NEWTON, RUNGE-KUTTA AND SCIENTIFIC SIMULATIONS

Zvonko Fazarinc

Palo Alto, California, USA

Key words: Scientific simulations, accelerated motion, numeric integration of Newton's equations, fourth order Runge-Kutta, Heun algorithm

Abstract: Scientific simulations of natural phenomena are powerful predictors of likely experimental outcomes. Improvement of their reliability and accuracy translates directly into time savings. Simulations are also powerful supporters of education. Created as teaching tools, simulations provide unique insights into mechanisms of addressed phenomena and can serve as experimental breadboards to the student.

Visual display of simulated behaviour is usually the last step in writing a simulation code. The relevant algorithms that convert forces into motion to be displayed are seldom given the attention of the expert scientist. His focus is understandably elsewhere and the simulation of motion has to draw on previous work or on libraries of integration algorithms.

This paper addresses the motion algorithm from the viewpoint of Newton's Laws and deals with the haphazard usage of pre-computer integration formulas such as the high order Runge-Kutta schemes. As these are offered as the'' high accuracy'' and conveniently packaged answer to all integration needs, these formulas have gained high acceptance without a demonstrated justification.

The goal of this paper is to highlight the mismatch with the computer age of the fourth order Runge-Kutta (FORK) integration formula and to analyze its performance in light of simpler formulas with more transparency and less expenditure of computer cycles.

I wish to dedicate this paper to the memory of my late friend and colleague professor dr.Lojze Trontelj with whom I had the pleasure to discuss the potential value of scientific simulations in the early days of computer evolution.

Newton, Runge-Kutta in simulacije v znanosti

Kjučne besede: simulacije v znanosti, pospešeno gibanje, numerična integracija Newtonovih enačb, metoda Runge-Kutta četrtega reda, Heunov algoritem

Izvleček: Znanstvene simulacije naravnih pojavov so pomembni napovedovalci verjetnih eksperimentalnih izidov. Povečanje njihove zanesljivosti in natančnosti ima za neposredno posledico velike prihranke časa. Simulacije so tudi močna podpora pri izobraževanju. Ustvarjene kot orodja za učenje, simulacije omogočajo edinstven vpogled v mehanizme obravnavanih fenomenov in študentom lahko služijo kot eksperimentalni poligon.

Vizualni prikaz simuliranega pojava je navadno zadnji korak pri pisanju simulacijskega računalniškega programa. Relevantnim algoritmom, ki sile prevedejo v prikazano gibanje, izkušeni strokovnjak le redko posveča posebno pozornost. Razumljivo je, da je osredotočen drugam, in simulacija gibanja se mora zato naslanjati na prejšnje raziskovalne rezultate ali na knjižnice integracijskih algoritmov.

Članek obravnava algoritem gibanja s stališča Newtonovih zakonov in nestrogo uporabo integracijskih formul, ki izvirajo iz predračunalniških časov, kot na primer postopke Runge-Kutta višjih redov. Ker jih ponujajo kot ''natančne'' in priročno prirejene odgovore na vse potrebe po integraciji, so te formule splošno sprejete brez dokazne upravičenosti.

Namen tega članka je osvetliti neskladje med računalniško dobo in Runge-Kutta (FORK) integracijsko formulo četrtega reda ter analiza njene uspešnosti v primerjavi s preprostejšimi, preglednejšimi formulami, ki zahtevajo manj računalniških ciklov.

Ta članek posvečam spominu na svojega preminulega prijatelja in kolego, profesorja dr. Lojzeta Trontlja, s katerim sem še v zgodnjih časih evolucije računalnikov razpravljal o potencialni vrednosti simulacij v znanosti

1 Introduction

Scientific simulations of natural phenomena can predict the outcomes of practical experiments if done correctly. For dangerous experiments their value is unprecedented, for expensive ones it is fiscally advantageous. Scientific simulation can also provide unique insights into the underlying mechanisms of phenomena studied. In this mode their educational potential is without precedent.

