | -10.836 | 250.083 |
| 5 | 10 | 0 | 16.7848 | 43.3144 | -26.530 | 265.042 |
| 6 | 14 | 0 | 26.2696 | 41.4093 | -15.140 | 270.333 |

mean difference of peak current speeds ( 3 flood events)
for failure forecasts is 17.502 cm/s

| false number | event number | dpeak time | model speed(cm/s) | observed speed(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|
| 2 | 6 | 0 | 40.9380 | 24.2013 | 16.737 | 260.750 |
| 3 | 7 | 0 | 48.1970 | 28.1868 | 20.010 | 263.917 |
| 5 | 12 | 0 | 45.2810 | 3.5180 | 41.763 | 270.125 |
| 6 | 13 | 0 | 42.1037 | 24.8829 | 17.221 | 270.208 |

mean difference of peak current speeds ( 4 flood events)
for "false" forecasts is 23.933

Table 3.6b. Table_flood for Morgans Point (GBM) forecast, adjusted

Morgans (GBM) ; Model Level 3
critical value (current speed along principal direction)
= 40.00 cm/s
Flood : gain = 1.00, bias = 0.00
Julian start time from control file 245.37

| failure number | event number | dpeak time | model speed(cm/s) | observed speed(cm/s) | dspeed (cm/s) | obs peak time(jd) |
|---|---|---|---|---|---|---|
| 3 | 3 | -1 | 11.7809 | 44.6761 | -32.895 | 250.083 |
| 7 | 7 | 0 | 4.6701 | 43.3144 | -38.644 | 265.042 |
| 9 | 9 | 0 | 5.7682 | 41.4093 | -35.641 | 270.333 |

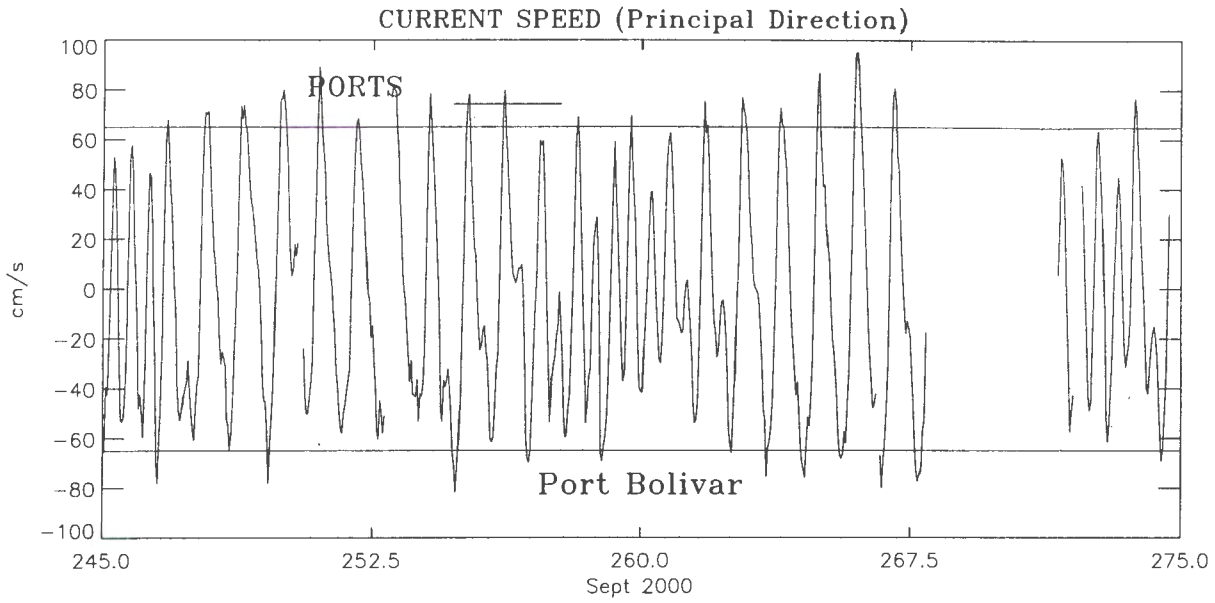
mean difference of peak current speeds ( 3 flood events)
for failure forecasts is 35.727 cm/s

