Web Based Real Time System for Wavepacket Dynamics

Aleksander Nowiński and Krzysztof Nowiński
ICM Warsaw University
Pawińskiego 5a, 02-106 Warsaw, Poland
{axnow,know}@icm.edu.pl
and
Piotr Bała
Faculty of Mathematics and Computer Science
N. Copernicus University
Chopina 12/18, 87-100 Toruń, Poland,

Abstract. In this paper we describe prototype systemm for web based interface to numerically intense simulations. Presented system solves time-dependent Schroedinger equation using wavepackets. Proposed solution consists from three main parts: client which acts as graphical user interface, server which receives requests form various clients and handles all running jobs. The last part is computing engine which can be treated as external application and in principle can be written in any computer language.

1 Introduction

The history of computer simulations in various areas of physics is relatively long and successful. The increasing rapidly computer power allows nowadays for routine simulations of complicated systems, or allows for advanced theories and models. In the past computers were used for simulations of atomic and molecular systems, for modeling of sets of interacting particles [1], fluid dynamics, quantum chromodynamics and many others [1, 2]. In last ten years available computer power allowed for calculations of wavepacket dynamics obeying time dependent Schroedinger equations, first for one dimensional systems[3] and nowadays for the systems with few degrees of freedom [4].

Developed numerical methods allowed for simulation of quantum evolution of different systems and processes, especially in this cases where classical approximation cannot be used. Numerical simulations in this area are also important because of lack of analytical methods, especially because of lack of analytical solutions of time-dependent Schroedinger equation for nontrivial potentials. The quantum evolution can be predicted analitycally only for very simple potentials (e.g. constant or harmonic), or in very special cases - for example for stationary solutions.

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2329, pp. 552–561, 2002.

Progress in experimental techniques, as well as new phenomena discovered, implicated interest in simulations of dynamics of quantum systems. The most important examples are photoionization of atoms[5] and molecules[6], dynamics of simple molecular systems[], electron tunneling in superlattices [], proton tunneling in molecules and biomolecules, photo adsorption and desorption[] or, recently, dynamics of Bose Einstein condensates[7, 8].

The wavepacket dynamics based on integration of time-dependent Schroedinger equation, especially in more than one dimensional case can be run only on high performance computers. The existing codes are developed by the research groups and are, in general, not user friendly and have limited visualization capabilities. This is caused by the need of high numerical efficiency of the code, which is, in most cases, run in a batch. One should note, that interpretation of the wavepacket dynamics usually cannot be performed without aid of advanced visualization tools during both simulation setup and analysis. This requires development of coupling mechanism between visualization and simulation codes.

Recent progress in computer technology, especially web based tools makes them good candidate for development of user interfaces to computing programs. Web based interfaces are easy to install, simple to use and can be used in distributed environment. However, even providing powerful user interface, web technology cannot achieve high numerical efficiency required for most simulation problems. This leads us to solutions consisting on web interface for the user and traditional, usually batch type, numerical simulation engine. Web access to high performance resources is being developed significantly in last few years. However most of the development goes to the design and production of general tools which can be used in different situations. Because of this, security is one of the most important and yet not solved issues. The another one is accounting, resource directory services or resource reservation mechanisms. The existing solutions like globus[9], Legion[10] or Unicore[11] address most important issues, Unfortunately, these solutions cannot efficiently create strongly coupled application, especially in the case when advance interactive visualization techniques must be used.

Recently we started effort to create an prototype of distributed computing system with Web interface dedicated to wavepacket dynamics. System is based on computing server, or number of servers, located locally or remotely, eg. in supercomputer centre, which offers computing resources for clients distributed all over the world through internet. In general, user should contact with computing server using web technology, with tools allowing for easy preparation of input data, job submission and steering, and analysis of results.

