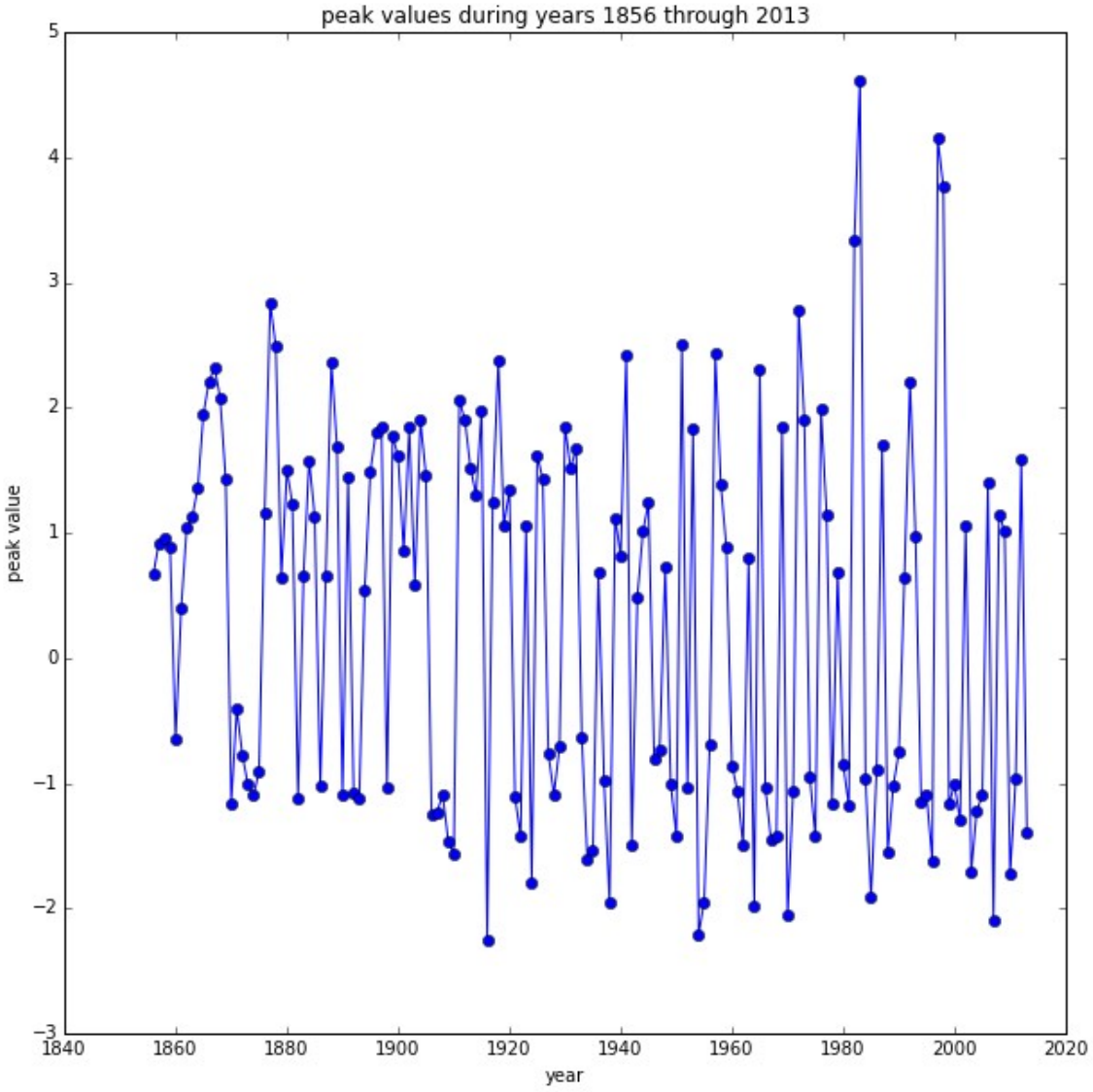
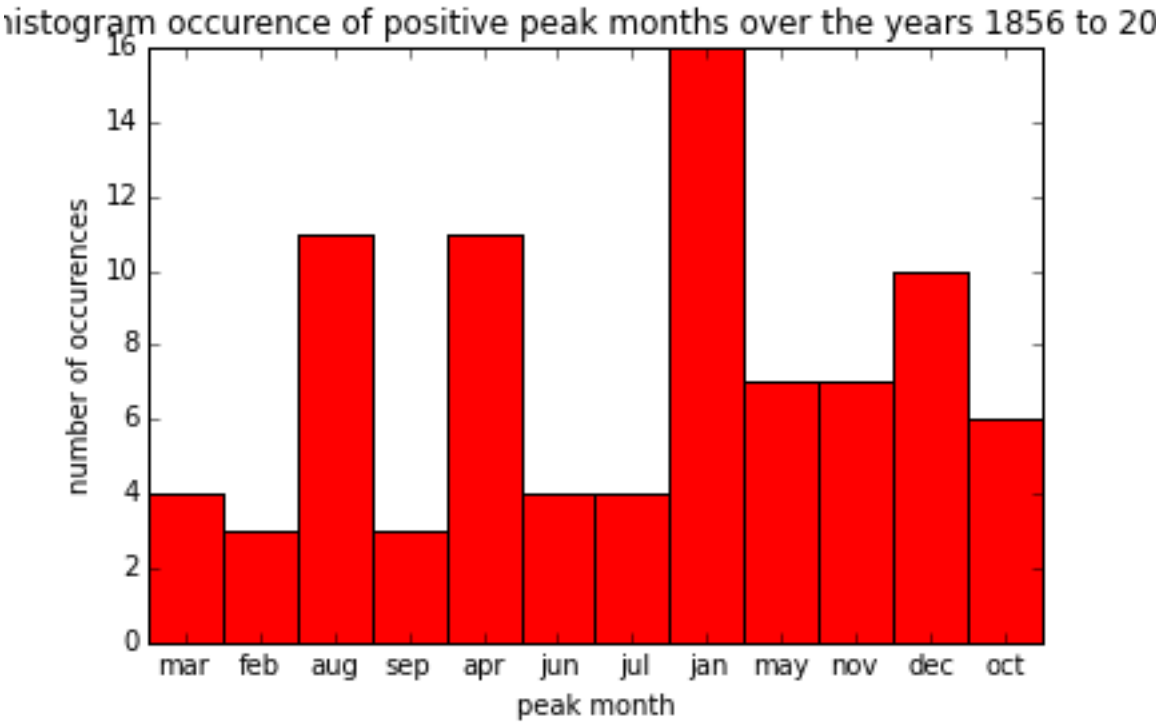
Assignment 9 Yevgeniy Krasnitskiy

