time_series_smooth.c

This C language program implements the Section 5.1 integer math and the Section 5.2 forecast reset functionality described in the companion paper https://arxiv.org/abs/1908.08123.

This documentation contains three parts:

1. Program Input
2. Program Output
3. Program Source Listing

To compile and run the program:

1. Download the time_series_smooth repository
2. Go to the **src** directory and execute compile_time_series_smooth.bat (installed gcc compiler is required)
3. The time_series_smooth.exe binary is created in the **demo** directory
4. Go to the **demo** directory and execute run_time_series_smooth.bat
5. The program output below is produced.

Options:

1. -h = help
2. -n = n_alpha - integer value of [1/alpha] default is 10
3. -r = reset smoother at count value plus one
4. -t = reset smoother time interval default is 5 seconds
5. -w = write verbose output to comma delimited file

The function in the program that contains the full algorithm is "time_series_smooth()". This function's source code, call in the main program, prototype, and data structure (EXP_SMOOTH_DATA) are shown below in **red**.

Program Input

```
time_series_smooth_input.txt
1 571
2 565
3 564
4 936
5 576
6 574
7 569
8 563
9 562
10 570
11 585
12 573
13 570
14 574
15 570
16 567
17 567
18 563
19 562
20 569
21 569
22 595
23 566
24 796
25 594
```

Program Output

```
---------Time Series Smoothing Algorithm----------
n_alpha = 10  reset_time = 5
_____count___observe__forecast______diff___diffsum
         1       571       571         0         0
         2       565       568        -3        -3
         3       564       566        -2        -5
         4       936       658       278       273
         5       576       641       -65       208
         6       574       629       -55       153
         7       569       620       -51       102
         8       563       612       -49        53
         9       562       606       -44         9
        10       570       602       -32       -23
        11       585       599       -14       -37
        12       573       594       -21       -58
        13       570       589       -19       -77
        14       574       586       -12       -89
        15       570       581       -11      -100
        16       567       576        -9      -109
        17       567       574        -7      -116
        18       563       570        -7      -123
        19       562       568        -6      -129
        20       569       568         1      -128
        21       569       567         2      -126
        22       595       571        24      -102
        23       566       568        -2      -104
        24       796       612       184        80
        25       594       609       -15        65
```

