Computational power needed to propagate a non-relativistic n-particle wave function

Ulrich Mutze 2014-04-18

Introduction

Making quantum mechanics finite dimensional by defining wave functions on a finite lattice of points is a self-suggesting trick for visualizing quantum dynamics on a PC. I have collected extensive experience with treating one to a few particles on one-dimensional lattices that way [1 - 4]. What this approach shows is that it is obvious how such programs can be extended to 3 space dimensions and to arbitrary particle numbers but that they than run so slowly that it is very hard and soon plainly impossible to generate useful information by running them.

It might be useful to point out, that precise coincidence of the numerical wave function dynamics with the one defined by some continuously defined Schrödinger equation is not a concern in such simulations. The famous image series Fig. 16 - Fig. 19 in the quantum mechanics text book by Schiff [5] which shows how a Gaussian wave packet behaves when passing a potential barrier and a potential well conveys the idea and lets our mental eye see details which are hidden from direct observation. If the aim is to extract quantitative information, it is already in classical many-particle simulations not the individual trajectory that counts but characteristics involving a huge number of trajectories. An example is the time which an arbitrary collection of moving particles in a container needs to reach thermal equilibrium [6].

Besides the role as an educational demonstrator such simulations may be attributed a significance that, although often present as a background reasoning, deserves to be made fully explicit. They can demonstrate that a Hermitian linear operator is all one needs to define executable quantum dynamics. All integration methods which assume a particular expression structure of the Hamiltonian and work by surgery on that expression, or use a priori knowledge on the intermediary states that have only to be taken into account, are thus seen not to be needed for the automatic and autonomous working of Nature. They only are devices invented by humans to reduce the computational burden imposed by the universal and natural evolution algorithm. What I consider as the universal and natural algorithm is the second order accurate leapfrog method [7], for which I developed an elegant, efficient, and slightly more accurate variant under the name 'asynchronous leapfrog method' [7-10]. For a 'universal and natural' integration mechanism it is important that it works autonomously on all states and is unable to create ever an error condition such as an numerical overflow. Such an algorithm for non-relativistic quantum mechanics of n-particle systems may comprise idealizations such as treating atomic nuclei as stationary, ignoring the effect of magnetic moments etc. Nature is relativistic anyway, but nevertheless most physicists expect that the non-relativistic idealization (together with optional idealizations as just mentioned) defines a consistent ‘model-world’ for which one should be able to formulate rules which define from any given state the passage to ‘the next state’, i.e. just what was called a universal and natural integration mechanism above.

Assessing the computational resources required by the universal and natural integration mechanism for a semi-realistic many-electron system

We consider a system of *n* particles which are enclosed in a cubic volume of side *a*. Actually, we are a bit more specific in considering a cube of metallic silver and treat the atoms as stationary so that the wave function of the whole system only depends on the positions and the spins of the conduction electrons.

For applying the simple leapfrog method for propagation we replace continuous space by a cubic lattice of lattice spacing Δx. Not ignoring the spin of the electron, we get for the state space of a single electron the dimension $d_1 = 2\left(\frac{a}{\Delta x}\right)^3$. Then the dimension of anti-symmetrized wave functions of *n* electrons is known to be given as the binomial coefficient $\binom{d_1}{n}$. In the following we do the computational steps by means of *Mathematica* in a way that the side of the cube is treated as a variable and all other data are either natural constants or the values referring to our silver system. All values for physical quantities refer to SI units. First we introduce Planck’s constant, the mass of the electron, and the size of a cell of the silver lattice:

```
In[95]:=  h = 6.62607 × 10^-34
Out[95]=  6.62607 × 10^-34
```

```
In[96]:=  me = 9.10938 × 10^-31
Out[96]=  9.10938 × 10^-31
```

```
In[97]:=  unitCellLengthAg = 408.6 * 10^-12
Out[97]=  4.086 × 10^-10
```