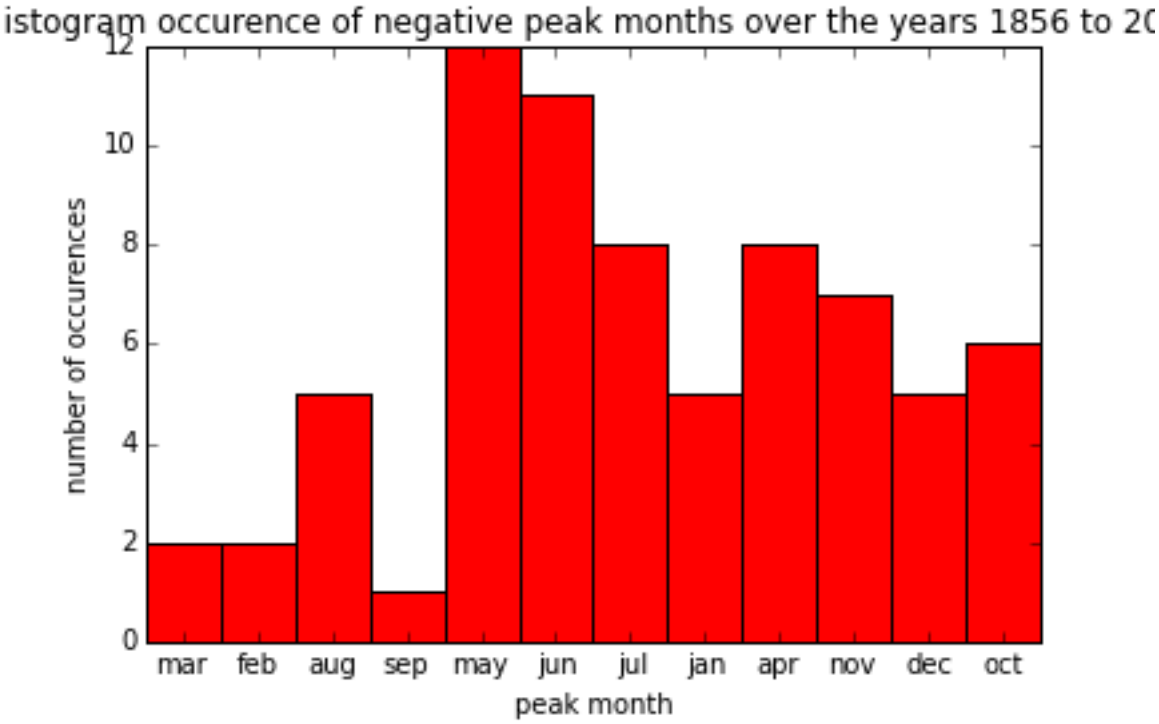
In this assignment we use El Nino data given on monthly basis over one hundred and fifty eight years. We consider the peaks of the data for each year, positive for some years, negative for other. There is an apparent periodicity, and we try to fit the smoothed data with a sum of sine and cosine functions.

