experience with speeding up the wave equation:

Project in INF 3331

Yapi Donatien Achou
Department of mathematics
University of Oslo
P.O Box 1072 Blindern, 0316 Oslo, Norway

October 31, 2012

1 Introduction

2 Mathematical model

We are interested in solving the 1D and 2D wave equation

$$u_{tt} = c^2 \Delta u \tag{1}$$

Where the initial condition for $1D$ problem is given by

$$u(x, 0) = sin(2\pi x) \tag{2}$$

and the initial condition for the $2D$ problem is given by.

$$u(x, y, 0) = 2sin(0.25\pi x)sin(0.25\pi y). \tag{3}$$

For both problems, the boundary condition is given by

$$u(0, t) = 0 \tag{4}$$

$$u(0, y, t) = u(x, 0, t) = 0 \tag{5}$$

3 Methods

The $2D$ wave equation is discretized using an explicit finite difference method. The discretized equation is then implemented in python, vectorized python and in C. We then perform some numerical experiments and compare the execution time of 1d wave equation versus 1d vectorized wave equation in python, 2d wave equation in python versus vectorized 2d wave equation in python and 2d vectorized wave equation versus 2d wave equation in pure c. To appreciate the value of the constant c in the wave equation, we fixed the time t in the 1d wave equation and vary the constant c.
We defined the speed up as the fraction of the CPU time for pure python to vectorized python and the fraction of CPU time of vectorized python to pure C.

4 Result

4.1 Speed up of 2d wave equation in python by pure C

Table 1: CPU time for vectorized python and pure C for 2D wave equation $dt = dy = dx, c = 0.0001, t = n$

mesh size n	CPU time vectorized python	CPU time pure C
100	0.095	0.058
200	1.70	0.33
300	8.50	0.94
400	21.50	2.07
500	43.27	3.85
600	73.02	6.42
700	116.97	9.95
800	170.04	14.54
900	244.00	20.43
100	333.53	27.63

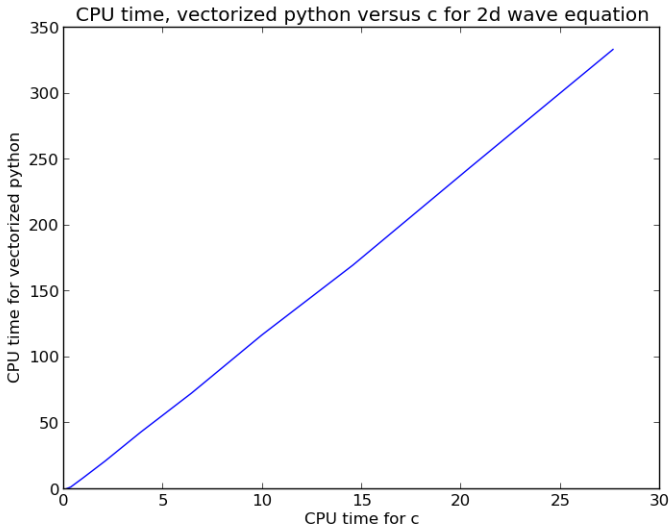

Figure 1: CPU time of vectorized python versus pure C for 2D wave equation

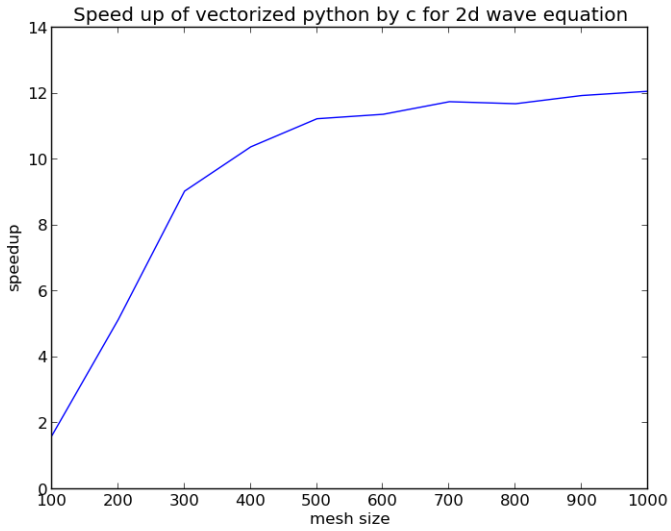

Figure 2: Speed up of vectorized python by C for 2D wave equation

4.2 Speed up of 2d wave equation by vectorized python

Table 2: CPU time for pure python and vectorized python 2D wave equation, $t = 2, c = 0.1$

mesh size	CPU time pure python	CPU time vectorized
5	0.001	0.00044
10	0.01	0.00062
20	0.081	0.001
50	1.13	0.043
100	9.29	0.02
500	1161.9	9.5
1000		76.42