The computational lattice for the electron wave function should be significantly finer than the atomic lattice. Of course, the finer the computational lattice, the larger the computational burden. Since, in the end, we will find the computational burden to be prohibitively large, we will here and in following steps systematically under-estimate the computational burden which then makes the final result even more conclusive.
In this spirit we chose a lattice spacing Δx given as

```
In[98]:=  Δx = 0.1 * unitCellLengthAg
Out[98]=  4.086 × 10^-11
```

Particle density ρ of conduction electrons is 6/2 + 8/8 = 4 silver atoms per unit cell (we have a face-centered cubic lattice) and thus also 4 conduction electrons

In[99]:=
```
ρ = 4. * unitCellLengthAg^-3
```
Out[99]=
```
5.86361 × 10^28
```

The number of conduction electrons in a cube of side *a* is thus given by

In[100]:=
```
np[a_] := ρ * a^3
```

The dimension of the whole state space is (as argued at the beginning of the section)

In[101]:=
```
d[a_] := Module[{n, d1}, n = np[a]; d1 = 2 * (a / Δx)^3; Binomial[d1, n]]
```

As an example let us consider a silver cube with side of one nanometer. To make the growth of the computational burden with size evident, we consider also values which are 10% and 20% larger :

In[102]:=
```
a0 = {1. * 10^-9 , 1.1 * 10^-9, 1.2 * 10^-9}
```
Out[102]=
```
{1. × 10^-9, 1.1 × 10^-9, 1.2 × 10^-9}
```

Here we see how the dimension of the state space grows:

In[103]:=
```
d[a0]
```
Out[103]=
```
{2.59119 × 10^182, 1.42963 × 10^243, 1.06242395640763 × 10^316}
```

The time step Δt for the time-discrete propagation has to be chosen such that for any state (and not only for the initial state of some specific situation of interest) the evolution steps of the leapfrog method can be repeated arbitrarily often, and nevertheless the norm of the state and the expectation value of the Hamiltonian are approximately constant. Actually they show a tiny wiggle deviation. The mean amplitude of this deviation is proportional to Δt^2). This can be shown to be the case for $\Delta t \, \| H \| < h$, where $\| H \|$ is the operator norm of the Hamiltonian [1,8]. What follows is an estimate of this operator norm. The potential energy is neglected here as unimportant against the kinetic energy of conducting electrons. This kinetic energy is obviously limited by the shortest wave length $2\,\Delta x$ that can be represented on our computational lattice. As shown in [1], for any bounded linear operator the norm can be found by an algorithm which is not more expensive than running a short simulation in which this operator is taken as Hamiltonian.

In[104]:=
```
normH[a_] :=
 Module[{pMax, λMin}, λMin = 2 * Δx; pMax = h / λMin; np[a] * pMax^2 / (2 * me)]
```

For the time step we get the upper limit

In[105]:=
```
dt[a_] := h / normH[a]
dt[a0]
```
Out[106]=
```
{3.13152 × 10^-19, 2.35275 × 10^-19, 1.81222 × 10^-19}
```

Each leapfrog step needs only one application of the Hamiltonian to the state and nevertheless it is second order accurate. In the spirit of under-estimation we count this application simply as d^2multiplications just as if the Hamiltonian would be given as a *d* times *d* matrix with pre-determined matrix elements. An algorithm which can be followed by Nature ‘in principle’ cannot work with pre-computed matrix elements since these may depend on external fields which may change with time even if they were strictly constant so far. So we had certainly to add some effort for computing the matrix elements. On the other hand, all the methods developed for treating many particle systems in interesting specific situations avoid to get involved in the manifold of all matrix elements of the Hamiltonian by problem driven sampling techniques. This prevents them from working autonomously in the general case. As emphasized already, this autonomous working in the general case is the task with which we see Nature confronted. To the degree to which we can see algorithms at least in some metaphorical sense as the means that Nature has for accomplishing this task, the simplest autonomous and universal algorithm that we humans can device may be indicative for what Nature actually does in this situation. For our autonomous and universal leapfrog-based algorithm we therefore count d^2 floating point operations per time Δt and thus have for the computational power, measured in floating point operations per second (flops), the expression

```
In[107]:= cp[a_] := d[a]^2 / dt[a]
```

and the values for our special case

```
In[108]:= cp[a0]

Out[108]= {2.144098047909444 × 10^383, 8.686989510505362 × 10^504, 6.228519685185083 × 10^650}
```