Below is the plot for peaks occurring during each year:

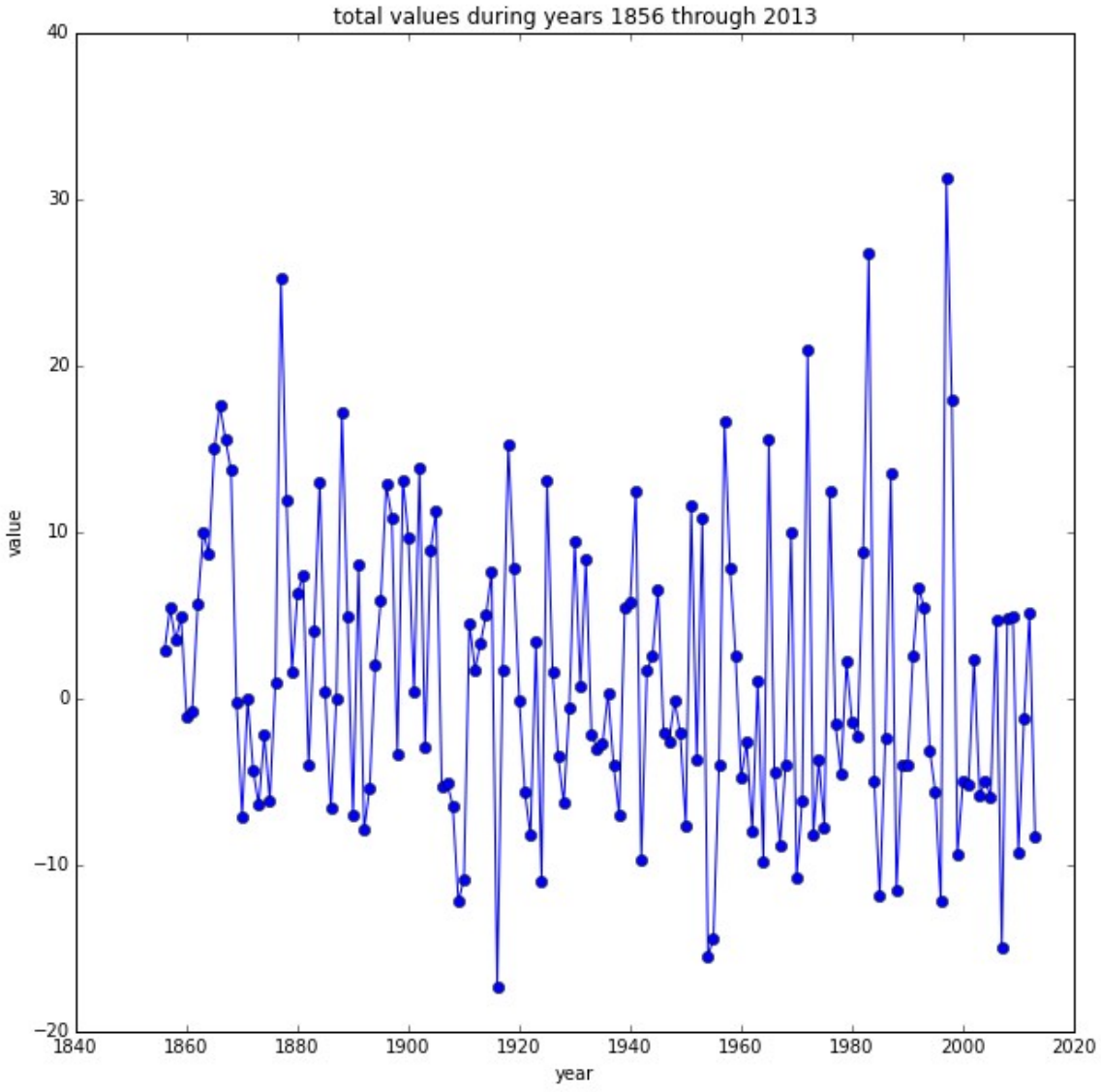


To see the frequency of positive peak values versus negative peak values on a