INSTITUTE OF PHYSICS PUBLISHING EUROPEAN JOURNAL OF PHYSICS

Eur. J. Phys. **26** (2005) S31–S38 doi:10.1088/0143-0807/26/5/S04

Random number generators and the Metropolis algorithm: application to various problems in physics and mechanics as an introduction to computational physics

Florent Calvayrac

Laboratoire de Physique de l'Etat Condensé, UMR CNRS 6087 and FR 2575, Université du Maine, 72085 Le Mans Cedex 9, France

E-mail: Florent.Calvayrac@univ-lemans.fr

Received 23 May 2005
Published 4 July 2005
Online at stacks.iop.org/EJP/26/S31

Abstract

We present known and new applications of pseudo random numbers and of the Metropolis algorithm to phenomena of physical and mechanical interest, such as the search of simple clusters isomers with interactive visualization, or vehicle motion planning. The progression towards complicated problems was used with first-year graduate students who wrote most of the programs presented here. We argue that the use of pseudo random numbers in simulation and extrema research programs in teaching numerical methods in physics allows one to get quick programs and physically meaningful and demonstrative results without recurring to the advanced numerical analysis methods.

1. Introduction

Computer experiments are rapidly gaining ground as a pedagogical tool at all levels of science teaching. At university level, it is often desirable to have the students write a computer simulation program themselves as, on top of bridging the gap between theoretical courses and lab experiments, it teaches them the inner working and principles of simulation programs and motivates them to study computer programming and numerical analysis. However, in physics, at least in French universities, one is confronted, in practice, with the fact that many students have little or no experience of writing long computer programs, or have a very limited knowledge of numerical analysis. Both factors combine to hinder the writing of simulation programs based, for instance, on ordinary or partial differential equation system

solvers; such systems model the majority of physical problems of interest. Students who are not ready to follow several courses on those problems can then lose their interest in the field of computational physics. This problem can be worked around by the use of symbolic or numerical computational packages, but we want to demonstrate in this paper that the use of pseudo-random-number generators allows one to address a variety of situations by writing simple high-level programs, thus rapidly obtaining results and keeping the students motivated for numerically more demanding approaches.

We present here the material that was used during a 10 hour introduction to computational physics at graduate level at the Université du Maine during the 2003 winter semester, and the student projects which gave the best results.

2. Pseudo-random-number generators

We start with a brief reminder of the principles of pseudo-random-number generators, inspired by the discussions of [1, 2]. Giving the standard definition of such generators (namely, a deterministic series of numbers, the input of which to a computer simulation program gives results indiscernible from those obtained with real random numbers), we slightly digress to the subject of randomness in physics, discussing classically chaotic and in practice non-deterministic situations such as non-biased lottery drawings or dice throwing, as opposed to truly non-deterministic situations such as radioactive decay.

True random numbers are crucial not only in advanced statistical physics computations, especially in multiple dimensions, to avoid spurious correlations, but also in one-keypad encryption and in computer security, since the packets used in TCP/IP communications have a random starting number to avoid sequence prediction and the so-called identity spoofing in interrupted but still open communications. One can note that computer motherboard vendors nowadays include hardware random-number generators in their products for this purpose; these numbers, obtained from overamplified electron shot noise in the circuits, are obtained at quite a low rate for computational physics purposes, due to the quantization process, but can nevertheless provide good sequence starts for pseudo-random-number generators.

Surprisingly, as found by Knuth in [2] the best uniform pseudo random generators seem to be the simplest ones, such as a complicated permutation I of a set of successive integers from, say, 0 to 2^{N-1}, giving as many equally spaced reals from 0 to 1 by a simple division by the upper limit. It is then obvious that there are no 'holes' in the random numbers generated in this way, to the cost of a repeat of the sequence and thus a correlation with a 2^N period. It is therefore better to use self-programmed generators or certified ones than those found in programming environments; C `rand()` seems to have only $N = 16$, such as certain electronic slot-machines whose output loops after only a few days and which have been legally exploited by clever players. Standard Unix `drand48()` is much better from this point of view, with $N = 48$, as are the ones provided for instance in [1].

We have the student experiment with the basic recurrence generator

$$I_{n+1} = (aI_n + b)[2^N],$$

checking by hand, with $N = 3$ for instance, that for well-chosen a and b one has indeed a one-cycle permutation of the integers, and with $N = 32$ and arbitrary large a and b a set of seemingly random numbers. This fact can be checked with a simple two-dimensional (2D) plot of successive numbers: correlations or holes in the distribution are obvious in such representations.