The task of a simulation designer is to assign relevant natural forces to objects that mimic their natural counterparts. The running simulation allows the objects to mutually interact and their resulting collective behaviour is studied. One of the common tools for tracking the evolution of a simulation is a dynamic visual display. This must contain some algorithm that converts the forces on objects into their velocity and position. This task is the focus of this paper.

The conversion of forces acting on objects into their positional changes would seem quite trivial to Isaac Newton and it may appear trivial to a practitioner of scientific simulations as well. A double integration of the forces acting on the mass in question is all that is necessary to obtain the object's instantaneous position. According to Newton's First Law of Motion /1/, an object's momentum M is preserved. It is defined for an object of mass m moving at velocity v as $M = mv$. The change of momentum dM/dt can be caused only by some force F acting on the object. Their relationship is given by Newton's Second Law of Motion as

$dM/dt = F$. Because we will ignore the change of mass in the continuation of this paper we will assign all changes of momentum to the velocity *v.* Consequently

$$dv(t) = F(t)dt/m \tag{1}$$

Furthermore, the temporal change of an object's position *ds/dt* is equal to its velocity *v*, thus

$$ds(t) = v(t)dt \tag{2}$$

The motion algorithm we employ in a simulation context must conform to equations (1) and (2) in the discrete domain of finite differences

$$\Delta v(t) = v(t+\Delta t) - v(t) = F\Delta t/m \tag{3}$$

and

$$\Delta s(t) = s(t+\Delta t) - s(t) = v\Delta t \tag{4}$$

where Δt is the smallest discrete time resolution of the simulation. Yet, the algorithms that are used to perform the two respective integrations do not necessarily produce the correct answers. In most cases they are chosen from a library of numerical integration routines with poorly defined behaviour, without relevance to the particular problem, sometimes producing inaccurate object positions and always consuming unnecessarily excessive computer resources. The positional accuracy of an object, which is subject to accelerating forces, can be critically important when the force is a function of object's position. Such is the case with gravitational and electromagnetic simulations as well as with all simulations of interacting objects.

In this paper we will use Newton's Laws of Motion /1/ as the reference for a critical analysis of the most frequently recommended integration algorithm known as the Fourth Order Runge-Kutta method /2/ in light of other integration algorithms.

2 Approach

The numerical double integration of forces may be addressed from a mathematician's viewpoint without regard to the physics of the problem. This approach had led to the majority of the numeric integration formulas in existence today and was driven by the quest for reduction of manual computation effort without compromising the accuracy of the approximation. We will elaborate on this in later sections.

The same integration question may also be addressed from the physicist's viewpoint and be driven by the demand for a match between Newton's Laws of Motion and the results produced by its discrete mimicry. This will be our approach in the search for the ideal algorithm. Let us first put on the mathematician's hat.

2.1 Discrete Integration Method from Mathematicians Viewpoint

All numeric integration formulas are based on the definition of the derivative's integral over a finite interval Δt

$$\int_{t}^{t+\Delta t} y'(t)dt = y(t+\Delta t) - y(t)$$

Various approximations of this integral lead to different families of numeric integration formulas. Newton's Interpolation formula, for example, leads to the Adams family while Taylor's expansion gives rise to the Runge-Kutta family. We will focus on the latter, which is based on

$$y(t+\Delta t) - y(t) = \Delta t\, y'(t) + \frac{\Delta t^2}{2} y''(t) + \\ + \frac{\Delta t^3}{3!} y'''(t) + \frac{\Delta t^4}{4!} y''''(t) + ... \tag{5}$$

The mathematician's task is now to decide where to truncate the infinite expansion, providing an acceptable error. A good guess is that the truncation error /3/ may be in the order of power of Δt of the first neglected term. Retaining then as many terms as reasonable and using as small a time step Δt as practical seems appropriate. Now the former would complicate the formula while the latter would require more repetitive evaluations to cover a desired time span. Neither of these seems to be an overriding consideration for today's computer performance. But we must consider the fact that a vast majority of existing numeric integration formulas were developed before and at the turn of the 20-th century, which this author likes to call the BC (before computers) era. The problems that needed to be addressed by numerical means were usually low order, nonlinear differential equations for which their changes, i.e. the first derivatives were known, arising usually from observations or measurements. When the humans were faced with a choice between numerous repetitive manual evaluations of a simpler formula versus fewer executions of a more complex formula, they would understandably choose the latter. And this is how the more complex integration formulas gained their BC fame.

