U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

# Simulation of a Poisson arrival process

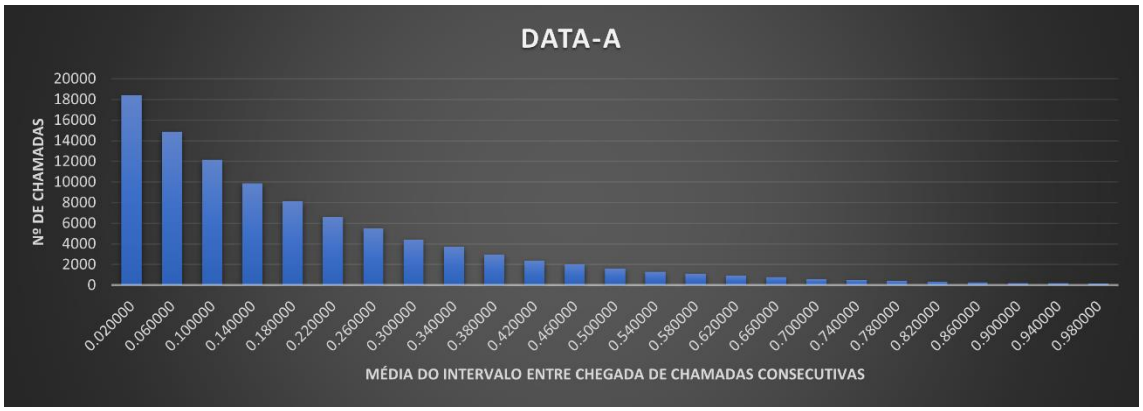
Rui Filipe Martins Barbosa up201605740

Durante esta simulação foram usados os seguintes valores:

- $\lambda$ = 5 chamadas/segundo
- número total de chamadas = 100000
- histograma com 25 colunas

a) Nesta alínea é pedido para obter um histograma do intervalo entre a chegada de chamadas consecutivas e um estimador da média desse mesmo intervalo, ambos através de um programa de simulação em C. Resultou isto:

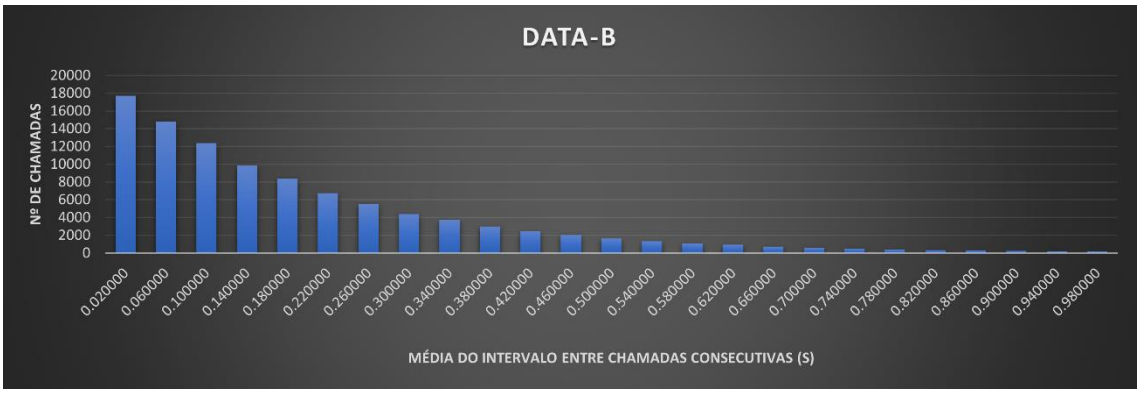
```
média do intervalo entre chamadas consecutivas = 0.199160
```




Observando os resultados, a média é praticamente igual à esperada $\frac{1}{\lambda} = \frac{1}{5} = 0.2$ e a distribuição resultante do gráfico correspondente ao intervalo entre chegadas de chamadas é exponencial.

b) Nesta alínea é pedido para obter o mesmo da anterior tendo em conta a definição de um processo de Poisson, isto é a probabilidade de ocorrer um evento num intervalo de tempo é dada por $\Delta * \lambda$ . Foi escolhido um $\Delta = 0.001$ , resultando assim $P(evento) = 0.001 * 5 = 0.005$

```
média do intervalo entre chamadas consecutivas = 0.199986
```




Observando os resultados, podemos tirar as mesmas conclusões que na alínea anterior, tanto em relação á média, bem como á distribuição exponencial resultante dos valores obtidos.

Em suma, com o reunir do conhecimento obtido por este trabalho, é seguro afirmar que uma simulação de tráfego pode ser aproximada por um processo de Poisson ou por uma simulação orientada a eventos.