2 Wavepacket Dynamics

The dynamics of the quantum particle can obtained by solving the time–dependent Schroedinger equation:

$$i\hbar \frac{\partial \Psi(x,t)}{\partial t} = \mathbf{H}\Psi(x,t), \tag{1}$$

with the Hamiltonian:

$$\mathbf{H} = -\frac{\hbar^2}{2m}\Delta + V(x). \tag{2}$$

where m is particle mass.

In general, the time–dependent Schroedinger equation (1) cannot be solved analytically. Recently an effective and powerful numerical methods have been developed and used to investigate different physical problems such as scattering, electron and proton tunneling, photodesorption or photoionization.

As mentioned above, the analytical solution of (1) is not known and numerical methods based on the discrete representation must be used. After a small time step Δt the propagated wavefunction can be obtained by using the time–evolution operator $\mathbf{U}$:

$$\Psi(t+\Delta t) = \mathbf{U}(\Delta t)\Psi(t) = \mathbf{T}e^{-\frac{i}{\hbar}\int_t^{t+\Delta t}\mathbf{H}dt'}\Psi(t), \tag{3}$$

where $\mathbf{T}$ is the time ordering operator. When the Hamiltonian $\mathbf{H}$ is time–independent the time ordering operator can be omitted and the integral can be approximated with the trapezoidal rule:

$$\mathbf{U}(\Delta t) \cong e^{-\frac{i}{\hbar}\mathbf{H}\Delta t}, \tag{4}$$

Once the evolution propagator $\mathbf{U}(\Delta t)$ and its action on the wavefunction are known, the dynamics of the system is obtained and the time–dependent Schroedinger equation (1) is solved.

The different methods has been proposed for the calculation of $\mathbf{U}(\Delta t)$. The most commonly used are: second–order differential scheme [3, 12, 13], split operator method [14–16], Calay method [17, 18], Chebychev polynomial expansion [19, 20] or Lanczos scheme [21–23]. In a particular case an algorithm based on the Chebychev polynomial expansion has been used, which was found to be stable and accurate [13, 24, 25]. The time evolution operator is expanded into Chebychev polynomial series:

$$U(\Delta t) = \sum_{m=1}^{\infty} a_m \Phi_m(-i\mathcal{H}) \cong \sum_{m=1}^{M} a_m \Phi_m(-i\mathcal{H}), \tag{5}$$

where Φ_m are complex Chebychev polynomials and a_m are expansion coefficients.

In numerical applications, the wavefunction $\Psi(t+\Delta t)$ as well as potential for electron motion are represented on a discrete grid. The grid points are usually equally spaced by the Δx, although adaptive grid methods are also possible. The discretization of the wavefunction introduces some limits on the maximal momentum p_{max} represented on the grid:

$$p_{max} = \frac{\pi}{\Delta x}. \tag{6}$$

The momentum limit implies that the energy range of the discretized Hamiltonian is given by the maximal and minimal energy represented on the grid:

$$\Delta E = E_{max} - E_{min}, \tag{7}$$

where:

$$E_{min} = V_{min}, \qquad E_{max} = V_{max} + \frac{p_{max}^2}{2m^*}, \tag{8}$$

and V_{min} and V_{max} are the minimum and maximum of the potential on the grid.

The potential part of the Hamiltonian can be calculated directly as multiplication of the potential energy and the wavefunction. The most powerful and accurate method of the calculation of the kinetic energy operator is based on the transformation of the wavefunction to the momentum space using the Fourier Transform technique [3]. Since the effect of the operator $\mathbf{H}$ on $\Psi(x,t)$ is calculated using the Fast Fourier Transform (FFT) twice, the described method requires 2M FFT's per time step. The number of FFT's in the Chebychev polynomial method is greater than in other methods of solving the time–dependent Schroedinger equation but longer time steps Δt can be applied and similar computational efficiency is achieved. This method is also found to be most stable and accurate [13, 25].

3 Description of Web Based Numerical Simulation System

System for web access to computer resources for wavepacket dynamics is basically designed as three-part construction - the client, central server (dispositor) and computing server. There can be many clients and many computing servers but only one central server per computing site. This solution is natural, because of the security control at computing site.