Table 3.7a. Table_ebb for Morgans Point (HSC) forecast, adjusted

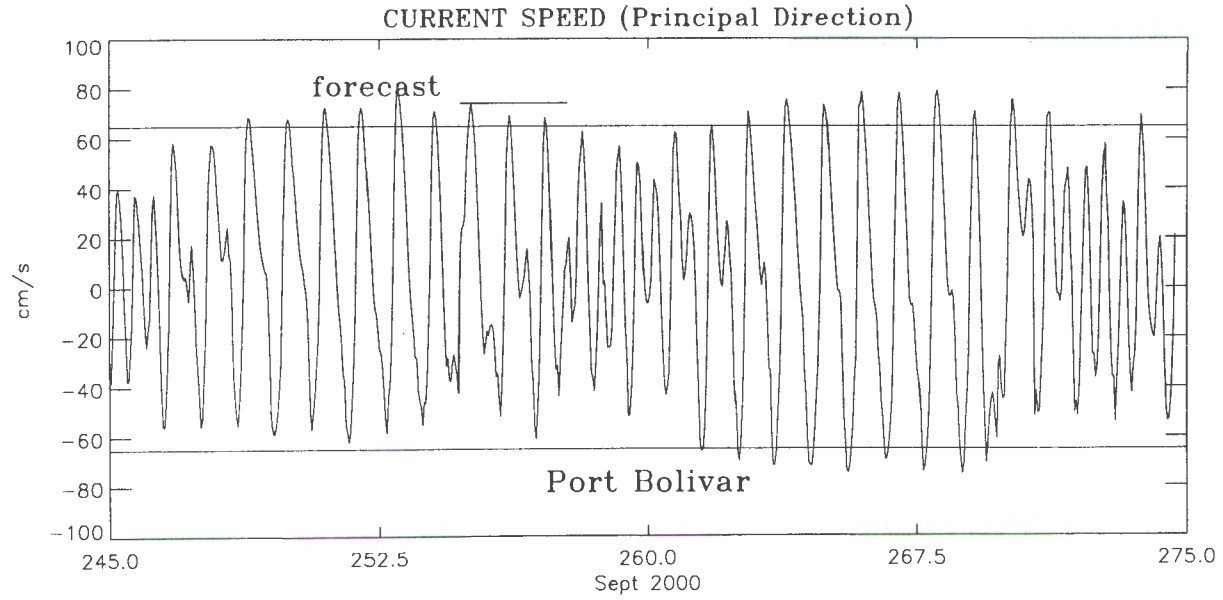
```
 Morgans Point  ; Model Level 3
 critical value (current speed along principal direction)
 =   40.00 cm/s
 Ebb : gain =   2.00, bias =   0.00
 Julian start time from control file  245.37

success  event  dstart  delta    dpeak   model     observed  dspeed  obs peak
 number  number  time  duration  time  spd(cm/s)  spd(cm/s) (cm/s)  time(jd)
   1       9      -3       2      -2   -53.5807   -49.0746   -4.506  264.667
   2      11       0       0       0   -40.3171   -42.1773    1.860  266.667
 mean difference of peak current speeds (  2 ebb events)
 for "failure" forecasts is   3.183cm/s

failure  event  dpeak   model       observed   dspeed  obs peak
 number  number time  speed(cm/s)  speed(cm/s) (cm/s)  time(jd)
   1       2      0    -36.9572     -41.1084    4.151  246.583
   2       3      0     -1.0715     -44.7812   43.710  249.667
   4       5      1    -26.8347     -42.6960   15.861  259.000
 mean difference of peak current speeds (  3 ebb events)
 for failure forecasts is  21.241 cm/s

 false   event  dpeak   model       observed   dspeed  obs peak
 number  number time  speed(cm/s)  speed(cm/s) (cm/s)  time(jd)
   1       1      0    -43.6220     -4.6212  -39.001  245.542
   4       8      0    -41.8298    -20.5811  -21.249  264.375
 mean difference of peak current speeds (  2 ebb events)
 for "false" forecasts is  30.125
```

