# Chapter 41
# Graphics Libraries - Simple Dislin Usage


*Modern data graphics can do much more than simply substitute for small statistical tables. At their best, graphics are instruments for reasoning about quantitative information. Often the most effective way to describe, explore and summarise a set of numbers — even a large set — is to look at pictures of those numbers.*

Edward R Tufte, The Visual Display of Quantitative Information.

*A picture paints a thousand words.*

Reportedly first used by Frederick R. Barnard in Printer's Ink (December, 1921), while commenting that graphics can tell a story as effectively as a large amount of descriptive text.

## 41.1 Introduction

In science and engineering graphics are essential part of the presentation of information.

A graphics library is generally a set of routines that can be called from one or more programming languages to help in the display of graphical output to a screen or monitor with the option normally of targetting a hard copy device.

Our resource file

```
https://www.fortranplus.co.uk/
fortran-information/
```

provides details of some of the graphics libraries available.

We will be using the Dislin library in our examples.

## 41.2 The Dislin Graphics Library

This is the dislin home page.



I. Chivers and J. Sleightholme, *Introduction to Programming with Fortran*,
https://doi.org/10.1007/978-3-319-75502-1_41

```
http://www.mps.mpg.de/dislin/
```

Here is a description of the software from the above page.

- Dislin is a high-level and easy to use plotting library for displaying data as curves, bar graphs, pie charts, 3D-colour plots, surfaces, contours, and maps. The library contains about 500 plotting and parameter setting routines. The approach used is to have only a few graphics routines with short parameter lists. A large variety of parameter setting routines can then be used to create customized graphics. Several output formats are supported such as X11, VGA, PostScript, PDF, SVG, CGM, HPGL, TIFF, GIF, PNG and BMP. Dislin is available for the programming languages C, Fortran 77, Fortran 90, Perl, Python and Java.

## 41.3 Example 1: Using Dislin to Plot Amdahl's Law Graph 1 – 8 Processors or Cores

Here is the source code for this program.

```
program ch41_dislin_01
  use dislin
  implicit none
  integer :: i, j
! Total number of processors and hence data ! points
  integer, parameter :: nprocs = 8
! Number of percentage values from
! 10% -> 90% 9
! 95% 1
! Total 10
  integer, parameter :: nn = 10
  real, dimension (nn) :: pp = (/ 0.1, 0.2, 0.3, &
    0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95 /)
  real, dimension (nprocs) :: x
  real, dimension (nprocs) :: y
  real, dimension (nprocs, nn) :: ydata
  integer :: nx
  integer :: ny
  character *30 cbuf

  do i = 1, nprocs
    x(i) = real(i)
  end do
! Amdahl calculations. Store in 2 d array and
! then
```

```
! assign to 1 d array for plotting.
  do i = 1, nprocs
    do j = 1, nn
      ydata(i, j) = 1/((1-pp(j))+pp(j)/i)
    end do
  end do
! Write the data to a file for verification
! purposes
  open (unit=10, file='amdahl_table_08.txt')
  do i = 1, nprocs
    write (unit=10, fmt=100) x(i), &
      ydata(i, 1:nn)
100 format (11(f7.2,2x))
  end do
  close (10)
  call disini
  call complx
  call axspos(450, 1800)
  call axslen(2200, 1400)
  call name('Number of processors', 'x')
  call name('Speed up', 'y')
  call titlin('Plot of Amdahls Law', 1)
  call titlin('8 Processors', 3)
  call labdig(-1, 'x')
  call ticks(10, 'xy')
  call graf(1.0, 8.0, 1.0, 1.0, 1.0, 7.0, 1.0, &
    1.0)
  call title
  call xaxgit
  call chncrv('line')
! Plot the curves. Copy from 2 d array to 1 d
! array
! before the call to curve.
  do i = 1, nn
    y = ydata(1:nprocs, i)
    call curve(x, y, nprocs)
  end do
  call legini(cbuf, 10, 3)
! Coordinates of the start of the legend
! for the curves.
  nx = 500
  ny = 450
  call legpos(nx, ny)
  call leglin(cbuf, '10%', 1)
  call leglin(cbuf, '20%', 2)
```

```
  call leglin(cbuf, '30%', 3)
  call leglin(cbuf, '40%', 4)
  call leglin(cbuf, '50%', 5)
  call leglin(cbuf, '60%', 6)
  call leglin(cbuf, '70%', 7)
  call leglin(cbuf, '80%', 8)
  call leglin(cbuf, '90%', 9)
  call leglin(cbuf, '95%', 10)
  call legtit('legend')
  call legend(cbuf, 3)
  call disfin
end program ch41_dislin_01
```