These are huge numbers. To bring them closer to something imaginable we subdivide our silver nano cube into tiny sub-cubes, the sides of which equal one Planck length, i.e.

```
In[109]:= lP = 1.616 × 10^-35

Out[109]= 1.616 × 10^-35
```

Considering each such sub-cube as a computer we compute how may flops each of these ‘elementary computers’ has to deliver. This may be called the ‘relative computational power cpr

```
In[110]:= cpr[a_] := cp[a] * (a / lP)^-3
```

which for our silver cube still has huge values:

```
In[111]:= cpr[a0]

Out[111]= {9.04834 × 10^305, 2.754325804710774 × 10^427, 1.521125940187467 × 10^573}
```

The most powerful computer on Earth which presently is known to Wikipedia is the Tianhe-2 in China with a peak computational power of

```
In[112]:= cpTianhe2 = 54.9 * 10^15

Out[112]= 5.49 × 10^16
```

flops. Our Planck length ‘elementary computer’ has thus to outperform the Chinese machine by a factor

```
In[113]:= cpr[a0] / cpTianhe2

Out[113]= {1.64815 × 10^289, 5.016986893826548 × 10^410, 2.770721202527263 × 10^556}
```

which is still huge and definitively says that Nature does not follow this logic. The dynamical laws of non-relativistic quantum mechanics for many-particle wave functions cannot be obeyed literally ! Of course, there is the idea that Nature achieves the dynamical evolution of wave functions somehow, and therefore is shown to have a computational power which exceeds that of digital classical computers by huge factors. If this would actually be the case, one would be well advised to try to build computers which make use of these fantastic capabilities of Nature. Time will tell whether this works out.

References

In[114]:=

```
[1] Ulrich
 Mutze : The direct midpoint method as a quantum mechanical integrator (2006)
http : // www.ma.utexas.edu / mp_arc / c / 06 / 06 - 356. pdf
```

In[114]:=

```
[2] Ulrich
 Mutze : The direct midpoint method as a quantum mechanical integrator II (2007)
http : // www.ma.utexas.edu / mp_arc / c / 07 / 07 - 176. pdf
```

In[114]:=

```
[3] Ulrich Mutze : Separated quantum dynamics (2008)
http : // www.ma.utexas.edu / mp_arc / c / 08 / 08 - 69. pdf
```

```
[4] Ulrich Mutze (2011) :
 "Relativistic Quantum Dynamics in 1D and the Klein Paradox" from
The Wolfram Demonstrations Project.
   http : // demonstrations.wolfram.com /
   RelativisticQuantumDynamicsIn1DAndTheKleinParadox /
```

```
 [5] Leonard I. Schiff : quantum mechanics, third edition, McGRAW - Hill 1968
```

```
[6] Ulrich Mutze and Stephan Leibbrandt (2012) :
 "Approach of a System of Particles towards Thermal Equilibrium" from
The Wolfram Demonstrations Project.
   http : // demonstrations.wolfram.com /
   ApproachOfASystemOfParticlesTowardsThermalEquilibrium /
```

In[114]:=

```
[7] Ulrich Mutze : An asynchronous leap - frog method (2008)
http : // www.ma.utexas.edu / mp_arc / c / 08 / 08 - 197. pdf
```

In[114]:=

```
[8] Ulrich Mutze (2011) : "On the Stability Limit of Leapfrog Methods" from
The Wolfram Demonstrations Project.
   http : //
 demonstrations.wolfram.com / OnTheStabilityLimitOfLeapfrogMethods /
```

In[114]:=

```
[9] Ulrich Mutze : An asynchronous leapfrog method II (2013)
http : // www.arxiv.org / abs / 1311.6602
```

In[114]:=

```
[10] Ulrich Mutze (2013) :
 "The Asynchronous Leapfrog Method as a Stiff ODE Solver" from
The Wolfram Demonstrations Project.
http : // demonstrations.wolfram.com /
   TheAsynchronousLeapfrogMethodAsAStiffODESolver /
```