Table 3.7b. Table_ebb for Morgans Point (GBM) forecast, adjusted

```
 Morgans (GBM)  ; Model Level 3
 critical value (current speed along principal direction)
 =   40.00 cm/s
 Ebb : gain =   2.00, bias =   0.00
 Julian start time from control file  245.37

failure  event  dpeak   model       observed   dspeed  obs peak
 number  number time  speed(cm/s)  speed(cm/s) (cm/s)  time(jd)
   1       1      0    -20.3867     -41.1084   20.722  246.583
   2       2      0     -7.8259     -44.7812   36.955  249.667
   4       4      1    -21.2948     -42.6960   21.401  259.000
   5       5      0    -29.5925     -44.1311   14.539  264.583
   6       6      0    -23.4823     -49.0746   25.592  264.667
   8       8      0    -22.7950     -42.1773   19.382  266.667
 mean difference of peak current speeds (  6 ebb events)
 for failure forecasts is  23.099 cm/s
```




Observations




unadjusted forecast

Figure 3.1. Event Analysis Plots at Port Bolivar for September 2000

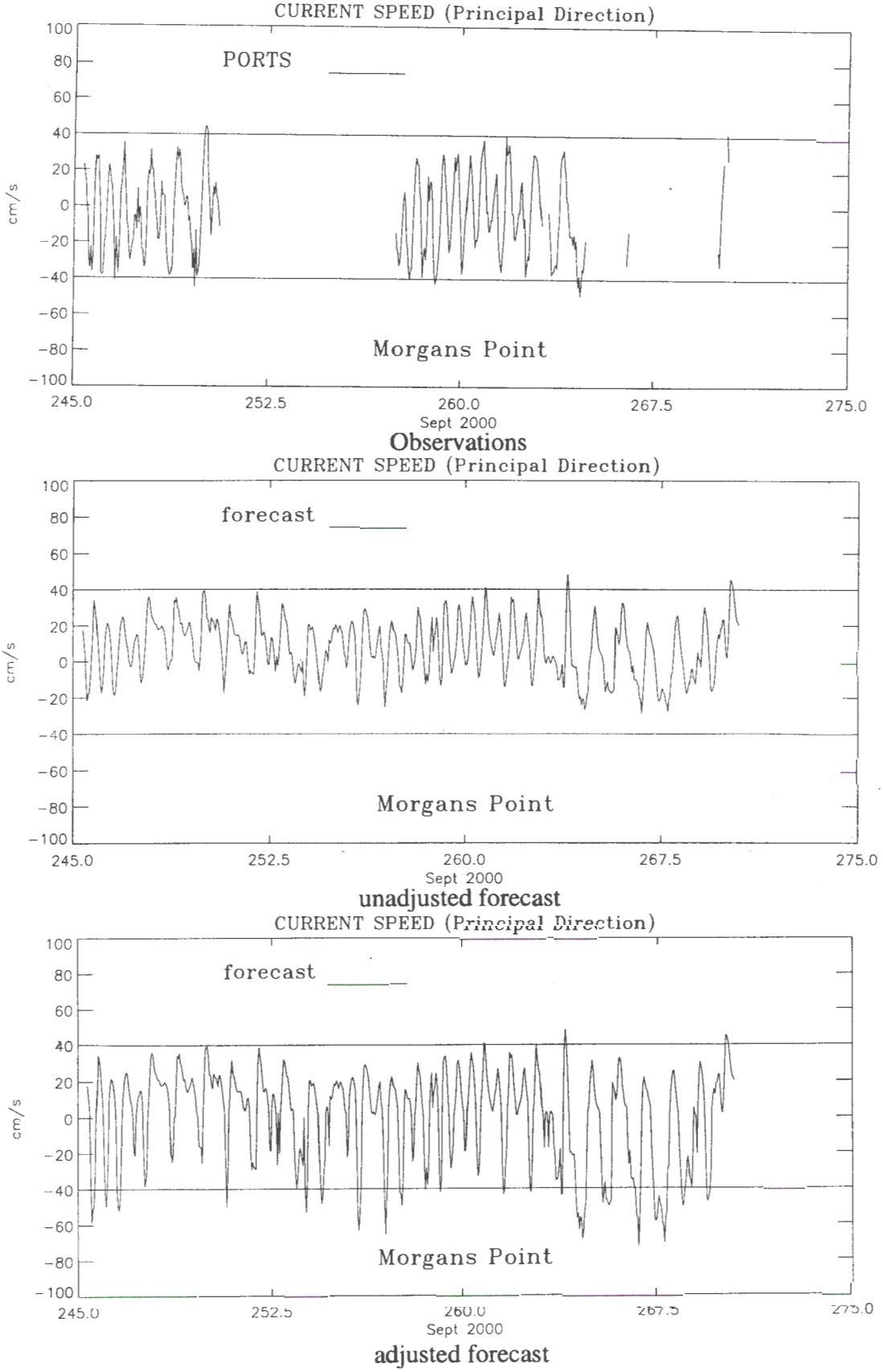



Figure 3.2. Event Analysis Plots at Morgans Point (HSC forecast) for September 2000