3. The central limit theorem and non-uniform pseudo random numbers

We now use the uniform pseudo-random-number generators of the previous section to check the central limit theorem, stating for practical purposes in physics that an arithmetic mean of N uniform random numbers approaches a normal Gaussian distribution of $1/\sqrt{N}$ standard deviation. Numerically, this amounts to computing the histogram of N_s samples of the sum of N pseudo random numbers obtained as above.

We give the corresponding program (as the others discussed in this paper) at the following address: http://www.univ-lemans.fr/~fcalvay/ejp/.

This program of less than ten lines of a high-level language turns out to be surprisingly hard to write for many students, as they have never implemented histogram computations; computer language courses often only mention ordered array filling with simple loops or iterators. However, even students reluctant to use computers are enthusiastic about the very quick and demonstrative result they get, which can be easily linked to the corresponding statistical physics or gas kinetics courses, where the binomial distribution and its limit are used routinely in averaging microscopic properties. One can even link this result to the problems found either in industry for tolerance problems in mass production, or in the standard argumentation of basic error and incertitude theory in physics, where one supposes that there is no bias in the experiments and that the sources of errors in the measurements are independent, small and additive. In those conditions, the measurements tend to a normal distribution exactly as in the small program considered here. It is easy to visualize histograms obtained with a varying number of random deviates being summed, or with a varying number of samples N_s to fill the histogram. One then sees the transition from a noisy curve for a small N or a small N_s to a smooth Gaussian curve for large values of these parameters, and checks that the width of the curve indeed has a $1/\sqrt{N}$ dependence, which nicely justifies the error made in the Monte Carlo numerical integration, and hence shows the superiority of this method in terms of precision–cost product in higher dimensions compared to other integration methods.

This very simple technique to generate non-uniformly distributed random numbers is, however, obviously limited to normal distributions. Other techniques [1] exist to generate random numbers with an arbitrary distribution $P(x)$, such as finding the inverse of the integral from $-\infty$ to x of the wished distribution and applying it to uniform random numbers, which is obviously quite involved in practice or limited to certain cases where the primitive and reciprocate of $P(x)$ have an analytic expression: Poisson-like distributions spring to mind here.

We prefer von Neumann's rejection method, which consists in finding a majorant M to the desired distribution $P(x)$, then in the drawing of uniform random numbers (x, y) with x in a chosen interval and $y < M$. If $y < P(x), x$ is accepted and output from the method, else the method restarts. It is obvious if one draws a graph representing $P(x)$ and illustrates the method on this graph that the result is correct; however, if M is too large, the method will slow down considerably by rejecting too many couples, and one has to generate twice as many uniform random numbers than with the inversion of the primitive of $P(x)$ method.

The program used to compute histograms and check the central limit theorem can be easily modified by the students to check the rejection method, and one then gets a source of random numbers with arbitrary distributions[1]. Such numbers can be used as a source of

[1] Some students planning to pursue a career in teaching remarked that they could use the program to grade their future pupils with a distribution of grades matching perfectly official instructions!

input either for other simulation programs, or to check in advance experimental instruments or data processing programs with streams of numbers distributed as expected from the real-life situation.

4. The Metropolis algorithm and heuristical optimization

Having discussed non-uniform random number generation by the rejection method, we can now address the well-known Metropolis algorithm [4], which can be seen as an application of this method. One of the goals of this algorithm is to simulate the evolution of a physical system described by a set of generalized coordinates (q_i), at temperature T, the probability of the transition in between two states of this system with an energy difference of ΔE being given by a Boltzmann factor $P(\Delta E, T) = \exp(-\Delta E/(k_B T))$, with k_B the Boltzmann factor.

The Metropolis algorithm consists in generating a trial random move $(q_i' = q_i + \Delta q_i)$ corresponding to a $\Delta E = E'(q_i') - E(q_i)$ energy change, and accepting this move if a random number y uniformly drawn in between 0 and 1 is lower than $P(\Delta E, T)$. From our description of the rejection method, one sees that this algorithm is exactly similar and will generate configurations (q_i) of energy $E(q_i)$ with a Boltzmann probability distribution $P(E - E_g, T)$ relative to the state of lower energy E_g of the system. This is interesting *per se* as it allows one to explore the configuration space of a thermodynamical system with an alternative method to molecular dynamics, and also because, by a progressive annealing, it allows one of find heuristically the ground state of a system.