## 41.4 Example 2: Using Dislin to Plot Amdahl's Law Graph 2 – 64 Processors or Cores

Here is the source code for this program.

```
program ch41_dislin_02
  use dislin
  implicit none
  integer :: i, j
! Total number of processors and hence data
! points
  integer, parameter :: nprocs = 64
! Number of percentage values from
! 10% -> 90% 9
! 95% 1
! Total 10
  integer, parameter :: nn = 10
  real, dimension (nn) :: pp = (/ 0.1, 0.2, 0.3, &
    0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95 /)
  real, dimension (nprocs) :: x
  real, dimension (nprocs) :: y
  real, dimension (nprocs, nn) :: ydata
  integer :: nx
  integer :: ny
  character *30 cbuf

  do i = 1, nprocs
    x(i) = real(i)
  end do
! Amdahl calculations. Store in 2 d array and
```

```
! then
! assign to 1 d array for plotting.
  do i = 1, nprocs
    do j = 1, nn
      ydata(i, j) = 1/((1-pp(j))+pp(j)/i)
    end do
  end do
! Write the data to a file for verification
! purposes
  open (unit=10, file='amdahl_table_08.txt')
  do i = 1, nprocs
    write (unit=10, fmt=100) x(i), &
      ydata(i, 1:nn)
100 format (11(f7.2,2x))
  end do
  close (10)
  call disini
  call complx
  call axspos(450, 1800)
  call axslen(2200, 1400)
  call name('Number of processors', 'x')
  call name('Speed up', 'y')
  call titlin('Plot of Amdahls Law', 1)
  call titlin('8 Processors', 3)
  call labdig(-1, 'x')
  call ticks(10, 'xy')
  call graf(1.0, 8.0, 1.0, 1.0, 1.0, 7.0, 1.0, &
    1.0)
  call title
  call xaxgit
  call chncrv('line')
! Plot the curves. Copy from 2 d array to 1 d
! array
! before the call to curve.
  do i = 1, nn
    y = ydata(1:nprocs, i)
    call curve(x, y, nprocs)
  end do
  call legini(cbuf, 10, 3)
! Coordinates of the start of the legend
! for the curves.
  nx = 500
  ny = 450
  call legpos(nx, ny)
  call leglin(cbuf, '10%', 1)
```

```
  call leglin(cbuf, '20%', 2)
  call leglin(cbuf, '30%', 3)
  call leglin(cbuf, '40%', 4)
  call leglin(cbuf, '50%', 5)
  call leglin(cbuf, '60%', 6)
  call leglin(cbuf, '70%', 7)
  call leglin(cbuf, '80%', 8)
  call leglin(cbuf, '90%', 9)
  call leglin(cbuf, '95%', 10)
  call legtit('legend')
  call legend(cbuf, 3)
  call disfin
end program ch41_dislin_02
```

It is similar to the previous example.

