CPU- and Runtime analysis

Introduction/Justification

It is of interest to the user to know, not only the accuracy of a given program but the calculation cost of using a given method, be it difference schemes, relaxation metho adaptive meshing.
A way of quantifying the calculation cost of an algorithm is by measuring the time it tak run it. There are at least two ways in which this can be done: the runtime and CPU-time give the required estimate. The runtime is the time as if calculated by a person timing th of the algorithm while the CPU-time is the time that is allocated in the processor for the in mind. The CPU-time will always be less than or equal to the runtime, as the runtime w affected by other processes that might be using the same processor.[**?**]

Method

The different numercal methods were thus applied on a test matrix; a simple matrix posed of two small rods inside a parallel-plate capacitor. 25 runs, call it a bundle of ru a given algorithm were clocked to diminish any fluctuations in current processes being dled. From this, the average CPU- and runtime (henceforth referred to as "time" unles erwise specified) was calculated for each run. Each bundle was run ten times from the mean and variance were calculated, call this one trial. Trials were run for iteratio ues ranging from 130 to 600 with increases of ten iterations. For each trial, dividing b number of iterations gave us a mean time per iteration, $\mu^{t/n}$, where t is the time and notes that this is per iteration. For the variance, the median, rather than the mean, was as this quantifies the uncertainty in the time measurement. The timing functions used chrono::high$_{r}esolution_{c}lockforruntimeandstd :: clock()forCPU - time$

Results

Below follows a table holding the results of the trials:

Table 1: Mean runtime per iteration for three different methods

	5 point - Jacobi	**9 point - Jacobi**	**9 point - Gauss-Seide**
CPU-time(ms):	0.11853	0.192614	0.19257
Runtime(ms):	0.512	0.9164	0.9167

Conclusions and discussion

The results are as expected: there is a difference between the 5- and 9-point difference sch while there is no percieved difference between the tested relaxation methods. The 9- difference method involves more points to calculate the potential in an element than the 5- does, hence the longer time per iteration. Meanwhile, the difference between the Jacobi an Gauss-Seidel methods is from where information is retrieved; the Gauss-Seidel retrieves data from the current iteration while the Jacobi method uses all information from the pre iteration. As neither involve more calculations than the other, the results show what is expe that the time per iteration is the same within their variance.

12 - Mar - 2105
Intel Performance Counter Monitor - A better way to measure CPU time

clock() function
https://www.gnu.org/software/libc/manual/html$_node/Processor-And-CPU-Time.htm$
$And-CPU-Time$
$R.McGrathetal$
$12-Mar-2015$
$TheGNUCLibraryReferenceManual$

CAN THIS BE USED???? chrono::high$_resolution_clock$
$http://www.cplusplus.com/reference/chrono/high_resolution_clock/$
N/A
$12-Mar-2015$