3.1 Client

Client is the application that prepares data for computation, sends the request to the central server and collects results. Typically (as in created example) it also offers limited capabilities of result visualization. What is most important, client program is an WWW applet written in Java, so it can be run from any workstation with java capable Web browser. It was tested with Sun Sparc Ultra (Solaris 2.8), Intel PC (Windows 98), Intel PC (RedHat Linux 6.2), Silicon Graphics 02 (Irix 6.5), and no problems were noticed.

The client applet has an visual editor capable to define wave function and any analytic potential on one dimensional computational grid. It also allowed to change some computation parameters, like mesh density, or number of time steps. The client acts as main graphical user interface. Because of required functionality any other existing software could be used. The input of parameters and visualization requirements for wavepacket dynamics cover quite wide range of possible applications and we have decided to spent some effort to develop interface dedicated to such type of simulations. One should note, that client can be

also used as user interface for other classes of simulations which are generally based on the mesh. This will however require some additional customization of the client.

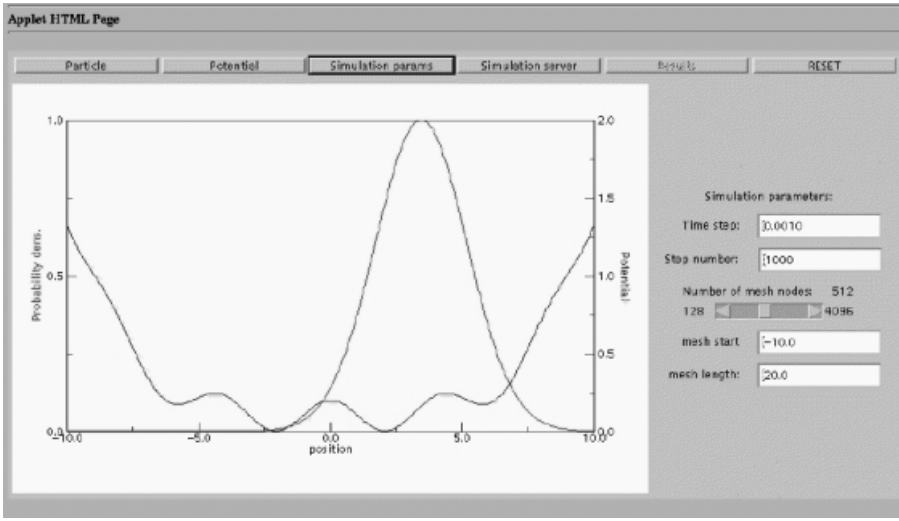

Fig. 1. The Client applet - grid parameters control screen.

3.2 Central Simulation Server

Central server (dispositor) is an application that receives all client requests, and forwards it to proper computation servers. It can provide more than one computation service, so the same dispositor can serve for different remote computation programs, and can be used to provide wide range of services. One computational service is typically a program that can receive remote data and send the result. The program is optimized to gain maximal possible performance and usually is designed to run in batch queue.

The dispositor after receiving request decides which computation server to use (there may be more than one computation server), and forwards the client data to it. Then it returns data back to the client. This model is necessary for Web-based computing, because of security limitations of WWW applets. It also benefits in central management and - what is more important - it hides all the computation from client. Such design simplifies client role: client does not have to know many computation server addresses, but only one - the dispositor address. In case of any system changes for example adding and removing computation servers, client may even not know about it. It also allows to implement basic load