Minimum search by the Metropolis algorithm indeed has the advantage over other methods such as the steepest descent or the conjugate gradient, as, if one starts from a trial state close to a local minimum but far from the global minimum, one still has a chance of finding the global minimum by 'jumping' over the barriers separating the local minimum valleys, thanks to thermodynamical activation, whereas the other mentioned methods will only be able to find the local minimum. Of course, compared to an exhaustive enumeration of the configurations of the system, the method is only a heuristical one since it has many problem-dependent ingredients and may fail if the initial trial state, the temperature annealing law, or the size of the random trial moves (Δq_i), are badly chosen.

4.1. Phase transitions and isomers of a Lennard–Jones cluster: interactive visualization

Atomic and molecular clusters constitute a target of choice for the Metropolis algorithm, either to explore the thermodynamic properties thereof, or to find the lowest energy isomers. The method has been applied, for instance, to metallic clusters such as in [5], but also to model Lennard–Jones (LJ) clusters (see [6, 7] and related references). We give here a prototype implementation of the method: the energy E of the $N = 13$ atom cluster with atoms at positions $\mathbf{r}_i$ (this number of atoms being arguably the best known model system in the literature on atomic clusters) is given by

$$E = \Sigma_{i<j} V_{\mathrm{LJ}}(|\mathbf{r}_i - \mathbf{r}_j| \qquad \text{with} \quad V_{\mathrm{LJ}}(r) = 4(r^{-12} - r^{-6})$$

in the Lennard–Jones units for distance, energy and temperature.

Moves which would bring the atoms farther than a certain distance r_{cut} from the centre of the cluster, although physical, have to be rejected if one wants to study a given N atom cluster, else, for high temperatures, one observes a cooling phenomenon where surface atoms of the cluster evaporate to reduce the free energy. Interested readers can refer to the source code of the program that we give for downloading.

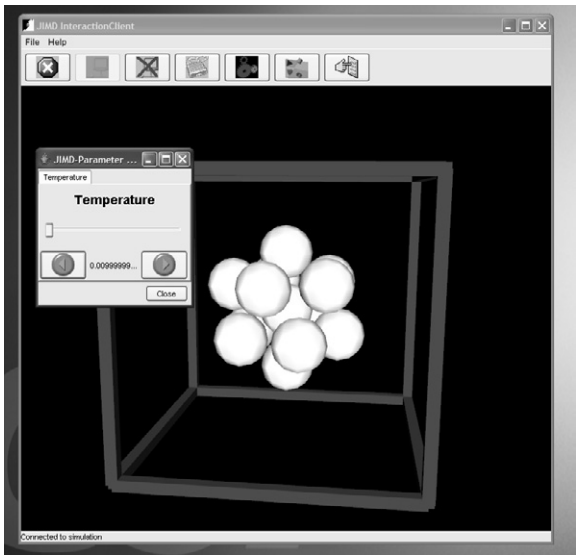

Figure 1. A screenshot of the JIMD visualization program interacting with the Metropolis simulation of a LJ13 cluster: the temperature can be interactively changed.

In this program, which was cowritten by graduate students S Maillet and M Buard, we also use an interactive remote visualization with a possibility of changing the views of the cluster as well as the temperature; for this we use the JIMD package [3], originally developed for molecular dynamics, which provides a simple environment for these purposes.

As can be seen in figure 1 which shows a screenshot of the JIMD window, the program finds an isomer for $T = 0$ with two horizontal rings of five atoms around a central three-atom vertical axis, and 'melts' at about $T = 0.3$, which can be seen by the fact that atoms have different nearest neighbours in time. For high temperatures at $T = 2.0$ for instance, one sees a gasification of the system. Those phase transitions can of course be studied with more rigour by computing quantitative ensemble quantities such as the calorific capacity, virial and so on; see the article by Calvo and Labastie about this subject in this issue of EJP. We feel that such a program is already quite demonstrative of the link that statistical and computational physics can make in between microscopic models (here, the Lennard–Jones potential) and macroscopic properties such as solid–liquid phase transitions.