monthly basis, we make the following histograms:





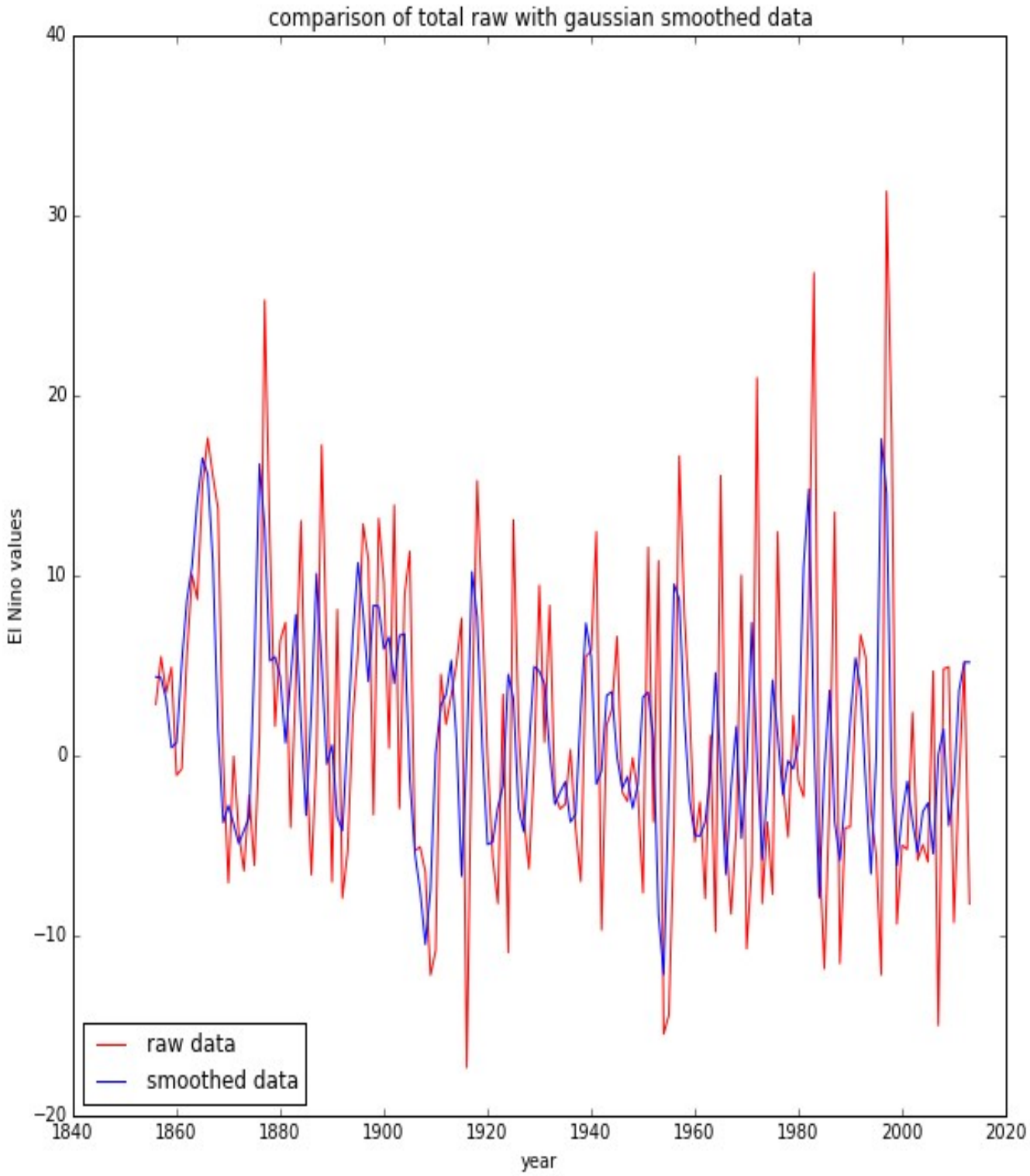
To analyze the data further, we can compute the annual total for the El Nino values:



We would like to smooth the data, and we use Gaussian smoothing over the period of three years in order to preserve the local structure of the peaks but to reduce the noise somewhat. The set of coefficients used in this smoothing procedure is:

coefficients = [0.242, 0.517, 0.242]

And the following plot can be produced:


comparison of total raw with gaussian smoothed data
El Nino values
year
raw data
smoothed data
40
30
20
10
0
−10
−20
1840
1860
1880
1900
1920
1940
1960
1980
2000
2020

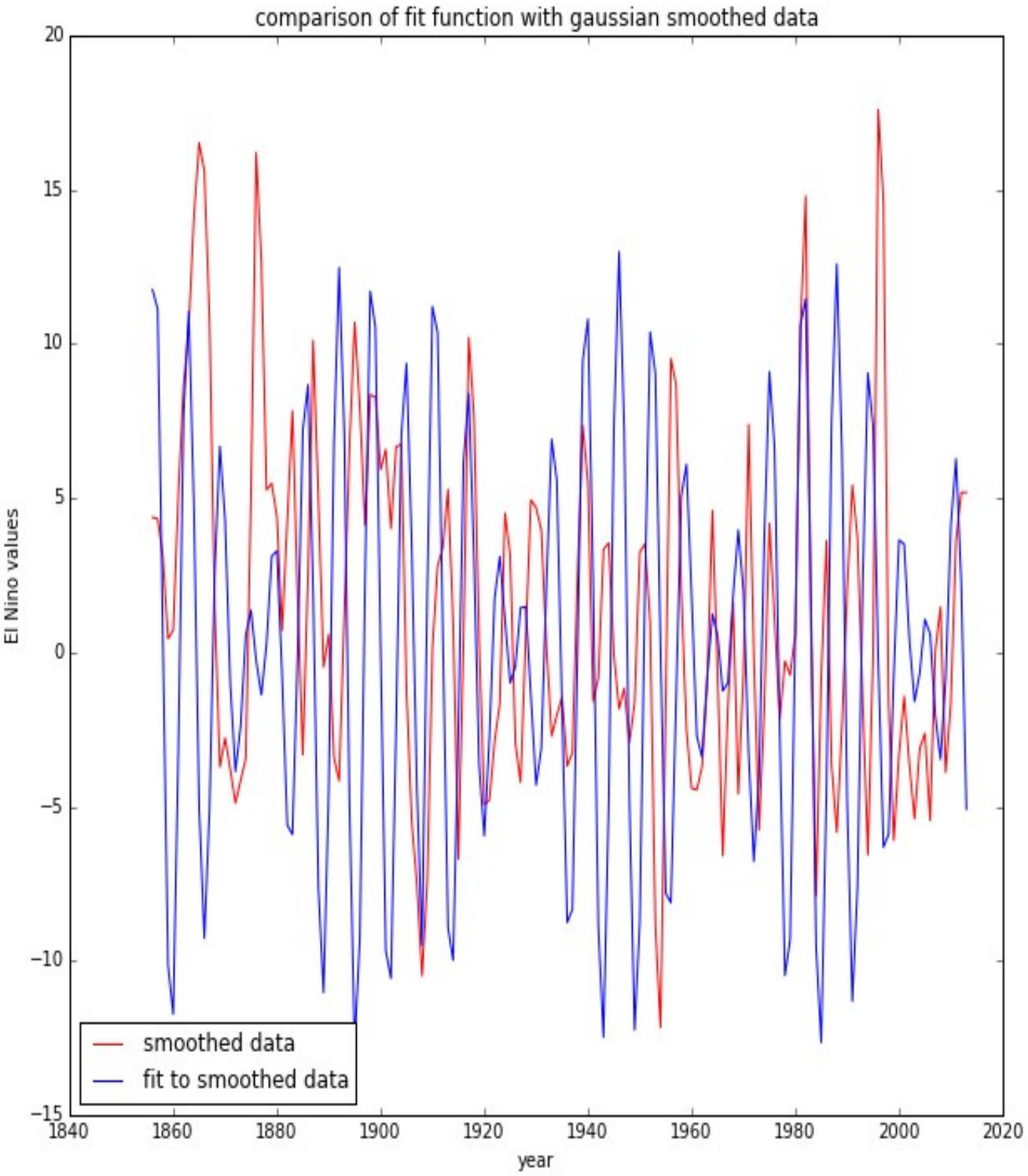
Now we would like to fit a sum of sine and cosine functions to this smoothed data. The following function is written:

```
def fit(time, a1, a2, T1, T2, phi1, phi2, dphi):
    fitFn = []
    for i in range(0, int(len(time)/3)):
        temp = a1*np.sin(time[i]*1/T1 + phi1) + a2*np.cos(time[i]*1/T2 + phi2)
        fitFn.append(temp)
    for i in range(int(len(time)/3), len(time)):
        temp = a1*np.sin(time[i]*1/T1 + phi1) + a2*np.cos(time[i]*1/T2 + phi2 + dphi)
        fitFn.append(temp)
    return fitFn
```