2.1.1 Newton

We will first choose a driving force function $F(t) = A\cos(\omega t)$, which is representative of all force functions that can be decomposed into Fourier series. This excludes the Dirac function.

First we integrate (1) and (2) to obtain our reference data

$$v(t) = \frac{A}{m\omega}\sin(\omega t) \qquad s(t) = -\frac{A}{m\omega^2}\cos(\omega t) \\ v(0) = 0 \qquad\qquad s(0) = -\frac{A}{m\omega^2} \tag{6}$$

2.1.2 Fourth-Order Runge-Kutta (FORK)

Had we followed the mindset of the BC era we would have included at least the first three or four terms of the expansion (5) thus placing the anticipated truncation error into the fourth or fifth order of Δt, respectively. This would then allow us the choice of a larger time increment as is desir-

able for manual integration. In turn, this would lead to a fairly complex, yet very popular Fourth-Order Runge-Kutta (FORK) formula /2/. Its systematic derivation involves much algebra and the interested reader is directed to this reference. The FORK formula is

$$\begin{aligned} k_0 &= \Delta t\, y'\left[t,\, y(t)\right] \\ k_1 &= \Delta t\, y'\left[t+\Delta t/2\,,\, y(t)+\Delta t\, k_0/2\right] \\ k_2 &= \Delta t\, y'\left[t+\Delta t/2\,,\, y(t)+\Delta t\, k_1/2\right] \\ k_3 &= \Delta t\, y'\left[t+\Delta t,\, y(t)+\Delta t\, k_2\right] \\ y(t) &= y(t-\Delta t)+\frac{k_0+2k_1+2k_2+k_3}{6} \end{aligned} \qquad (7)$$

In (7) the slope is allowed to be the function of time and of the dependent variable *y*(*t*) should such be the case. We will use this formula to evaluate the motion quantities (3) and (4) and will then compute the deviation from Newton's answers in (6).

To evaluate (3) and (4) we must plug the respective values of *F* and *v* into (7). Because these are only functions of time the FORK parameters k_0 through k_3 adopt very simple forms. The result is shown below

$$\begin{aligned} v(t) &= v(t-\Delta t)+\frac{A\Delta t}{6m}\left[\cos\omega(t-\Delta t)+4\cos\omega(t-0.5\Delta t)+\cos\omega t\right] \\ s(t) &= s(t-\Delta t)+\frac{\Delta t}{6}\left[v(t-\Delta t)+4v(t-0.5\Delta t)+v(t)\right] \end{aligned} \qquad (8)$$

In Fig.1 we have superimposed the positions *s(t)* as calculated from (6) and from (8) as functions of the number of time increments Δt.

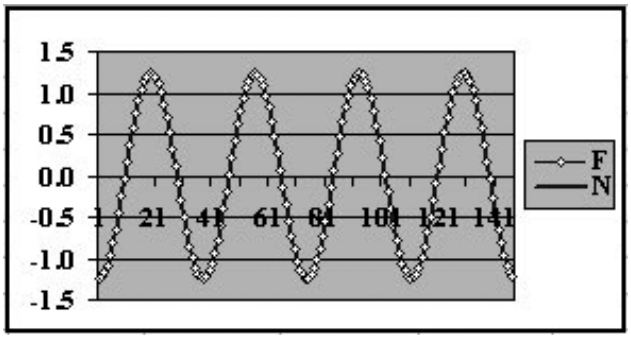

Fig. 1: Newton and FORK position for cosinusoidal driving force.