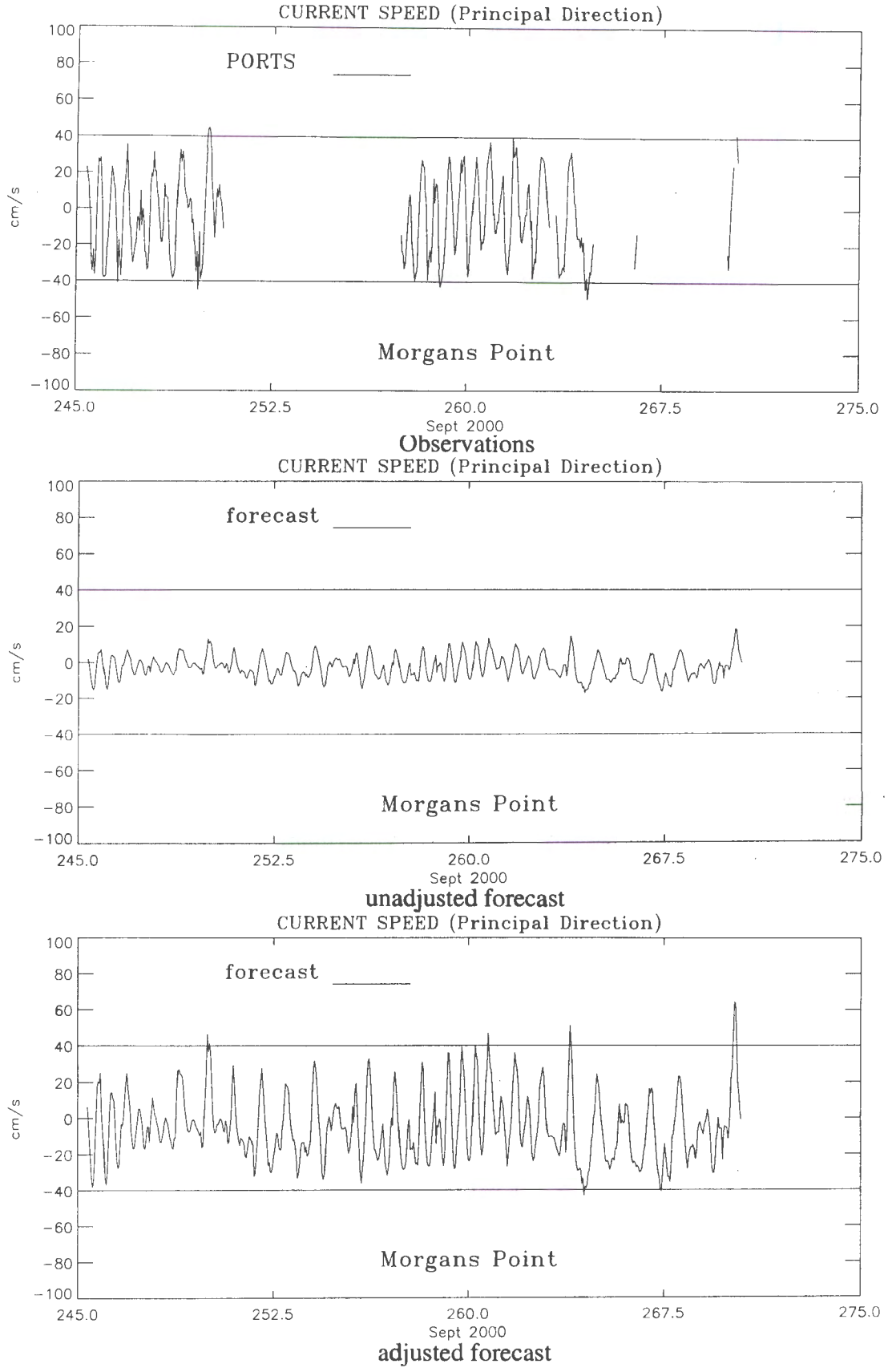



Figure 3.3. Event Analysis Plots at Morgans Point (GBM forecast) for September 2000

## 4. OPERATIONAL USE AND ENHANCEMENTS

In the nowcast/forecast system operational environment, it will be necessary to evaluate and assess the quality of the current speed forecasts for major ebb and flood events with the associated potential for vessel collision and groundings and oil spills. While the majority of the formal acceptance statistics (NOS, 1999) will be met prior to operations, the ability of the nowcast/forecast system to forecast extreme current events will need to be assessed on an ongoing event by event basis.

It is envisioned that the operational institution will perform the event evaluation on a monthly basis and that this evaluation will be included in the monthly nowcast/forecast system evaluation bulletins. At the end of each year, a yearly summary will be performed based on these monthly bulletins. As a result, programs similar to these documented herein will be required. It is hoped that these programs may be adapted for the final implementation programs.

The specification of critical values to define critical flood and ebb current events may cause failures and false alarms with current speeds which are not significantly different than observed current speeds. As a result, Program Match.evnt_crnt.f provides table2.out tables for failures and false alarms as well as for successes. In these event tables, the differences between the model and observed peak current speeds for each event is given.

As noted herein, separate flood and ebb adjustment factors for both bias and gain are now provided. However, time lags have not yet been considered. The treatment of time lags deserves further study as a means to further improve principal component direction current forecasts.