here we would like to set a phase offset a third of the way through the years and thus the cosine receives an adjustment in phase for the period around the year 1900. The function is called with the following parameters:

```
cycleFit = fit(yearsElNino, 7, 6, .95, 1.12, 1.5, 1.2, 1.7)
```

That is the amplitudes are a1 = 7, a2 = 6; the periods are T1 = 0.95, T2 = 1.12; the phases are phi1 = 1.5, phi2 = 1.2, and the phase offset dphi = 1.7
These are meant to be trial values, to be replaced by more accurate values when chi squared is computed.
These values and functions yield the following graph:


comparison of fit function with gaussian smoothed data
El Nino values
year
smoothed data
fit to smoothed data
20
15
10
5
0
−5
−10
−15
1840
1860
1880
1900
1920
1940
1960
1980
2000
2020

In order to improve on this result, the program from last week was modified to read:

```
module fit_data_mod

  integer, parameter :: MAX_SIZE = 400

  !! setup parameters, you may need to modify these
  !
  real, parameter :: xStepMin  =  50,  xStepMax  =  100     ! phase change timee
 step
  real, parameter :: aMin =    5,  aMax =    10   ! sine amplitude
  real, parameter :: zpMin = -2,  zpMax  = 2 ! zero point offset
  real, parameter :: pMin =  .5,  pMax   =   5    ! period
  real, parameter :: phiMin  =  -10,  phiMax  =   10    ! phase

  real, parameter :: PI = 3.1415

  !! module variables
  !
  integer :: numIterations   ! number of random searches per process
  real :: xstep              ! time step
  real :: a1, a2             ! amplitudes
  real :: zp                 ! zero point
  real :: p1, p2         ! periods
  real :: phi1, phi2, dphi   ! phases

 real :: xx(MAX_SIZE), yy(MAX_SIZE)

contains

!!!!!!!!
! Read in data from file (unit # 1)
!
subroutine read_data()
  implicit none
  integer :: i, k
  real    :: x, y

  k = 0
  do i = 1, 158
    read(1,*) x, y
    k = k + 1
    xx(i) = k
    yy(i) = y
  end do

end subroutine read_data

!!!!!!!!
! Smooth raw data
!
```

```
subroutine smooth_data()
  implicit none
  integer :: i
  real    :: smoothy

  do i = 1, 158
    smoothy = (yy(i) + yy(i+1) + yy(i+2) + yy(i+3) + yy(i+4))/5.0
!     write(3,*) xx(i), smoothy
  end do

end subroutine smooth_data

!!!!!!!!
! Calculate random variables for an interation
!
subroutine randomize()
  implicit none
  real :: r
 call random_number(xstep)      ! uniformly distributed on internal [0.0, 1.0)
  xstep = xStepMin + (xStepMax - xStepMin)*xstep

  call random_number(a1)      ! sine amplitude
  a1 = aMin + (aMax - aMin)*a1

  call random_number(a2)      ! cosine amplitude
  a2 = aMin + (aMax - aMin)*a2

  call random_number(zp)        ! zero point
  zp = zpMin + (zpMax - zpMin)*zp
   write(*,*) zp
  call random_number(p1)         ! sine period
  p1 = pMin + (pMax - pMin)*p1

  call random_number(p2)         ! cosine period