While deviations are not discernible in Fig.1 we have depicted in Fig.2 the percentage deviation between the FORK evaluated *s(t)* and that predicted by Newton. These are actual errors associated with our particular example and have nothing to do with the elaborate but often meaningless truncation errors /3/

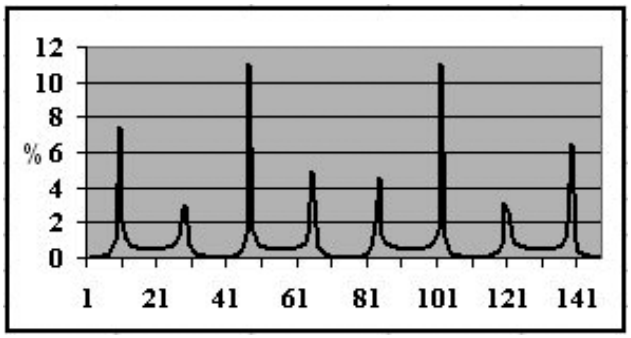

Fig. 2: Fractional error between Newton and FORK positions for cosinusoidal driving force

2.1.3 Second Order Runge-Kutta

Let us now choose to include only the first two terms of the Taylor expansion (5) With this choice we are committing, in the mathematician's mind, to a mere third order accuracy yet we do retain the control over Δt. The second term of expansion (5) calls for *y*''(*t*), which we do not know but can approximate by the central difference $\left[y'(t+\Delta t/2)-y'(t-\Delta t/2)\right]/\Delta t$, by the forward difference $\left[y'(t+\Delta t)-y'(t)\right]/\Delta t$ or by the backward difference $\left[y'(t)-y'(t-\Delta t)\right]/\Delta t$. We will return to this later but choose for now the forward difference and obtain instantly the following second order formula

$$y(t+\Delta t)=y(t)+\Delta t\,\frac{y'(t+\Delta t)+y'(t)}{2} \qquad (9)$$

Expression (9) is known as the trapezoidal but also as the Heun or Second Order Runge-Kutta formula /4/ in which we have replaced the fractional times with their linear approximations. Belonging to the second order brand this formula was discouraged by the BC mathematicians and remained in disrepute ever since. So let us now take a quantitative look at the error issue in light of what we have just learned about the FORK formula behaviour. To this end we adopt the same force function $F(t)=A\cos(\omega t)$ with its initial conditions spelled out in (6) to obtain from (3) and (4) the following equations

$$\begin{aligned} v(t+\Delta t) &= v(t)+\frac{A\Delta t}{2m}\left[\cos\omega\,(t+\Delta t)+\cos\omega t\right] \\ s(t+\Delta t) &= v(t)+\frac{\Delta t}{2}\left[v(t+\Delta t)+v(t)\right] \end{aligned} \qquad (10)$$

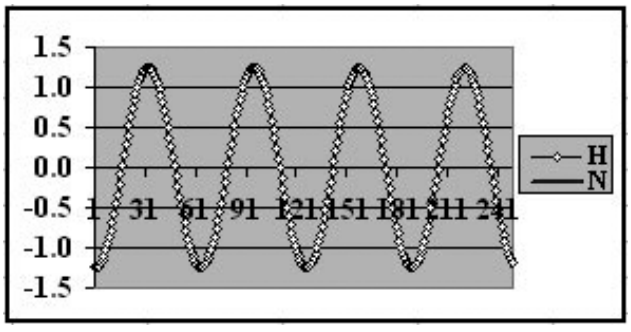

Fig. 3: Superposition of Newton and Heun position for cosinusoidal driving force.

Fig.3 displays the position *s*(*t*) as function of index $t/\Delta t$ and Fig.4 the percentage deviation of *s*(*t*) from the true position given by (6). This time we have experimentally adjusted the time increment Δt in such a way that the errors in Fig.4 and in Fig.2 are about the same.

As discernible from two sets of plots we had to compute almost twice as many points in (10) as we did in (8) to obtain the match of errors. Thus, the FORK formula gives us the same accuracy with fewer passes. No real surprise here but to compare the net computational effort we must establish some measurable criteria. Ignoring the memory accesses and all other overhead and not allowing any duplicate computations, we can count the number of floating point operations (FLOPs) needed for each time step.