## 41.5 Example 3: Using Dislin to Plot Gustafson's Law Graph 1 – 64 Processors or Cores

Here is the source code for this program.

```
program ch41_dislin_03
  use dislin
  implicit none
  integer :: i, j
! Total number of processors and hence data
! points
  integer, parameter :: nprocs = 64
! Number of percentage values from
! 10% -> 90% 9
! 95% 1
! Total 10
  integer, parameter :: nn = 10
  real, dimension (nn) :: pp = (/ 0.1, 0.2, 0.3, &
    0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95 /)
  real, dimension (nprocs) :: x
  real, dimension (nprocs) :: y
  real, dimension (nprocs, nn) :: ydata
  integer :: nx
  integer :: ny
  character *30 cbuf

  do i = 1, nprocs
```

```
      x(i) = real(i)
    end do
  ! gustafson calculations. Store in 2 d array and
  ! then
  ! assign to 1 d array for plotting.
    do i = 1, nprocs
      do j = 1, nn
        ydata(i, j) = i - (1-pp(j))*(i-1)
      end do
    end do
  ! Write the data to a file for verification
  ! purposes
    open (unit=10, file='gustafson_table.txt')
    do i = 1, nprocs
      write (unit=10, fmt=100) x(i), &
        ydata(i, 1:nn)
  100 format (11(f7.2,2x))
    end do
    close (10)
    call disini
    call complx
    call axspos(450, 1800)
    call axslen(2200, 1400)
    call name('Number of processors', 'x')
    call name('Speed up', 'y')
    call titlin('Plot of Gustafsons Law', 1)
    call titlin('64 Processors', 3)
    call labdig(-1, 'x')
    call ticks(10, 'xy')
    call graf(0.0, 64.0, 0.0, 10.0, 0.0, 70.0, &
      0.0, 10.0)
    call title
    call xaxgit
    call chncrv('line')
  ! Plot the curves. Copy from 2 d array to 1 d
  ! array
  ! before the call to curve.
    do i = 1, nn
      y = ydata(1:nprocs, i)
      call curve(x, y, nprocs)
    end do
    call legini(cbuf, 10, 3)
  ! Coordinates of the start of the legend
  ! for the curves.
    nx = 500
```

```
  ny = 450
  call legpos(nx, ny)
  call leglin(cbuf, '10%', 1)
  call leglin(cbuf, '20%', 2)
  call leglin(cbuf, '30%', 3)
  call leglin(cbuf, '40%', 4)
  call leglin(cbuf, '50%', 5)
  call leglin(cbuf, '60%', 6)
  call leglin(cbuf, '70%', 7)
  call leglin(cbuf, '80%', 8)
  call leglin(cbuf, '90%', 9)
  call leglin(cbuf, '95%', 10)
  call legtit('legend')
  call legend(cbuf, 3)
  call disfin
end program ch41_dislin_03
```

It is similar to the first example.