Program Source Listing

```
/*****************************************************************************
 * This is a C language source listing of a computer program which exercises
 * a time series smoothing algorithm based on Exponential Smoothing.
 *
 * The time_series_smooth function contains the smoothing algorithm.
 *
 * Author: James F. Brady 2019
 *****************************************************************************
 * Definitions:
 * alpha = smoothing constant 0 < alpha < 1
 * n_alpha = set integer value of [1/alpha]
 * N_ALPHA = default integer value of [1/alpha]
 * reset_time = set sample number reset time
 * RESET_TIME = default sample number reset time
 * reset_count = reset smoother at count value plus one
 * xt = current sample (observation)
 * stx1 = first smoothed statistic
 * stx2 = second smoothed statistic
 * n = sample number (may be reset)
 * ft = forecast
 * last_update_time = timestamp of last update
 * count = running sample count for output
 * diff = xt-ft for output
 * diffsum = sum of xt-ft for output
 *****************************************************************************
 * Args:
 *   input file name
 * Options:
```

```
 *   -h = help
 *   -n = n_alpha - integer value of [1/alpha] default is 10
 *   -r = reset smoother at count value plus one
 *   -t = reset smoother time interval default is 5 seconds
 *   -w = write verbose output to comma delimited file
 ******************************************************************************/
/************************************
 * Includes
 ************************************/
#include <sys/time.h>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <unistd.h>
#include <stdint.h>

/************************************
 * Defines
 ************************************/
#define N_ALPHA 10
#define RESET_TIME 5

/************************************
 * Data Structures
 ************************************/
typedef struct
{
    int      n_alpha;
    int      stx1;
    int      stx2;
    int      n;
    int      ft;
    int      reset_time;
    int      last_update_time;
}EXP_SMOOTH_DATA;

/************************************
 * Prototypes
 ************************************/
int time_series_smooth(EXP_SMOOTH_DATA *, int);
int sleep(int);

/************************************
 * Main
 ************************************/
int main(int argc, char **argv)
{
  FILE            *in_file;
  FILE            *out_file;
  int             opt;
  int             count;
  int             xt;
  int             diff;
  bool            errorFlag = false;
  int             diffsum = 0;
  int             reset_count = 0;
  char            out_file_name[100];
  int             out_file_set = 0;
```

```
EXP_SMOOTH_DATA data;

/*********************
 * Initialize data
 *********************/
memset((char *)&data, 0, sizeof(data));
/*************************
 * Initialize variables
 *************************/
data.n_alpha = N_ALPHA;
data.reset_time = RESET_TIME;
/*******************************
 * Get options
 *******************************/
while ((opt = getopt(argc, argv, "hn:r:t:w:")) != EOF)
{
  switch (opt)
  {
    /*******************************
     * help
     *******************************/
    case 'h':
      printf("\n*****************************************************\n");
      printf("* Args:\n");
      printf("* input file name\n");
      printf("*\n");
      printf("* Options:\n");
      printf("* -h = help\n");
      printf("* -n = n_alpha - integer value of [1/alpha] default is 10\n");
      printf("* -r = reset smoother at count value plus one\n");
      printf("* -t = reset smoother time interval default is 5 seconds\n");
      printf("* -w = write verbose output to comma delimited file\n");
      printf("*****************************************************\n");
      errorFlag = true;
      break;
    /*********************************************
     * Option -n  integer value of [1 / alpha]
     *********************************************/
    case 'n':
      if ((data.n_alpha = strtol(optarg, 0, 0)) == 0)
      {
        fprintf(stderr,"Invalid n_alpha = %s\n",optarg);
        errorFlag = true;
      }
      if (data.n_alpha < 0)
      {
        fprintf(stderr,"Invalid n_alpha = %d\n",data.n_alpha);
        errorFlag = true;
      }
      break;
    /*********************************************
     * Option -r reset smoother at count value plus one
     *********************************************/
    case 'r':
      if ((reset_count = strtol(optarg, 0, 0)) == 0)
      {
        fprintf(stderr,"Invalid reset_count = %s\n",optarg);
        errorFlag = true;
      }
      if (reset_count < 0)
      {
        fprintf(stderr,"Invalid reset_count = %d\n",reset_count);
        errorFlag = true;
      }
      break;
    /***************************************************
     * Option -t reset smoother time interval
     ***************************************************/
    case 't':
      if ((data.reset_time = strtol(optarg, 0, 0)) == 0)
      {
        fprintf(stderr,"Invalid reset_time = %s\n",optarg);
        errorFlag = true;
      }
      if (data.reset_time < 0)
      {
        fprintf(stderr,"Invalid reset_time = %d\n",data.reset_time);
        errorFlag = true;
      }
      break;
    /************************************************
     * Option -w write output to comma delimited file
     ************************************************/
    case 'w':
      if (strcpy(out_file_name,optarg) != 0)
    {
        out_file_set = 1;
      }
    else
      {
        fprintf(stderr,"Invalid file name = %s\n",optarg);
        errorFlag = true;
      }
      break;
    /**********************************
     * default
     **********************************/
    default:
      break;
  }
}
/*******************************************