## ACKNOWLEDGMENTS

Dr. Eddie H. Shih, NOS/COOPS, and formally of NOS/CSDL is especially acknowledged for providing essential design inputs in the development of the program set, which was initially used to assess the East Coast Ocean Forecast System and the NWS/TDL Extratropical Storm Surge Model forecasts at NOS/NWLON water level stations along the East Coast.

## REFERENCES


Bethem, T. D., and H. R. Frey, 1991: Operational physical oceanographic real-time data dissemination, *Proceedings, IEEE Oceans 91*, 865 - 867.

Frey, H. R., 1991: Physical oceanographic real-time systems for operational use. *Proceedings, IEEE Oceans 91*, 855 - 858.

NOS, 1999: NOS procedures for developing and implementing operational nowcast and forecast systems for PORTS, *NOAA Technical Report NOS/CO-OPS 20*, Silver Spring, MD.

Richardson, P. H. and R.A. Schmalz, 1999: Evaluation of forecast inputs: supplemental program documentation. NOAA, National Ocean Service, Coast Survey Development Laboratory, *NOS/CSDL Galveston Bay Nowcast/Forecast System Report Series, User Information Report 1 ,unpublished working report*, Silver Spring, MD.

Schmalz, R. A. and P.H. Richardson, 1996: Preliminary nowcast/forecast requirements for Galveston Bay. NOAA, National Ocean Service, Coast Survey Development Laboratory, *NOS/CSDL Galveston Bay Nowcast/Forecast System Report Series, System Requirements Report 1, unpublished working report*, Silver Spring, MD.

Schmalz, R. A. and P.H. Richardson, 1998a: Development of a prototype nowcasting/forecasting System for Galveston Bay: Hindcast Studies. NOAA, National Ocean Service, Coast Survey Development Laboratory, *NOS/CSDL Galveston Bay Nowcast/Forecast System Report Series, Technical Development Report 1, unpublished working report*, Silver Spring, MD.

Schmalz, R. A. and P.H. Richardson, 1998b: Development of a prototype nowcasting/forecasting System for Galveston Bay: Nowcast/Forecast Studies. NOAA, National Ocean Service, Coast Survey Development Laboratory, *NOS/CSDL Galveston Bay Nowcast/Forecast System Report Series, Technical Development Report 2, unpublished working report*, Silver Spring, MD.