## 41.6 Example 4: Using Dislin to Plot Tsunami Events

Here is the source code for this program.

```
program ch41_dislin_04
  use dislin
  logical :: trial, screen
  real :: long, lat

  screen = .false.
  trial = .false.

! read in the tsunami data

  call datain(trial)

! I now have all the tsunami data latitude and
! longitude values read in to the arrays in the
! tsunam common block.

  iproj = 1
  lat = 0.0
  long = 180.0
```

```
      nreg = 0

    ! dislin initialisation routines and setting of
    ! some basic components
    ! of the plot. these are based on two sample
    ! dislin programs.

    ! initialise dislin

      call disini

    ! choose font

      call psfont('times-roman')

    ! determines the position of an axis system.
    ! the lower left corner of the axis system

      call axspos(400, 1850)

    ! the size of the axis system
    ! are the length and height of an axis system in
    ! plot coordinates. the default
    ! values are set to 2/3 of the page length and ! height.

      call axslen(2400, 1400)

    ! define axis title

      call name('longitude', 'x')

    ! define axis title

      call name('latitude', 'y')

    ! this routine plots a title over an axis
    ! system.

      call titlin('plot of 3034 tsunami events ', 3)

    ! determines which label types will be plotted
    ! on an axis.
    ! map defines geographical labels which are
    ! plotted as non negative floating-point
    ! numbers with the following characters 'w', ! 'e', 'n'
```

```
 and 's'.

  call labels('map', 'xy')

! plots a geographical axis system.

  call grafmp(-180., 180., -180., 90., -90., &
    90., -90., 30.)

! the statement call gridmp (i, j) overlays an
! axis system with a longitude
! and latitude grid where i and j are the number
! of grid lines between  labels in
! the x- and y-direction.

  call gridmp(1, 1)

! the routine world plots coastlines and lakes.

  call world

! the angle and height of the characters can be
! changed with the routines
! angle and height.

  call height(50)

! this routine plots a title over an axis
! system.
! the title may contain up to four lines of text
! designated
! with titlin.

  call title

! this is a call to the routine that actually
! plots each event.

  call plotem(trial, nreg)

! disfin terminates dislin and prints a message
! on the screen.
! the level is set back to 0.

  call disfin
```

```
end program ch41_dislin_04

subroutine datain(trial)
  common /tsunam/reg0la(378), reg0lo(378), &
    reg1la(206), reg1lo(206), reg2la(41), &
    reg2lo(41), reg3la(54), reg3lo(54), &
    reg4la(60), reg4lo(60), reg5la(1540), &
    reg5lo(1540), reg6la(80), reg6lo(80), &
    reg7la(144), reg7lo(144), reg8la(245), &
    reg8lo(245), reg9la(285), reg9lo(285)

  logical :: trial
  character (80) :: filnam

  if (trial) then
    print *, ' entering data input phase'
  end if
  filnam = 'tsunami.txt'
  open (unit=50, file=filnam, err=100, &
    status='old')
  go to 110
100 print *, ' error opening data file'
  print *, ' program terminates'
  stop
110 do i = 1, 378
    read (unit=50, fmt=120) reg0la(i), reg0lo(i)
  end do
100 format (1x, f7.2, 2x, f7.2)
  do i = 1, 206
    read (unit=50, fmt=120) reg1la(i), reg1lo(i)
  end do
  do i = 1, 41
    read (unit=50, fmt=120) reg2la(i), reg2lo(i)
  end do
  do i = 1, 54
    read (unit=50, fmt=120) reg3la(i), reg3lo(i)
  end do
  do i = 1, 60
    read (unit=50, fmt=120) reg4la(i), reg4lo(i)
  end do
  do i = 1, 1540
    read (unit=50, fmt=120) reg5la(i), reg5lo(i)
  end do
  do i = 1, 80
```

```
    read (unit=50, fmt=120) reg6la(i), reg6lo(i)
  end do
  do i = 1, 144
    read (unit=50, fmt=120) reg7la(i), reg7lo(i)
  end do
  do i = 1, 245
    read (unit=50, fmt=120) reg8la(i), reg8lo(i)
  end do
  do i = 1, 285
    read (unit=50, fmt=120) reg9la(i), reg9lo(i)
  end do
  if (trial) then
    do i = 1, 10
      print *, reg0la(i), '   ', reg0lo(i)
    end do
    print *, ' exiting data input phase'
    read *, dummy
  end if
end subroutine datain

subroutine plotem(trial, nreg)
  use dislin
  common /tsunam/reg0la(378), reg0lo(378), &
    reg1la(206), reg1lo(206), reg2la(41), &
    reg2lo(41), reg3la(54), reg3lo(54), &
    reg4la(60), reg4lo(60), reg5la(1540), &
    reg5lo(1540), reg6la(80), reg6lo(80), &
    reg7la(144), reg7lo(144), reg8la(245), &
    reg8lo(245), reg9la(285), reg9lo(285)

! this subroutine plots all of the tsunamis onto
! the map as coloured
! points, with a different colour per region. i
! have chosen
! a dot size of 1 mm, and step through the
! colour pallette.
! the default may not be appropriate.

  logical :: trial
  integer :: nreg
  integer :: kolour = 10
  data dwidth/1.0/

  if (trial) then
    dwidth = 5.0
```

```
  print *, ' entering plot points'
end if
call incmrk(-1)
if (nreg==0) then
  call setclr(kolour)
  call curvmp(reg0lo, reg0la, 378)
  kolour = kolour + 30
  call setclr(kolour)
  call curvmp(reg1lo, reg1la, 206)
  kolour = kolour + 30
  call setclr(kolour)
  call curvmp(reg2lo, reg2la, 41)
  kolour = kolour + 30
  call setclr(kolour)
  call curvmp(reg3lo, reg3la, 54)
  kolour = kolour + 30
  call setclr(kolour)
  call curvmp(reg4lo, reg4la, 60)
  kolour = kolour + 30
  call setclr(kolour)
  call curvmp(reg5lo, reg5la, 1540)
  kolour = kolour + 30
  call setclr(kolour)
  call curvmp(reg6lo, reg6la, 80)
  kolour = kolour + 30
  call setclr(kolour)
  call curvmp(reg7lo, reg7la, 144)
  kolour = kolour + 30
  call setclr(kolour)
  call curvmp(reg8lo, reg8la, 245)
  kolour = kolour + 30
  call setclr(kolour)
  call curvmp(reg9lo, reg9la, 285)
else if (nreg==1) then
  kolour = 10
  call setclr(kolour)
  call curvmp(reg0lo, reg0la, 378)
else if (nreg==2) then
  kolour = 20
  call setclr(kolour)
  call curvmp(reg1lo, reg1la, 206)
else if (nreg==3) then
  kolour = 30
  call setclr(kolour)
  call curvmp(reg2lo, reg2la, 41)
```

```
    else if (nreg==4) then
      kolour = 40
      call setclr(kolour)
      call curvmp(reg3lo, reg3la, 54)
    else if (nreg==5) then
      kolour = 50
      call setclr(kolour)
      call curvmp(reg4lo, reg4la, 60)
    else if (nreg==6) then
      kolour = 60
      call setclr(kolour)
      call curvmp(reg5lo, reg5la, 1540)
    else if (nreg==7) then
      kolour = 70
      call setclr(kolour)
      call curvmp(reg6lo, reg6la, 80)
    else if (nreg==8) then
      kolour = 80
      call setclr(kolour)
      call curvmp(reg7lo, reg7la, 144)
    else if (nreg==9) then
      kolour = 90
      call setclr(kolour)
      call curvmp(reg8lo, reg8la, 245)
    else if (nreg==10) then
      kolour = 100
      call setclr(kolour)
      call curvmp(reg9lo, reg9la, 285)
    end if
    if (trial) then
      print *, ' exiting plot points'
    end if

  end subroutine plotem
```