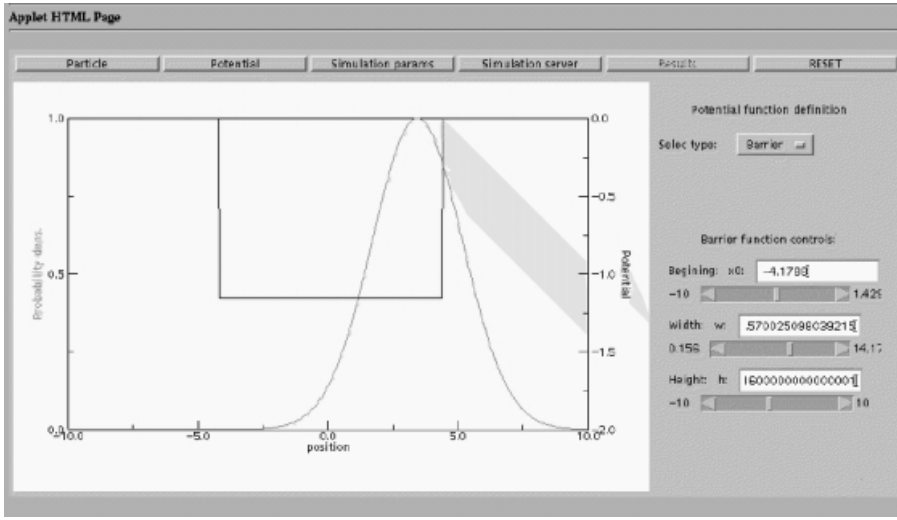

Fig. 2. The Client Applet - change of square well potential parameters.

balancing algorithms, that make system far more effective. At the moment there is no load balancing implemented and computation server is chosen randomly from the list. In the limited access mode, computation dispositor may also have abilities to make client authorization. The last, but not least benefit of the model is that there is only one channel to open in firewall of computing centre which simplifies significantly security management.

3.3 Computation Server

Computation server is rather simple application that receives form dispositor data containing request, and then it runs computation program and finally sends results back to the dispositor. Server is very simple and because of that it's small and fast, not requiring computation power, that must be used for computation program. This allows for server installation on the computer other than computational engine.

At this moment server is separated from computation program, so main numerical code was not changed at all - it *doesn't know* that it's run from the web, not locally. This has strong practical implications: server can be used for any existing computational codes without significant modifications of the numerical engine.

Server has a limit of number of computations that can be run parallelly, to avoid overloading. This this is implemented as system of "tokens", and is also used at the dispositor to determine which machines are free for computations.

3.4 Communication

Network layer is based on standard TCP/IP socket connection and a simple communication protocol over TCP. We decided not to use Java RMI, because in case of running the system on supercomputers, that doesn't have any Java implementation we would have to rewrite the computation server in C, without RMI. It also makes possible to write clients in other languages for example as plugins to data visualization programs or as stand alone applications.

The main goal was to provide user with real-time, interactive simulations of wavepacket dynamics, and therefore existing solutions based on the batch processing cannot be used. Developed computation server is design to start simulations imidiatelly on less loaded machine in the pool, provides user with partial results as soon as they are available.

General scheme of computation is that user prepares his job with client. Then client connects to a dispositor, and requests computation service. Then dispositor decides if there are available computation servers - finds proper token for the service. Then it accepts client's request and receives the job data, and forwards it to the proper server. Computation server receives the data, and uses it to initialize computation (for example writes input data onto disk as files). Then computation server starts to send results as fast as computation program produces it. Dispositor then forwards result to client. This allows for real time visualization of results as soon as they available on the computational server. If the client disconnect job is aborted, and if job has been finished dispositor and computation server disconnects. At this moment there is no possibility of connecting to a running job or reconnection. Client receives results and presents it to the user.

4 Details of System Design

All parts but computation program are written in Java (TM), what makes possible to use whole system on very wide set of machines, without concerning about portability and language implementation details. Effectively it means that we can use cluster of different workstations for different computations, and clients can use in fact any workstation without architecture or operating system limits.

All parts of the system were designed as modules, so in all programs are easily separable modules, responsible for protocol, for request creation and so on. Every module can be easily exchanged and extended, so it's easy to improve system. It was important to make configuration of the system easy and elastic - all servers use an XML configuration files, which provide easy configuration in human-readable form.