  p2 = pMin + (pMax - pMin)*p2

  call random_number(phi1)       ! phase
  phi1 = phiMin + (phiMax - phiMin)*phi1

  call random_number(phi2)       ! phase
 phi2 = phiMin + (phiMax - phiMin)*phi2

  call random_number(dphi)       ! phase
  dphi = phiMin + (phiMax - phiMin)*dphi

end subroutine randomize

end module fit_data_mod

program fit_data
  use mpi
  use fit_data_mod
  implicit none
```

```
  character(len=32) :: arg
  integer :: i, err, iter, seed(12), rank, size, out_unit
  real :: chi, chimin, chisq, resid, sk, yk, ykk

  !! initialize MPI
  !
  call MPI_Init(err)

 call MPI_Comm_rank(MPI_COMM_WORLD, rank, err)
  call MPI_Comm_size(MPI_COMM_WORLD, size, err)

  !! get number of iterations from the command line
  !
  call get_command_argument(1, arg)
  read(arg, '(i12)') numIterations

  if (len_trim(arg) == 0) then
    if (rank == 0) then
      print *, 'usage: fit_data num_iterations'
      print *, '       please input # interations as a command line argument'
      print *
    end if
    call MPI_Finalize(err)
    stop
  end if

  call read_data()
  call smooth_data()

  if (rank == 0) then
    print *, "Running MPI with size of", size
    print *, "Number of iterations is ", numIterations
    print *
    ! write(*,*) 'Going to work on 158 data points'
    ! write(*,*) 'fitting cosine + sin wave + phase adjustement at later time
steps'
    ! write(*,*) 'From class for initial values'
    ! write(*,*) 'sine amplitude = 8 (a1)'
    ! write(*,*) 'cosine amplitude = 8 (a2)'
    ! write(*,*) 'zeropoint = 2 (zp)'
    ! write(*,*) 'sine period = 1.7 (p1)'
    ! write(*,*) 'cosine period = 1.5 (p2)'
    ! write(*,*) 'phase = -7 (phi)'
    ! write(*,*) 'phase shift = 2 (phi1)'
    ! write(*,*) 'timestep = 50 (xstep)'
    ! write(*,*)
  end if
! Initialize random number generator to something based on MPI rank so
  !   that it is different for every MPI process.
  !
  call random_seed(get=seed)
  seed = (rank + 1)*seed          ! some modification based on the MPI rank
  call random_seed(put=seed)
```

```
  out_unit = 21 + rank    ! files must be different for each MPI rank
  chimin = 50

  do iter = 1, numIterations
    call randomize()

    do i = 1, 158
      yk = a1*sin(2*PI*(xx(i)/p1) + phi1) + zp
      if (i .lt. xstep) then
        sk = a2*cos(2*PI*(xx(i)/p2) + phi2)
      else
        sk = a2*cos(2*PI*(xx(i)/p2) + phi2 + dphi)
      end if
      ykk = yk + sk
      resid = (ykk – yy(i))
resid = (resid*resid)/yy(i)
      chi = chi + resid
    end do

    chisq = chi/158
    chi = 0
    if (chisq .lt. chimin) then
      write(*,*) rank, iter, chimin, a1, a2, p1, p2, phi1, phi2, dphi
      write(out_unit,'(i3,1x,i9,10f8.2)') rank, iter, chimin, a1, a2, p1, p2,,
phi1, phi2, dphi, xstep, zp
      chimin = chisq
    end if
  end do

  call MPI_Finalize(err)

end program
```