4.2. Simple vehicle trajectory optimization

In this part, we present an application of extrema research with the Metropolis algorithm to vehicle motion planning which we devised with another group of graduate students (M Yan and D Novais). Various methods have been devised and applied to motion planning in the past [8], mainly to articulate robots, using essentially analytical methods. Here we present a simpler approach, applied to perhaps more amusing problems for motorsport-attracted students: trajectory optimization of an idealized car through a curve. This problem is similar to that treated in [9].

4.2.1. A car model. Since the development of a physics simulator of a realistic car is a huge task in itself with many input parameters and equations such as tyre behaviour, we model the physics of a car in a plane curve by

- the maximum speed $v_{\max}$ and the maximum longitudinal acceleration $a_{\max}$ corresponding roughly to gear ratios, the engine rotation speed, power and torque curves, the wheel radius, car mass, air and tyre resistance,
- the maximum longitudinal deceleration $b_{\max}$ corresponding to brake efficiency, car mass and tyre behaviour,
- the maximal lateral acceleration $s_{\max}$ corresponding to car mass and design, suspension and tyre behaviour,

and constrain the trajectory of the car to remain within those limits. This implies that we do not consider trajectories with skidding, nor we take into account the fine aspects of the car dynamics.

4.2.2. Trajectory discretization. The trajectory of the car is discretized by an ordered set of N couples (ax_i, ay_i) representing the successive Cartesian accelerations of the car's centre-of-mass as well as speeds in the lab's frame. We find this a more robust choice than optimizing the positions in time and differentiating like in [9].

Total trajectory time T_t, which is to be minimized by playing the role of a fictitious energy E in the Metropolis algorithm with a non-physical temperature Θ, ($k_B = 1$), is computed by the sum over all intervals of driving times Δt_i:

$$T_t = \sum_{i=1}^{N-1} \Delta t_i .$$

Car positions as a function of time (x_i, y_i) are obtained by a simple numerical integration from the initial speed, position and the set of discretized accelerations and time intervals: altogether we work on a configuration $(q_j) = (ax_i, ay_i, \Delta t_i, vx_1, vy_1, x_1, y_1)$ to be optimized by random changes (Δq_j).

4.2.3. Constraints and the Metropolis algorithm. As stated above, the longitudinal and lateral accelerations as well as the speed norm are constrained to stay between their maximum and minimum values.

Car positions (x_i, y_i), obtained by a simple numerical integration from the initial speed, position and the set of discretized accelerations are also constrained to stay on the track, defined, for instance, as the surface in between two 90° arcs of radii R_1 and R_2 connecting two semi-infinite straight tracks. Arbitrary track shapes could, of course, be described at the cost of more complicated geometrical definitions.

To enforce starting and ending conditions, the first and the last point of the trajectory at indices 1 and N have positions constrained to be at the beginning and end of the track. The starting position is, however, free to change on an axis perpendicular to the track.

Those constraints are enforced in the Metropolis algorithm by simply rejecting trial changes to the trajectory which violate them. This requires that the initial guess of the trajectory respect the constraints; in the case of the 90° arc, we simply consider another arc driven at low speed in the middle of the track.

We find that the best results are obtained with a fictitious starting temperature Θ_0 of 1.0 which is brought to zero as $\Theta = \Theta_0 k^{-0.3}$ where k is the iteration number in the Metropolis algorithm. Those annealing cycles are repeated every 5000 000 iterations. The convergence and good behaviour of the method can be seen by printing the ratio of rejected to accepted moves in the algorithm; too many rejected moves at the start imply a too low fictitious temperature, and this ratio should stabilize at 0.5 when the system is close to a minimum.