The developed system is fully functional and is its applications are not reduced to the wavepacket dynamics. It is capable to serve computations and to perform on line numerical processing such as remote data visualization, access to parts of numerical data and so on. It can be easily adopted for different services. Generally, adding new service to the system requires three steps:

1. Providing computation program,
   Program can be written in any language, in general modifications to the code are not required.
2. Adopting service provider - part of computation server.
   This modifications are small and can be easily performed.
3. Preparing client software
   Client software can be written in any language, however communication with the computing server must be added. Presently this is written in Java and is available in form of library which can be used for different clients.

All this tasks can be easily performed.

5 Conclusions

Developed systems for web based real time wavepacket dynamics proves that it's relatively easy to build small and medium scale system that can be used for serving small and medium scale remote job processing. Presented system is functional and proves that web based real time simulations can be efficiently performed even in the area dominated by the batch processing. The modifications to the existing simulation programs are very limited, in practice, most of existing codes written in Fortran or C can be used as is. Modular approach opens system for other application areas as well as for further development.

It could be used as computation server distributing workstation-scale computations between machines available inside the computing center, as an remote access for custom visualization of parts of the numerical data stored in the center for example for meteograms from weather simulations, or medical images. Nowadays system is used for educational purposes as easy to learn interface to advanced simulations in quantum physics.

Second conclusion is that it's possible to extend the system for classic queuing systems, allowing clients to prepare larger jobs and then disconnect and reconnect to collect results. This, connected with well prepared load-balancing may create nice compact grid application.

Last conclusion is a side effect of project, but we think that it may be most important for future work. The system we used did not involved any modification of the numerical code. We simply took Fortran code, recompiled it for desired architecture and used program as it was - as *black-box* application. It was possible to implement kind of grid without any modifications. What is most important we were able to receive results on line, in time of the running job. This is typical for advanced numerical grid applications, with special network code and carefully prepared for this task (see VisIt, Globus). In our system it was done far simpler way - we connect not to an application but read output files. This approach has three main consequences:

- very limited interaction with application,
- no need of any numerical code modification,

- no special skill required from numerical programmer required to prepare new application for the system.

In the present design one has no way to interact (as other grid systems do) except halting the job - but we assume that vast majority of the numerical application are absolutely non interactive, and will not be. What one gets is fact that one can use code as it is. The only recommendation, but not requirement, is that it's better to receive output as an ASCII text or in machine independent format. New developers of the simulation code don't have to learn new skills like network programming, what shall not be underestimated.

Acknowledgements This work is partially supported by European Commission under IST grant 20247. The computations were performed at the ICM, Warsaw University.