Williams, R. G., H. R. Frey, T. Bethem, 1990. Houston Ship Channel/Galveston Bay current prediction quality assurance miniproject, *NOAA Technical Memorandum NOS OMA 53*, Rockville, MD.

## APPENDIX A. JCL AND CONTROL FILES

JCL and control files for each of the seven programs are provided below as shown in Table 3.1.

reform.jcl

```
f77 reform_ports.f calc_uv.f calcjd.f -o reform_ports

rm *.o

reform_ports < reform.morgn.n > out
```

reform.morgn.n

```
0     idebug
305.998
morgn.nov00.raw
morgn.nov00.ex
```

regap.jcl

```
f77 regap.f calc_uv.f -o regap
rm *.o

regap < regap.nov00.n > out
```

regap.morgn.n

```
 8
morgn.dec00.spr10
morgn.dec00.fix
```

reform2.jcl

```
# f77 reform2.f calc_uv.f -o reform
# rm *.o

  reform < reform2.nov00P.n > out
```

reform2.n

```
0     debug option
morgn.dec00.fix
morgn.dec00.obs
 336.0 start time
 367.0 end time
```

readcurr.jcl

```
# f77 read_NF.curr.f calcuvnew.f -o readcurr

# rm *.o


  readcurr < readcrr_sep00.n > out
# readcurr < readcrr_oct00.n > out
# readcurr < readcrr_nov00.n > out
# readcurr < readcrr_dec00.n > out
# readcurr < readcrr_jan01.n > out
# readcurr < readcrr_feb01.n > out
# readcurr < readcrr_mar01.n > out

# rm *.frc*
  rm *.now*
```

readcrr_sep00.n

```
245
0     idebug
9
bolvr.now3
10
bolvr.frc3
11
morg2.now3
12
morg2.frc3
13
morgn.now3
14
morgn.frc3
15
bolv2.now3
16
bolv2.frc1
 3    model level
30    nfiledays
 0    nmissgbm
 0    nmisshsc
/opseadisk2/HGOPS.dt/gbm2000/200009/20000901/uvb.245.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000901/uvb.245.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000902/uvb.246.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000902/uvb.246.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000903/uvb.247.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000903/uvb.247.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000904/uvb.248.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000904/uvb.248.00z
```

readcrr_sep00.n (continued)

```
/opseadisk2/HGOPS.dt/gbm2000/200009/20000905/uvb.249.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000905/uvb.249.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000906/uvb.250.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000906/uvb.250.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000907/uvb.251.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000907/uvb.251.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000908/uvb.252.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000908/uvb.252.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000909/uvb.253.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000909/uvb.253.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000910/uvb.254.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000910/uvb.254.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000911/uvb.255.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000911/uvb.255.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000912/uvb.256.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000912/uvb.256.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000913/uvb.257.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000913/uvb.257.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000914/uvb.258.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000914/uvb.258.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000915/uvb.259.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000915/uvb.259.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000916/uvb.260.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000916/uvb.260.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000917/uvb.261.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000917/uvb.261.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000918/uvb.262.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000918/uvb.262.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000919/uvb.263.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000919/uvb.263.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000920/uvb.264.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000920/uvb.264.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000921/uvb.265.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000921/uvb.265.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000922/uvb.266.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000922/uvb.266.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000923/uvb.267.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000923/uvb.267.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000924/uvb.268.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000924/uvb.268.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000925/uvb.269.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000925/uvb.269.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000926/uvb.270.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000926/uvb.270.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000927/uvb.271.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000927/uvb.271.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000928/uvb.272.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000928/uvb.272.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000929/uvb.273.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000929/uvb.273.00z
/opseadisk2/HGOPS.dt/gbm2000/200009/20000930/uvb.274.00z
/opseadisk2/HGOPS.dt/hsc2000/200009/20000930/uvb.274.00z
```

match.jcl

```
# f77 match.evnt_crnt.f calcjd.f prdirection.f timehi.f jdgreg.f -o match

# rm *.o


# match < match_currN.feb01.n > out
# match < match_currP.sep00.n > out
# match < match_curr.jan01.n > out
# match < match_morgn.feb01.n > out
# match < match_morg2.sep00.n > out
# match < match_bolvr.feb01.n > out
  match < match_docu2.sep00.n > out


# rm out
# rm table2.out
# rm table_ebb
# rm table_flood
# rm pltO*
# rm pltN*.bolvr.*
# rm pltF*
```

