```
PROGRAM potential_field
    IMPLICIT NONE
    ! Variable declarations
    CHARACTER(LEN=58), PARAMETER    ::  filedne = '("POTENTIAL_FIELD: Error, &
                                        &file ''",A,"'' does not exist.")'
    CHARACTER(LEN=218), PARAMETER   ::  output = '("POTENTIAL_FIELD: The &
                                        &average electric field strength was ",&
                                        &1pg12.5," N/C, with",/,&
                                        &"POTENTIAL_FIELD: a sample standard &
                                        &deviation of ",1pg12.5," N/C.",/,&
                                        &"POTENTIAL_FIELD: This was calculated &
                                        &using ",I4," data-points.")'
    CHARACTER(LEN=32)               ::  infile  = 'in.dat',                     &
                                        outfile = 'out.dat'
    INTEGER, PARAMETER              ::  data_max = 1012
    INTEGER                         ::  i,                                      &
                                        ioval,                                  &
                                        num_args,                               &
                                        data_size = data_max
    REAL(8)                         ::  mean,                                   &
                                        stdev
    REAL(8), DIMENSION(data_max)    ::  field,                                  &
                                        postn,                                  &
                                        potntl
!-------------------------------------------------------------------------------!

    ! Get the input and output file-names from the command-line
    num_args = COMMAND_ARGUMENT_COUNT()
    IF ( num_args >= 1 ) CALL GET_COMMAND_ARGUMENT(1,infile)
    IF ( num_args >= 2 ) CALL GET_COMMAND_ARGUMENT(2,outfile)

    ! Read in data from the input file
    OPEN(UNIT=10, FILE=infile, IOSTAT=ioval, STATUS='old')
    IF ( ioval /= 0 ) THEN ! Make sure file exists
        WRITE(0, filedne) TRIM(infile)
        STOP
    END IF
    DO i = 1,data_max ! Read until end of file or reach maximum amount of data
        READ(10,*,IOSTAT=ioval) postn(i), potntl(i)
        IF ( ioval /= 0 ) THEN
            data_size = i - 1
            EXIT
        END IF
        IF ( i == data_max ) WRITE(6,*) 'POTENTIAL_FIELD: Could not read '//   &
            'all input data. Truncated after ', data_max, ' elements.'
    END DO
    CLOSE(10)

    ! Calculate the negative derivative of the input data
    field(1:data_size) = differentiate(postn(1:data_size), potntl(1:data_size))
    field(1:data_size) = -1.0 * field(1:data_size)

    ! Send the data to the output file
    OPEN(10, FILE=outfile, STATUS='unknown')
    WRITE(10,*) '#Position                  Field Strength'
    DO i = 1, data_size
        WRITE(10,*) postn(i), field(i)
    END DO
    CLOSE(10)

    ! Calculate the statistics and print results to screen
    CALL stats(field(1:data_size), mean, stdev)
    WRITE(6,output) mean, stdev, data_size

    STOP
```

```
CONTAINS
    FUNCTION differentiate ( independent, dependent )
        IMPLICIT NONE
        ! Input and output variables
        REAL(8), DIMENSION(:), INTENT(IN)   ::  dependent,                      &
                                                independent
        REAL(8), DIMENSION(:), ALLOCATABLE  ::  differentiate
        ! Local variables
        INTEGER ::  i, ret_size
    !-------------------------------------------------------------------------!
        ! Figure out how much data there is to process
        ret_size = MIN(SIZE(dependent),SIZE(independent))
        ALLOCATE(differentiate(1:ret_size))

        ! Calculate derivative for first data-point
        differentiate(1) = (dependent(2) - dependent(1))/(independent(2) -     &
            independent(1))

        ! Calculate derivative for data-points in the middle
        FORALL (i = 2:(ret_size - 1)) differentiate(i) = (dependent(i+1) -     &
            dependent(i-1))/(independent(i+1) - independent(i-1))

        ! Calculate the derivative for the last data-point
        differentiate(ret_size) = (dependent(ret_size) -                       &
            dependent(ret_size-1))/(independent(ret_size) -                    &
            independent(ret_size -1))

        RETURN
    END FUNCTION differentiate


    SUBROUTINE stats ( array, mean, stdev )
        IMPLICIT NONE
        ! Input and output variables
        REAL(8), DIMENSION(:), INTENT(IN)   ::  array
        REAL(8), INTENT(OUT)                ::  mean,                          &
                                                stdev
        ! Local variables
        INTEGER ::  i, num
        REAL(8) ::  running_tot
    !-------------------------------------------------------------------------!
        ! Compute the mean
        num  = SIZE(array)
        mean = SUM(array) / REAL(num,8)

        ! Compute the standard deviation
        DO i = 1, num
            running_tot = running_tot + ( array(i) - mean )**2
        END DO
        stdev = SQRT(1.d0/REAL((num - 1),8) * running_tot)

        RETURN
    END SUBROUTINE stats

END PROGRAM potential_field
```