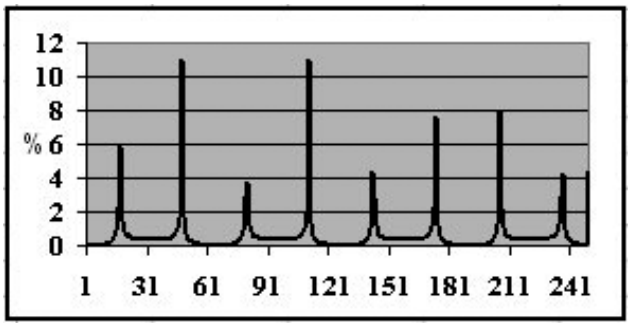

Fig. 4: Fractional error between Newton and Heun positions for cosinusoidal driving force

Formula (10) requires 30 FLOPs while (8) requires 49 provided that we have precomputed the repeating quantities such as $A\Delta t/2m$, $A\Delta t/6m$, $\Delta t/2$, etc. and allotted 10 FLOPs for each trigonometric function evaluation. One could then say that the two formulas are identical in terms of net computational effort for the same accuracy. But formula (10) gives us 70% more individual computed points. We will reexamine this issue after we have completed a more severe comparative test of the two formulas from a physicist's viewpoint.

2.2 Discrete Integration from Physicist's Viewpoint

While the truncation error /3/ might be the sole criterion to a mathematician when judging various integration formulas, to a physicist it is the trustworthiness of the simulated phenomenon that matters. We will therefore subject formulas (7) and (9) to a more realistic scrutiny because in simulations we seldom encounter a simple, clean textbook case of a well defined external force. The force is commonly self induced by the motion and is then fed back to the object involved. We will therefore make the force a function of the object's position and elect the following relationship.

$$\frac{F}{m}=\frac{\Delta v(t)}{\Delta t}=-\omega^2 s(t) \qquad (11)$$

where ω^2 is a proportionality constant for the moment. Such opposing functional dependence of force on position is commonly encountered in gravitational, electromagnetic, Van DerWall and other simulations of natural phenomena /5/, /6/.

2.2.1 Newton

First we solve Newton's equations (1) and (2) for the case of force function (11)

$$\frac{dv(t)}{dt}=-\omega^2 s(t) \qquad \frac{ds(t)}{dt}=v(t) \text{ or}$$

$$\frac{d^2 s(t)}{dt^2}=-\omega^2 s(t)$$

Solving for *s(t)* this second order differential equation yields the harmonic solution

$$\begin{aligned} s(t)&=s(0)\cos\omega t+\frac{v(0)}{\omega}\sin\omega t \\ v(t)&=v(0)\cos\omega t-s(0)\omega\sin\omega t \end{aligned} \qquad (12)$$

Position *s(t)* from equation (12) with *v*(0) = 0 is plotted in Fig.5 as the black solid line.

2.2.2 FORK formula

We introduce the force function (11) into equation (3) and then use the FORK formula (7) in both (3) and (4) to end up with an expression for position *s(t)*. Because the FORK formula integrates only one first order equation at a time, we would need a separate application of (7) to (3) and another to (4). Two sets of distinguishable factors "*k*" would have to be employed in general. But because (3) and (4) are coupled the following simplification is available from /7/

$$\begin{aligned} m_0&=\Delta t\, y'[t,\ y(t),\ y'(t)] \\ m_1&=\Delta t\, y'\left[t+\Delta t/2,\ y(t)+y'(t)\Delta t/2,\ y'(t)+m_0/2\right] \\ m_2&=\Delta t\, y'\left[t+\Delta t/2,\ y(t)+y'(t)\Delta t/2+m_0\Delta t/4,\ y'(t)+m_1/2\right] \\ m_3&=\Delta t\, y'\left[t+\Delta t,\ y(t)+\Delta t\, y'(t)+m_1\Delta t/2,\ y'(t)+m_2\right] \end{aligned}$$

$$\begin{aligned} y'(t+\Delta t)&=y'(t)+(m_0+2m_1+2m_2+m_3)/6 \\ y(t+\Delta t)&=y(t)+\Delta t\, y'(t)+\Delta t(m_0+m_1+m_2)/6 \end{aligned}$$