A)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <math.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>
#include "lista.h"

#define LAMBDA 5
#define DELTA 0.04
#define N_EVENTS 100000
#define RAND_MAX 2147483647
#define HIST_SIZE 25
//delta =(1/5)*(1/lambda)

#define ARRIVAL 1
#define DEPARTURE 0

int print_csv( char * csv_file , int * histogram){
  FILE *file_out;
  file_out = fopen(csv_file, "w+");

  if(file_out == NULL){
      perror("fopen");
      return -1;
  }
  printf("\nwrite of data to %s started \n" , csv_file);
  fprintf(file_out, "position, Time, arrival Calls of %d calls)\n", N_EVENTS);

  for (int j = 0; j < HIST_SIZE; j++){
    fprintf(file_out, "%d, %lf, %d\n", j, (2*j+1)/(float)(HIST_SIZE*2), histogram[j]);
  }

  fclose(file_out);
  return 0;
}

float rand01(){ return (double)rand() / (double)((unsigned)RAND_MAX + 1); }

int main(int argc, char* argv[]){

  if(argc < 2)  {
    perror("need another file name to save data");
    return -1;
  }

  //check if the csv file exists
  char filename1[100];
  snprintf(filename1, sizeof filename1 , "%s%s", argv[1],".csv");
  if ( access( filename1 , F_OK ) != 0 ) {
```

```c
    printf("%s does not exist" , argv[1] );
    return -1;
  }

  int n_events;
  float c = 0 , summation = 0 , curr_time = 0;
  int *histogram = (int*)malloc(HIST_SIZE*sizeof(int));
  lista *event = NULL;

  srand(time(NULL));
  event = adicionar(event , ARRIVAL , 0);

  for (n_events = 0; n_events < N_EVENTS; n_events++) {

    c = - (1 / ((float)LAMBDA)) * log( rand01() );
    curr_time = event->tempo + c;
    event = rem(event);
    event = adicionar(event, ARRIVAL, curr_time);

    for(int i = 0; i < HIST_SIZE; i++) {
      if((c > i*DELTA) && (c <= (i+1)*DELTA)) {
        histogram[i]++;
        break;
      }
    }
  }

  printf("média do intervalo entre chamadas consecutivas = %lf\n", (float)(curr_time / n_events) )

  if(print_csv(filename1, histogram) != 0){
      perror("hist print error");
      return -1;
  }

  puts("\nwrite of data complete");
  return 0;
}

```

B)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <math.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>
#include "lista.h"

#define LAMBDA 5
#define DELTA 0.04
#define TOTAL_EVENTS 100000
#define RAND_MAX 2147483647
#define HIST_SIZE 25
//delta =(1/5)*(1/lambda)

#define ARRIVAL 1
#define DEPARTURE 0

int print_csv( char * csv_file , int * histogram){
  FILE *file_out;
  file_out = fopen(csv_file, "w+");

  if(file_out == NULL){
      perror("fopen");
      return -1;
  }
  printf("\nwrite of data to %s started \n" , csv_file);
  fprintf(file_out, "position, Time, arrival Calls of %d calls)\n", TOTAL_EVENTS);

  for (int j = 0; j < HIST_SIZE; j++){
    fprintf(file_out, "%d, %lf, %d\n", j, (2*j+1)/(float)(HIST_SIZE*2), histogram[j]);
  }

  fclose(file_out);
  return 0;
}

float rand01(){ return (double)rand() / (double)((unsigned)RAND_MAX + 1); }

int main(int argc, char* argv[]){

  if(argc < 2)  {
    perror("need file name to save data");
    return -1;
  }

```

```c
  //check if the csv file exists
  char filename1[100];
  snprintf(filename1, sizeof filename1 , "%s%s", argv[1],".csv");
  if ( access( filename1 , F_OK ) != 0 ) {
    printf("%s does not exist" , argv[1] );
    return -1;
  }

  int n_events = 0;
  float c = 0 , curr_time = 0 , prev_time = 0;
  float prob = LAMBDA*1e-3;
  int *histogram = (int*)malloc(HIST_SIZE*sizeof(int));
  lista *event = NULL;

  srand(time(NULL));

  for (int j = 0; n_events < TOTAL_EVENTS; j++) {
    if( rand01() < prob) {
      if(event != NULL) event = rem(event);

      curr_time = j * 1e-3 ;
      event = adicionar(event, ARRIVAL, curr_time - prev_time );
      c = (curr_time - prev_time);

      for(int i = 0; i < HIST_SIZE; i++) {
        if((c > i*DELTA) && (c <= (i+1)*DELTA)) {
          histogram[i]++;
          break;
        }
      }
      prev_time = curr_time;
      n_events++;
    }
  }

  printf("curr_time : %f\n", curr_time  );
  printf("média do intervalo entre chamadas consecutivas = %lf\n", (curr_time / n_events) );

  if(print_csv(filename1, histogram) != 0 ){
      perror("hist print error");
      return -1;
  }

  puts("\nwrite of data complete");
  return 0;
}

```