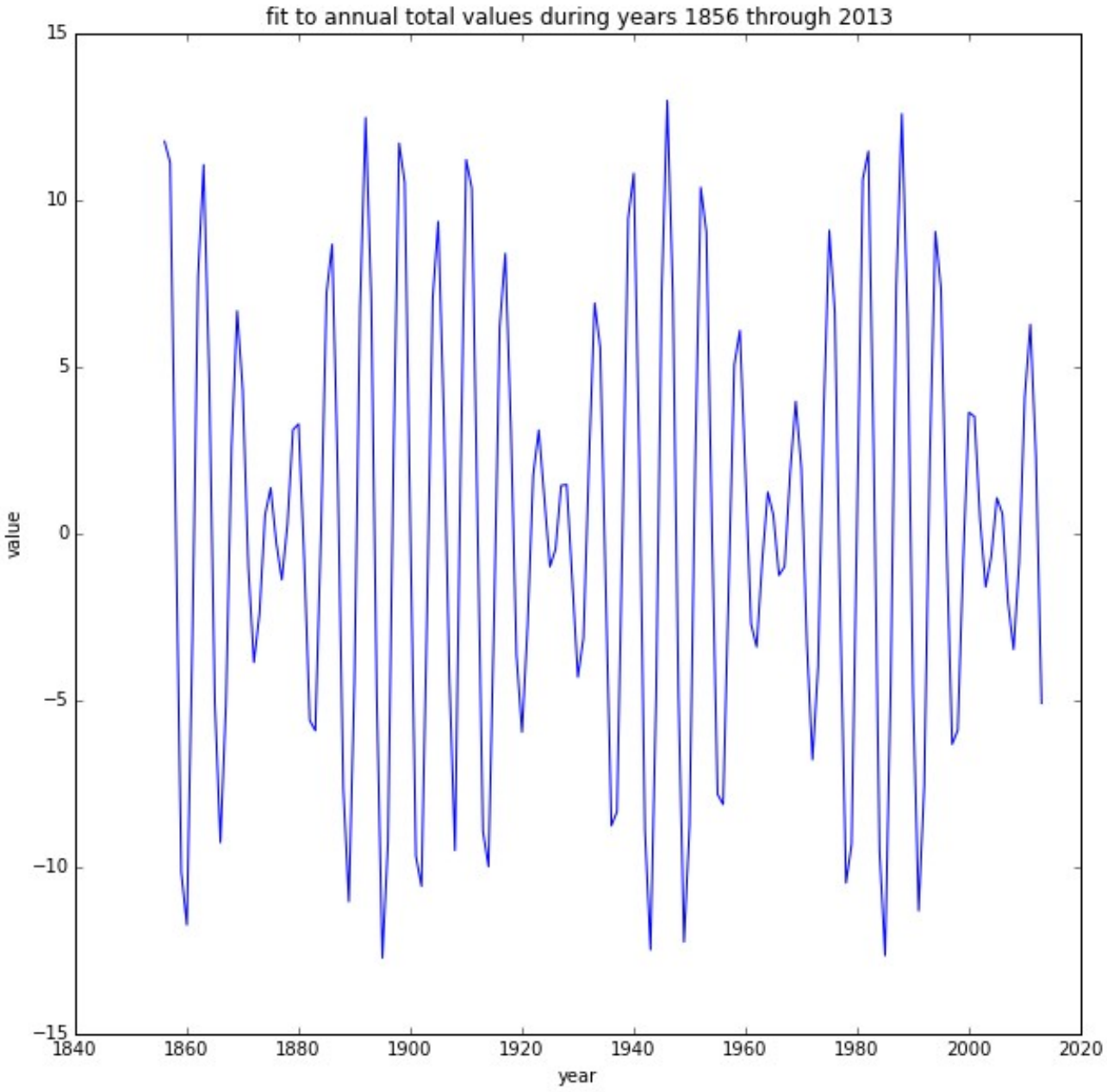
As well as modifying the file fort.1 to contain the data for the years and total values.

However, when the routine was implemented with 8 processors, no fort.xx files were built up.

Unfortunately my lack of familiarity with Fortran prevents me from resolving the issue at the moment. The following were modified from last week's values: the functions used to fit data from linear + sinusoidal modified to two sinusoidal functions; the number of points instead of being 1000 is modified to being 158 (this modification was also made in the Makefile call but did not produce any difference).
The intervals on which the parameters are randomly chosen have been determined from the preliminary plot shown above.

At the moment further analysis is difficult to perform and I need to study Fortran commands and file structure. My plan is to return to this program very soon and complete the assigned tasks.

For the moment, the above parameters give the fit that looks like this:



a1 Sin(time/T1 + phi1) + a2 Cos(time/T2 + phi2) , time < 1900
a1 Sin(time/T1 + phi1) + a2 Cos(time /T2 + phi2 + dphi) , time > 1900

a1 = 7, a2 = 6; the periods are T1 = 0.95, T2 = 1.12; the phases are phi1 = 1.5, phi2 = 1.2, and the phase offset dphi = 1.7