Because for our case the slope $y'(t)=v'(t)=-\omega^2 s(t)$ is only a function of time *t* the above equations simplify to

$$\begin{aligned} m_0&=-\Delta t\,\omega^2 s(t) \\ m_1&=m_2=-\Delta t\,\omega^2 s(t+\Delta t/2) \\ m_3&=-\Delta t\,\omega^2 s(t)+\Delta t \end{aligned} \qquad (13)$$

$$\begin{aligned} s(t+\Delta t)&=s(t)+\Delta t\, v(t)+\Delta t(m_0+m_1+m_2)/6 \\ v(t+\Delta t)&=v(t)+(m_0+2m_1+2m_2+m_3)/6 \end{aligned}$$

At this time we must point out that scientific simulations are commonly run with constant time increments Δt for a variety of practical reasons. One is the preservation of relative temporal occurrences of events under observation while simplicity of coding does not lag far behind. The half increment appearing in m_1 and m_2 must therefore be ap-

proximated by a linear interpolation

$$m_1 = m_2 = -\Delta t\omega^2 \frac{s(t) + s(t+\Delta t)}{2}$$

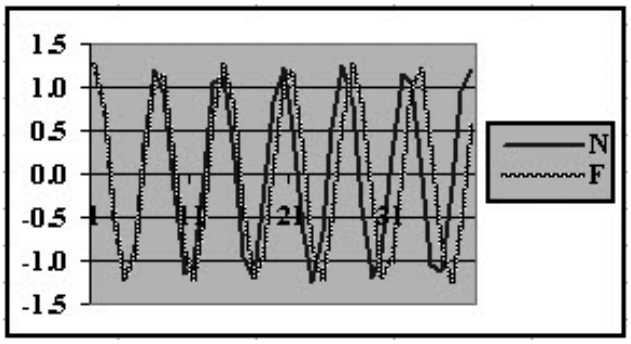

Fig. 5: Newton and FORK position when the driving force depends on position.

With this substitution the Runge-Kutta equation (13) produces the position *s(t),* shown in Fig. 5, superimposed on Newton's prediction from (12). The plot was computed for $\omega\Delta t = 1$, which is just over six points per period of the resulting harmonic motion *s(t)*.

The rugged waveform is the consequence of that. It was chosen intentionally for later comparisons with other options.

Fig. 6: Long term behaviour of Newton and FORK with positional feedback.

In order to gain an insight into the long term stability of the FORK solution we have plotted in Fig.6 the Newton prediction for 8500 time increments and superimposed on it the FORK answer from (13) for 8000 time increments. It is not doubtable that the FORK solution (13) would continue to provide stable answers beyond 8000 points but we will later prove this to be the case.

2.2.3 Second Order Formula

Let us continue to retain the physicist's viewpoint and argue with the mathematician who would want to convince us that only an infinite number of terms in the Taylor expansion would guarantee a perfect match of a numeric algorithm with Newton's formulas. On the other hand there are no doubts that Newton's equations (1) and (2) describe a second order system. Why should a second order numeric formula not suffice to adequately describe the accelerated motion? So, instead of choosing an existing integration algorithm we will force a second order integration formula to conform to Newton's laws of motion.

We start again from formulas (3) and (4). In them we have intentionally avoided the specification of respective temporal arguments of *F* and *v.* The discrete domain, characterized by finite time increments, often confronts us with the question of what happens before something else. We could, for example, compute the new velocity from Equation (3) by using the present force $F(t+\Delta t)$, the previous force *F(t),* their average as seen before or some other more general combination of the two. Similarly, we could compute the position s*(t)* from Equation (4) using some arbitrary combination of the previously evaluated and contemporaneous velocities. Without knowing the outcomes produced by a given choice, we have no reason to prefer one over the other. Therefore we will elect as yet undefined fractions of the force *a* $F(t+\Delta t)$ and (1-*a*) *F(t)* as the contributions to velocity changes as indicated below

$$v(t+\Delta t) = v(t) + \frac{\Delta t}{m}[aF(t+\Delta t) + (1-a)F(t)] \qquad (14)$$