References

1. J. W. Eastwood R. W. Hockney. *Computer Simulation Using Particles.* Cambridge University Press, Cambridge, 1987.
2. M. M. Woolfson and G. J. Pert. *An Introduction to Computer Simulation.* Oxford University Press, 1999.
3. D. Kosloff and R. Kosloff. A Fourier method solution for the time dependent Schroedinger equation as a tool in molecular dynamics. *J. Comput. Phys.*, 52:35–53, 1983.
4. D. H. Zhang, Minghui Yang, and S-Y. Lee. Branching ratio in the HD+OH reaction: A full-dimensional quantum dynamics study on a new ab initio potential energy surface. *J. Chem. Phys.*, 114(20), May 2001.
5. J. H. Eberly, R. Grobe, C. K. Law, and Q. Su. Numerical experiments in strong and super-strong fields. In M. Gavrila, editor, *Atoms in string fields.* Academic Press, San Diego, 1992.
6. L. Pesce, Z. Amitay, R. Uberna, S. R. Leone, M. Ratner, and R. Kosloff. Quantum dynamics simulation of the ultrafast photoionization of Li_2. *Journal of Chemical Physics*, 114(3):1259–71, Jan 2001.
7. P. A. Ruprecht, M. J. Holland, K. Burnett, and M. Edwards. Time-dependent solution of the nonlinear Schrodinger equation for Bose-condensed trapped neutral atoms. *Phys. Rev. A*, 51(6):4704–11, 1995.
8. R. Dum, A. Sanpera, K.-A. Suominen, M. Brewczyk, et al. Wave packet dynamics with Bose-Einstein condensates. *Phys. Rev. Lett.*, 80(18):3899–902, 1998.
9. University of Chicago, Chicago. IL USA The globus project. http://www.globus.org
10. University of Virginia, Charlottesville. VA USA *Legion.* http://legion.virginia.edu
11. Pallas. Germany *Unicore.* http://www.unicore.org
12. R. Kosloff. Time-dependent quantum-mechanical methods for molecular dynamics. *J. Phys. Chem.*, 92(8):2087–2100, April 1988.
13. C. Leforestier, R. H. Bisseling, C. Cerjan, M. D. Feit, R. Freisner, A. Guldberg, A. Hammerich, G. Jolicard, W. Karrlein, H.-D. Meyer, N. Lipkin, O. Roncero, and R. Kosloff. A comparison of different propagation schemes for the time dependent Schrodinger equation. *J. Comput. Phys.*, 94(1):59–80, 1991.

14. M. D. Feit, J. A. Fleck, and A. Steiger. Solution of the Schroedinger equation by a spectral method. *J. Comput. Phys.*, 47:412, 1982.
15. M. D. Feit and J. A. Fleck. Solution of the Schrodinger equation by a spectral method ii. Vibrational energy levels of triatomic molecules. *J. Chem. Phys.*, 78:301–308, 1983.
16. M. D. Feit and J. A. Fleck. Wave packet dynamics and chaos in the Henon-Heiles system. *J. Chem. Phys.*, 80(6):2578–2586, 1984.
17. D. Neuhauser, M. Baer, R. S. Judson, and D. J. Kouri. A time-dependent wave packet approach to atom-diatom reactive collision probabilities: Theory and application to the H+H_2 (j=0) system. *Journal of Chemical Physics*, 93(1):312–22, July 1990.
18. D. Neuhauser, M. Baer, R. S. Judson, and D. J. Kouri. The application of time-dependent wavepacket methods to reactive scattering. *Computer Physics Communications*, 63(1-3):460–81, Feb. 1991.
19. U. Peskin, R. Kosloff, and N. Moiseyev. The solution of the time dependent Schrodinger equation by the (t,t') method: The use of global polynomial propagators for time dependent Hamiltonians. *Journal of Chemical Physics*, 100(12):8849–55, June 1994.
20. R. Kosloff. Time dependent approach to femtosecond laser chemistry. In: *Eighteenth International Conference on Physics of Electronic and Atomic Collision (XVIII ICPEAC)*, Aarhus, Denmark, 21-27 July 1993, pages 152–170.
21. G. Torres-Vega. Lanczos method for the numerical propagation of quantum densities in phase space with an application to the kicked harmonic oscillator. *J. Chem. Phys.*, 98(9):7040, 1993.
22. H. Tal-Ezer, R. Kosloff, and C. Cerjan. Low-order polynomial approximation of propagators for the time-dependent Schrodinger equation. *Journal of Computational Physics*, 100(1):179–87, May 1992.
23. R. Kosloff. Propagation methods for quantum molecular dynamics. *Annual Review of Physical Chemistry*, 45:145–152, 1994.
24. H. Tel-Ezer and R. Kosloff. An accurate and efficient scheme for propagating the time dependent schroedinger equation. *J. Chem. Phys.*, 81:3967–3971, 1984.
25. T. N. Truong, J. J. Tanner, P. Bala, J. A. McCammon, D. J. Kouri, B. Lesyng, and D. Hoffman. A comparative study of time dependent quantum mechanical wavepacket evolution methods. *J. Chem. Phys.*, 96:2077–2084, 1992.