OpenMP parallel for scheduling

The OpenMP parallel for worksharing construct supports different types of scheduling. The objective of this exercise is to define a scheduling strategy that is best suited for a specific code, propose a parallelization and analyze the resulting behavior.

1 Package content

The `Schedules` directory contains a single file named `schedules.c`. The relevant part of this file is represented by the following two lines of code

```
for (n=size; n>=100; n-=100){
  mat_mult(n, A+ptr[n], B+ptr[n], C+ptr[n]);
```

At every iteration of this loop a matrix-matrix product of the form $C = A \cdot B$ is performed, where A, B and C are dense matrices of size `n`. Note that the A, B and C matrices are different at every iteration and thus there are no loop-carried dependencies. In the code, `size` is set to be equal to 1500.

The code can be compiled using the `make` command: just type `make` inside the `Schedules` directory. This will generate an executable `main` file that can be launched like this

```
$ ./main
```

Upon execution, the `main` program will print the execution time for the loop above.

2 Assignment

-  assuming the OpenMP parallel for construct has been used to parallelize the loop above, identify the scheduling type that is best suited for this code and explain your choice in the `responses.txt` file. Report in this file also the execution time of the code using 1, 2 and 4 threads with the scheduling type you have chosen and with the default, static scheduling. The observed results confirm what you expected?
-  use the OpenMP parallel for construct to parallelize the loop above in the `schedules.c` file. Compile and run the program using 1, 2 and

4 threads, with both the scheduling type you have chosen and the default static scheduling; report the corresponding execution times in the `responses.txt` file as described above.