The original program on which this is based was written by Ian whilst he was on secondment to the United Nations Environment Programme. Section 9 of their Environmental Data Reports cover natural disasters and these include

- Earthquakes
- Volcanoes
- Tsunamis
- Floods
- Landslides
- Natural dams

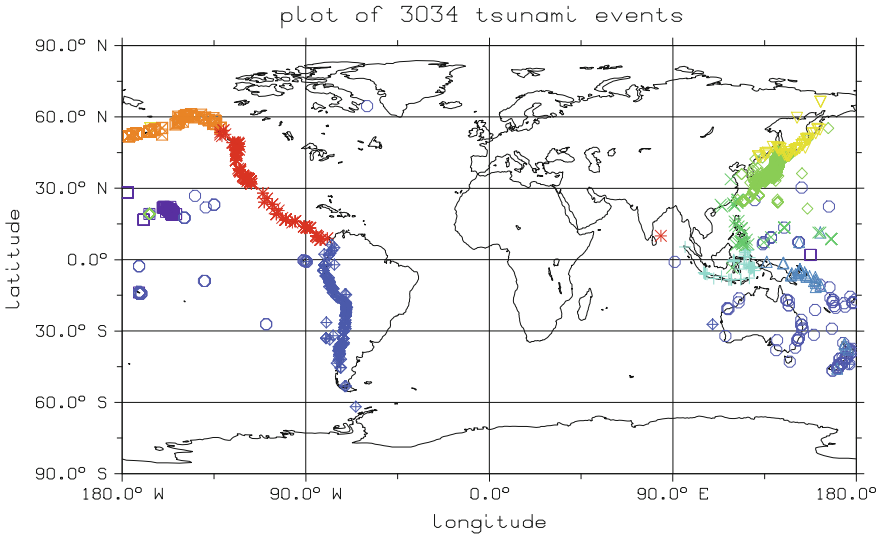
- Droughts
- Wildfires

See the bibliography for more details of these publications. The tsunami data sets are from this chapter.

The tsunami data file and graphics program can be found at:

```
https://www.fortranplus.co.uk/
```

Here is the plot produced by this program.




As you can see there are a lot of tsunami events in the Pacific rim area. A colour A4 pdf of the plot can be found at the Fortranplus website.