match_crr2.sep00.n

```
0          idebug
September 2000
Observed (PORTS) current data vs. forecast current data,
Event Analysis
275.000    endjd
forecast
 5         number of station comparisons
Bolivar Roads
245.00     startjd
/usr/people/philr/galves/NF_eval/currents/ports/bolvr.sep00.obs
/usr/people/philr/galves/NF_eval/currents/forcast/bolvr.frc3.sep00
  65.0     crlevel - critical value
 322.0     princpal direction
 1.00 1.00 gain, flood and ebb
-0.0 -0.0  bias, flood and ebb
Morgans Point
245.37     startjd
/usr/people/philr/galves/NF_eval/currents/ports/morgn.sep00.obs
/usr/people/philr/galves/NF_eval/currents/forcast/morgn.frc3.sep00
  40.0     crlevel
 341.0     principal direction
 1.00  1.00 gain, flood and ebb
 0.0 -00.0  bias, flood and ebb
```

match_crr.sep00.n (continued)

```
Morgans (GBM)
245.37    startjd
/usr/people/philr/galves/NF_eval/currents/ports/morgn.sep00.obs
/usr/people/philr/galves/NF_eval/currents/forcast/morg2.frc3.sep00
  40.0    crlevel
 341.0    principal direction
 1.00  1.00 gain, flood and ebb
 0.0  -0.0  bias, flood and ebb
Morgans Point
245.37    startjd
/usr/people/philr/galves/NF_eval/currents/ports/morgn.sep00.obs
/usr/people/philr/galves/NF_eval/currents/forcast/morgn.frc3.sep00
  40.0
 341.0
 1.00  2.00 gain, flood and ebb
 0.0 -00.0  bias, flood and ebb
Morgans (GBM)
245.37
/usr/people/philr/galves/NF_eval/currents/ports/morgn.sep00.obs
/usr/people/philr/galves/NF_eval/currents/forcast/morg2.frc3.sep00
  40.0
 341.0
 1.00  2.00 gain, flood and ebb
 0.0   0.0  bias, flood and ebb
no*       plot option
plotFl.bolvr.sep00
plotO.bolvr.sep00
plotFl.morgn.sep00
plotO.morgn.sep00
```

IDL< curr.prdir.pro

cntrl.morgn_sep00

```
ps
landscape
1        idebug
!17Morgans Point!X
 245.00  tmin
 244.75  start time
 275.00  tmax
 274.75  end time
2000
1        number of curves
!17CURRENT SPEED (Principal Direction)!X
!17forecast!X
/usr/people/philr/galves/NF_eval/currents/sa.current/plotfiles/forc/pltF3.morgn.sep00
-100.00 100.00 10  yrange, and number of tick marks
Sept 2000
 40.00   crlevel
```

IDL< curr.multcurv.pro

c.morgn_sep00

```
ps
landscape
0        idebug
!17Morgans Point!X
 245.00  tmin
 245.00  start time
 275.00  tmax
 274.71  end time
5        number of curves
!17MORGANS POINT CURRENT SPEED (Principal Direction)!X
!17observed!X
observed
/usr/people/philr/galves/NF_eval/currents/sa.current/plotfiles/obs/pltO1.morgn.sep00
!17nowcast!X
nowcast
/usr/people/philr/galves/NF_eval/currents/sa.current/plotfiles/nowc/pltN1.morgn.sep00
!17predicted!X
predicted
/usr/people/philr/galves/NF_eval/currents/sa.current/plotfiles/pred/plotP.morgn.sep00
!17forecast!X
forcast
/usr/people/philr/galves/NF_eval/currents/sa.current/plotfiles/forc/pltF1.morg2.sep00
!17adj fcst!X
forcast
/usr/people/philr/galves/NF_eval/currents/sa.current/plotfiles/forc/pltF1.gbm1.sep00
-100.0 100.0 8  yrange, and number of tick marks
September 2000
 40.00   crlevel
```