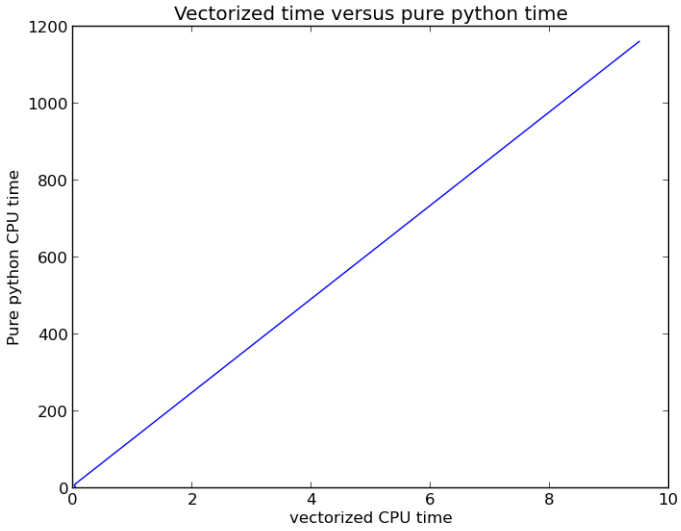

Figure 3: Pure python CPU time versus vectorized python CPU time

4.3 Speed up of 1d wave equation by vectorized python

Table 3: CPU time for pure python and vectorized python 1D wave equation, $t = 2, c = 1.5$

mesh size	CPU time pure python	CPU time vectorized
5	0.0016	0.0007
10	0.0024	0.00123
20	0.01	0.0024
50	0.06	0.0063
100	0.26	0.012
500	6.76	0.36
1000	27.29	1.69

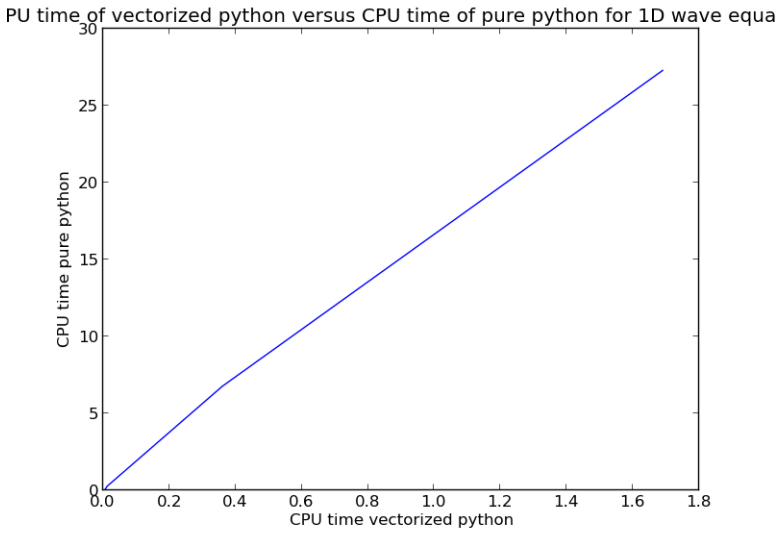

Figure 4: Pure python CPU time versus vectorized python CPU time for 1D wave equation

5 Solution of 1D wave equation with changing coefficient c

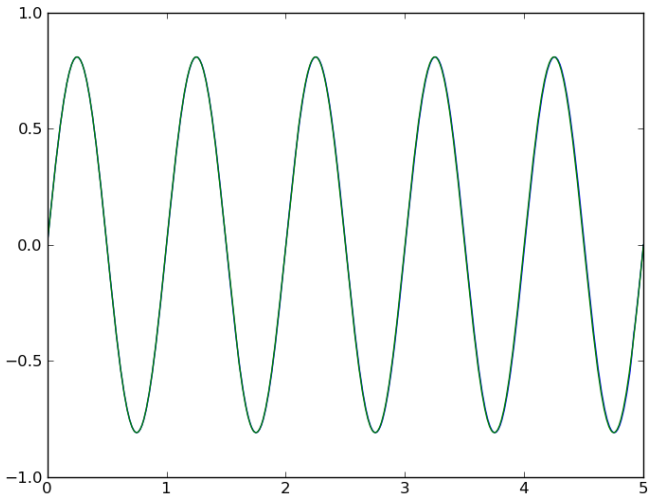

Figure 5: 1D wave equation at t=1, with c = 0.1,analytical solution and numerical solution