 * The errorFlag is set
 *******************************************/
if (errorFlag)
{
  exit(1);
}
/**********************************
 * Check args passed
 **********************************/
if (argc - optind < 1)
{
  fprintf(stderr,"usage: %s [opt-hn:r:t:w:] file name\n",argv[0]);
  exit(1);
}
/*******************
 * Open input file
 *******************/
if ((in_file = fopen(argv[argc-1],"r")) == NULL)
```

```c
{
  fprintf(stderr,"Error opening input file = %s\n", argv[argc-1]);
  exit(1);
}
/*********************
 * Print heading
 *********************/
fprintf(stdout,"\n");
fprintf(stdout,"---------Time Series Smoothing Algorithm----------\n");
fprintf(stdout,"n_alpha = %d",data.n_alpha);
fprintf(stdout,"  reset_time = %d",data.reset_time);
if (reset_count)
{
  fprintf(stdout,"  reset_count = %d",reset_count);
}
fprintf(stdout,"\n_____count___observe__forecast______diff___diffsum\n");
/*************************************************
 * Option -w open output file and write heading
 *************************************************/
if (out_file_set)
{
  if ((out_file = fopen(out_file_name,"w")) == NULL)
  {
    if (out_file == NULL)
    {
       fprintf(stderr,"Error opening output file = %s\n",out_file_name);
       exit(1);
    }
  }
  fprintf(out_file,"%s\n","Time Series Smoothing Algorithm");
  fprintf(out_file,"%s%d","n_alpha = ,",data.n_alpha);
  fprintf(out_file,"%s%d",",,reset_t = ,",data.reset_time);
  if (reset_count)
  {
    fprintf(out_file,"%s%d",",,reset_c = ,",reset_count);
  }
  fprintf(out_file,"\n%s\n","count,observe,forecast,diff,diffsum,n,stx1,stx2");
}

/****************************************************************
 * Read input file and write xt and ft to stdout
 ****************************************************************/
for (;;)
{
  /******************************
   * Read input file
   ******************************/
  if (fscanf(in_file, "%d%d", &count, &xt) == EOF)
  {
    break;
  }
  /******************************************************
   * Call time series smoothing function
   ******************************************************/
  time_series_smooth(&data, xt);

  /******************************************************
   * Write xt and ft and diffs to stdout
   ******************************************************/
  diff = xt - data.ft;
```

```
    diffsum += diff;
    fprintf(stdout,"%10d%10d%10d%10d%10d\n",
                    count,
                    xt,
                    data.ft,
                    diff,
                    diffsum);
    /******************************************************
     * Option -w write xt and ft and diffs to output file
     ******************************************************/
    if (out_file_set)
    {
      fprintf(out_file,"%d%s%d%s%d%s%d%s%d%s%d%s%d%s%d\n",
              count,",",xt,",",data.ft,",",diff,",",diffsum,
                    ",",data.n,",",data.stx1,",",data.stx2);
    }
    /***********************************************
     * Option -r reset the smoother at count value
     ***********************************************/
    if (reset_count && reset_count == count)
    {
      sleep(data.reset_time + 1);
    }
  }
  /********************
   * Close input File
   ********************/
  fclose(in_file);
  /******************************
   * Option -w close output file
   ******************************/
  if (out_file_set)
  {
    fclose(out_file);
  }
  return(0);
}

/*****************************************************
 * Time Series Smoothing Algorithm
*****************************************************/
int time_series_smooth(EXP_SMOOTH_DATA *data, int xt)
{
  struct timeval curr_time;

  /**********************************
  * Check xt range
  ***********************************/
  if (xt > INT32_MAX / data->n_alpha)
  {
    xt = INT32_MAX / data->n_alpha;
  }
  if (xt < INT32_MIN / data->n_alpha)
  {
    xt = INT32_MIN / data->n_alpha;
  }

  /***************************************
  * Get the current time
  ****************************************/
  gettimeofday(&curr_time, NULL);

  /***********************************************************
  * Reset data->n if last update > RESET_TIME_INTERVAL
  ***********************************************************/
  if (((curr_time.tv_sec - data->last_update_time)) >
        data->reset_time)
  {
    data->n = 0;
  }

  /****************************************
  * Update data->last_update_time
  ****************************************/
  data->last_update_time = curr_time.tv_sec;

  /***************************************************************
  * Double Exp Smooth if data->n >= data->n_alpha
  ***************************************************************/
  if (data->n >= data->n_alpha)
  {
    data->stx1 = (xt + (data->n_alpha-1) * data->stx1)
                 / data->n_alpha;
    data->stx2 = (data->stx1 + (data->n_alpha-1) * data->stx2)
                 / data->n_alpha;
    /***************************************************************
    * If data->n_alpha > 1
    ***************************************************************/
    if (data->n_alpha > 1)
    {
      data->ft = 2 * data->stx1 - data->stx2 + (data->stx1 - data->stx2)
                 / (data->n_alpha-1);
    }
    /***************************************************************
    * If data->n_alpha = 1
    ***************************************************************/
    else
    {
      data->ft = data->stx1;
    }
  }
  /*********************************************************************
  * Average if data->n < data->n_alpha
  *********************************************************************/
  else
  {
    data->n++;
    data->stx1 = (xt + (data->n-1) * data->stx1)
                 / data->n;
    data->stx2 = data->stx1;
    data->ft = data->stx1;
  }
  return(0);
}
```