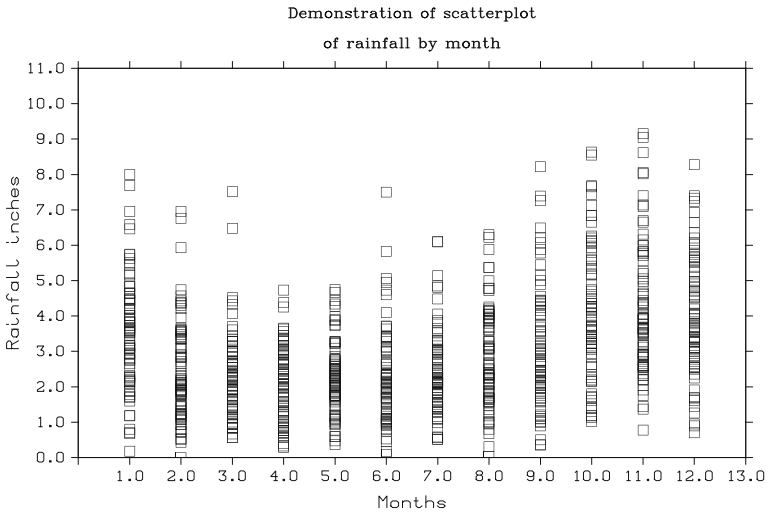
## 41.7 Example 5: Using Dislin to Plot the Met Office Data

Here is the source code for this program.

```
program ch41_dislin_05

  use dislin

  parameter (n=906)
  dimension x(n), y(n)

  open (unit=100, file='aberporth_rainfall.csv',&
          status='old')

  do i = 1, n
    read (100, 100) x(i), y(i)
100 format (f3.0, 6x, f4.2)
  end do

! Must call initialisation routine

  call disini

! Plot a border round a page

  call pagera

! Bounding box
!
! 0 ,    0       2969 ,    0
! 0 , 2099       2969, 2099

! Position of axis systems

  call axspos(450, 1800)

! axis length

  call axslen(2400, 1400)

! Change symbol rectangle by default

  call symbol(4, 0, 0)

! X axis

  call name('Months', 'X')

! Y axis
```

```
    call name('Rainfall inches', 'Y')

    call labdig(1, 'X')
    call ticks(1, 'XY')

    call titlin('Demonstration of scatterplot', 1)
    call titlin('of rainfall by month', 3)

  ! call mylab('Jan',1,'X')
  ! call mylab('Feb',2,'X')
  ! call mylab('Mar',3,'X')
  ! call mylab('Apr',4,'X')
  ! call mylab('May',5,'X')
  ! call mylab('Jun',6,'X')
  ! call mylab('Jul',7,'X')
  ! call mylab('Aug',8,'X')
  ! call mylab('Sep',9,'X')
  ! call mylab('Oct',10,'X')
  ! call mylab('Nov',11,'X')
  ! call mylab('Dec',12,'X')

  ! Plot a 2 d axis system

    call graf(0.0, 13.0, 1.0, 1.0, 0.0, 11.0, 0.0, &
      1.0)

  ! Scatter plot

    call qplsca(x, y, n)

  ! Must call terminating routine

    call disfin

  end program ch41_dislin_05
```

Here is dislin output.




These examples have shown some of the capability of the dislin library. Most graphics libraries will offer similar functionality.

## 41.8 Graphics Production Notes

Most graphics libraries will enable you to view the image on the screen. They will also have the option to generate the image in one or more file formats.

We used Postscript and encapsulated postscript in the production of the graphics included in the book.

There is a brief coverage of the postscript programming language in Chap. 3.

Encapsulated PostScript (EPS) is a postscript document with additional restrictions which is intended to be usable as a graphics file format. EPS files can be thought of as more-or-less self-contained postscript documents that describe an image or drawing and can be placed within another postscript document.

When generating the book we use the LaTeX $->$ ps $->$ pdf print option.

## 41.9 Bibliography

United Nations Environment Programme, Environmental Data Report: 1989–1990, Second Edition, Blackwell Reference, 1989.
United Nations Environment Programme, Environmental Data Report: 1991–1992, Third Edition, Blackwell Reference, 1991.

• Part 9 of these two publications are dedicated to natural disasters

Andries Van Dam; James D. Foley, The Fundamentals of Interactive Computer Graphics Addison-Wesley.

• The classic graphics textbook. Dated, but very good.

Edward R Tufte, Visual Explanations, Images and quantities, evidence and narrative, Graphics Press, Chesire, Connecticut.

Edward R Tufte, The Visual Display of Quantitative Information, Graphics Press, Chesire, Connecticut.

• Two very good books on how to present information visually

## 41.10 Problems

**41.1** Try out the examples in this chapter.

**41.2** Have a look at our resource file to find out more about what libraries are available.