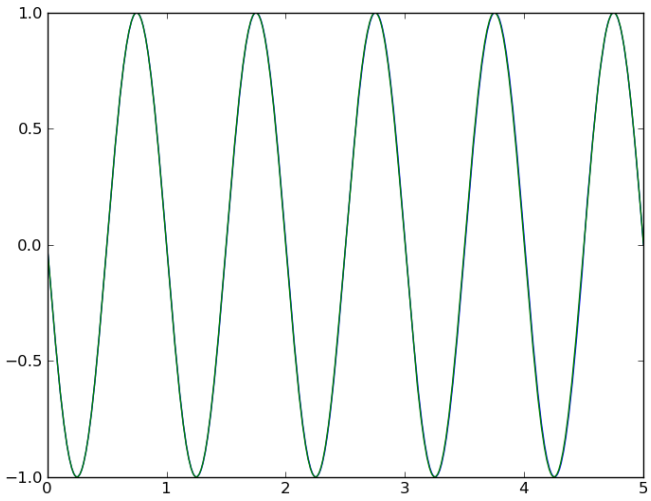

Figure 6: 1D wave equation at t=1, with c=0.5, analytical solution and numerical solution

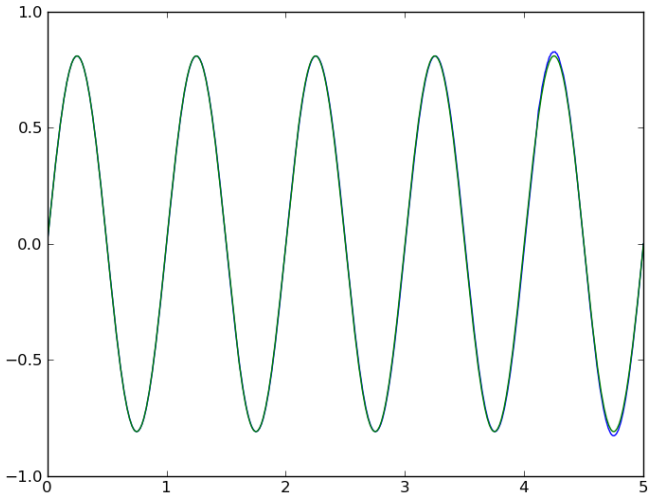

Figure 7: 1D wave equation at t=1, for c=0.9, analytical solution and numerical solution

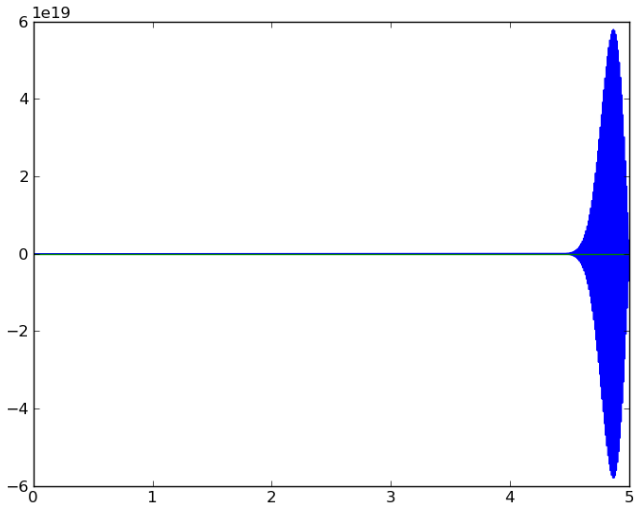

Figure 8: 1D wave equation at t=1, for c=1.01, analytical solution and numerical solution

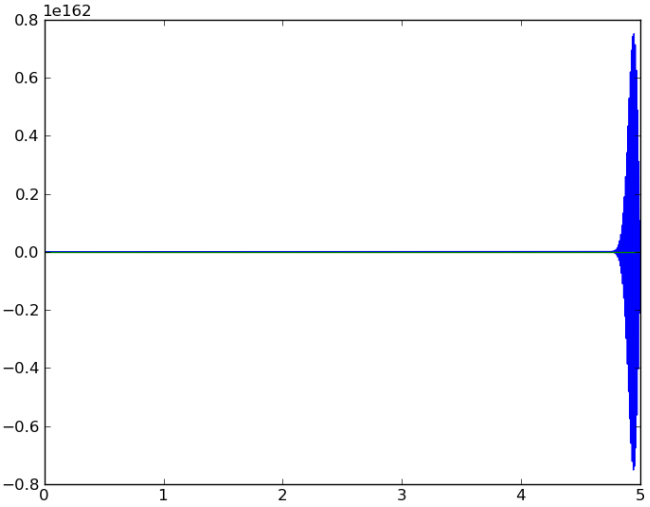

Figure 9: 1D wave equation at t=1, for c=1.5, analytical solution and numerical solution

6 conclusion and remark

The 1D and 2D wave equation was solved using an explicit finite difference method. By vectorizing the loops in python, we observed a significance speed up. Interpreted programming languages such as python gives a flexibility to the programmer when implementing a numerical scheme. However, this flexibility induce extra time at run time. Compiling programming languages such as C are more demanding during implementation, however, this pays off during run time, making compiling programming languages faster than interpreted programming language at run time.
Due to stability criteria, the constant c in the wave equation must carefully chosen, for stable solutions.