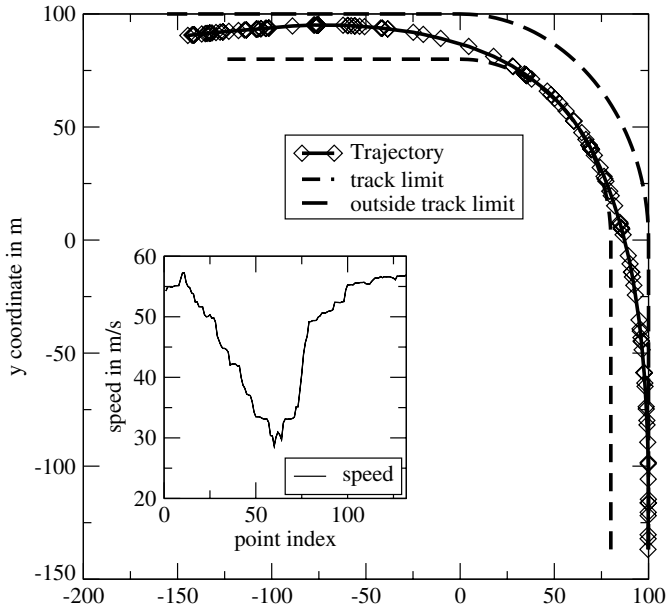

Figure 2. The trajectory and speed of a vehicle in a curve obtained after heuristical optimization with the Metropolis algorithm.

We start with steps of a size of $\Delta q_j^0 = 500$ units (this rather large value induces the initial trajectory to become disordered and far from any local minimum) which we modulate as a function of Θ by the following formula, which is known [6] in the case of LJ clusters to give the optimal random changes to the configuration and found to be usable in this case:

$$\Delta q_j = \Delta q_j^0 \left(\left(\frac{16}{1-\sqrt{\Theta}} \right)^{\frac{1}{6}} - \left(\frac{16}{1+\sqrt{\Theta}} \right)^{\frac{1}{6}} \right).$$

4.2.4. Results. The results presented in this section were obtained on a 90° curve with the inner radius 80 m and the outer radius 100 m, joining two straight tracks of length 136 m and 124 m and width 20 m. The trajectory was discretized with 132 points. The maximal speed of the car was fixed to 80 m s^{-1}, and the maximum longitudinal and lateral accelerations were fixed to 10 m s^{-2}, values that are typical of a sports car. 6×10^7 Monte Carlo steps were used, representing 120 annealing cycles and a run time of about 5 min on a contemporary PC.

We present typical results in figures 2, showing the optimal trajectory and speed in the curve. The optimal trajectory found is very close to those recommended in car piloting manuals [10], with the car starting at the outside of the track and aiming for the apex of curve, maximizing the radius of curvature of the trajectory and thus the speed for a given maximal lateral acceleration.

The speed starts at the maximum value compatible with the maximum possible speed after braking in the curve for the given maximal lateral acceleration, and increases after the curve where there is a slight braking; this would be avoided in a more realistic car model since it would generate a spin of the car. However, we feel that the main qualitative trends of an optimal trajectory and speed control are quickly found by the program. The results show

the heuristical nature of the optimization process, and the slight noise in the curves the errors brought by the discretization of the curve in finite segments.

5. Conclusion

In this paper, we have recalled the basic principles of random-number generation, the Metropolis algorithm and heuristical optimization, and presented several applications of these methods to physics and mechanics. This was done at a level which we found to be compatible with the level of average students on BSc or first year MSc courses in physics without a detailed knowledge of numerical analysis or advanced computer science and computer programming. Thanks to random number generators, we could quickly illustrate some points of statistical physics or devise some fun and unexpected applications such as vehicle trajectory planning, and thus spark the interest of some students in computational physics, without hammering at numerical techniques that are at first too advanced. Another route would be to use ready-made commercial or free numerical or symbolic packages, but we think that hands-on experience of a high-level language is a valuable tool for a professional career.

References

[1] Press W H *et al* 1992 *Numerical Recipes in Fortran 77: The Art of Scientific Computing* (New York: Cambridge University Press)
[2] Knuth D E 1981 *The Art of Computer Programming* vol 2 (Reading, MA: Addison-Wesley)
[3] Vormoor O 2001 *Comput. Sci. Eng.* **3** 98–104
See also http://www.user.gwdg.de/ovormoo/jimd/index.html
[4] Metropolis N *et al* 1953 *J. Chem. Phys.* **21** 1087–92
[5] Kohl C 1997 *PhD Thesis* F-A Universität Erlangen-Nürnberg
[6] Calvo F 1999 *Ann. Phys., Fr.* **24** 4
[7] Calvo F and Labastie P 2005 *Eur. J. Phys.* **26** S23
[8] Gupta K and del Pobil A P (ed) 1998 *Practical Motion Planning in Robotics: Current Approaches and Future Directions* (West Sussex: Wiley)
[9] Ferentinos K P, Arvanitis K G and Sigrimis N 2002 *J. Glob. Optim.* **23** 155–70
[10] Frère P 1959 *Je conduis mieux* (Verviers: Gerard)