Furthermore we will choose as yet unknown fractions of two primary velocity choices to determine their relative contributions to the change of position

$$s(t+\Delta t) = s(t) + [bv(t+\Delta t) + (1-b)v(t)]\Delta t \qquad (15)$$

Parameters *a* and *b* have values between zero and unity and we will try to extract them by a parameter optimization procedure designed to force a match between the simulated results produced by the above equations and the true accelerated motion expressed by Newton's Laws. We will do this for our specific choice of force defined by (11), which yields the following version of (14)

$$v(t+\Delta t) = v(t) - \omega^2\Delta t[as(t+\Delta t) + (1-a)s(t)] \qquad (16)$$

Expressions (15) and (16) represent the numeric integration algorithm that we are trying to force in compliance with (1) and (2).To this end we must find a closed form solution of the coupled equations (15) and (16). We have done this in the Appendix AP1 with the result

$$\begin{aligned} s(n\Delta t) &= s(0)B^{n/2}\cos n\Omega\Delta t + \\ &+ \frac{s(0)\omega\Delta t(b-a)/4 + v(0)/\omega}{\sqrt{1-\omega^2\Delta t^2(a-b)^2/4}} B^{n/2}\sin n\Omega\Delta t \\ B &= \frac{1+(1-b)(1-a)\omega^2\Delta t^2}{1+ab\omega^2\Delta t^2} \\ \Omega\Delta t &= \tan^{-1}\omega\Delta t\frac{\sqrt{1-\omega^2\Delta t^2(a-b)^2/4}}{1-(a+b-2ab)\omega^2\Delta t^2/2} \end{aligned} \qquad (17)$$

A comparison of position *s*(*t*) in (17) with that in (12) suggests that the quantity $B^{n/2}$ must be unity, calling for $1+(1-b)(1-a)\omega^2\Delta t^2 = 1+ab\omega^2\Delta t^2$, which further entails a + b = 1. The numerator of the second term of the $s(n\Delta t)$ expression in (17) demands b – a = 0 to conform to (12).

These two requirements imply the following overall conclusion of our analysis

$$a = b = 0.5 \tag{18}$$

Our final form of the set (15) and (16) is now highly reminiscnet of the Heun formula

$$\begin{aligned} v(t+\Delta t) &= v(t) - \omega^2 \Delta t \left[s(t+\Delta t) + s(t) \right] / 2 \\ s(t+\Delta t) &= s(t) + \Delta t [v(t+\Delta t) + v(t)]/2 \end{aligned} \tag{19}$$

For *a = b = 0.5* expression (17) takes on the form.which except for the angular frequency Ω matches Newton's prediction (12) in all other details.

$$\begin{aligned} s(n\Delta t) &= s(0)\cos n\Omega\Delta t + \frac{v(0)}{\omega}\sin n\Omega\Delta t \\ \Omega\Delta t &= \tan^{-1}\frac{\omega\Delta t}{\sqrt{1-\omega^2\Delta t^2/4}} \end{aligned} \tag{20}$$

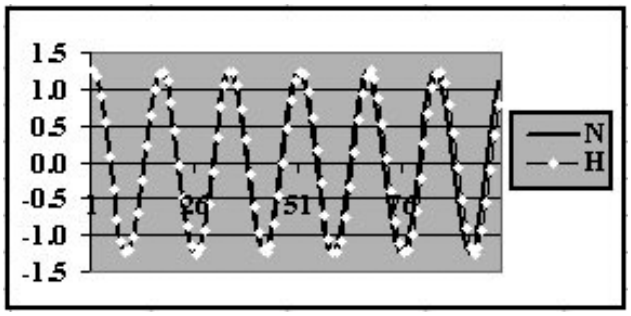

Fig. 7: Newton and Heun position when driving force depends on position

It should therefore not come as a surprise that an evaluation of (20) produces Figs.7 and 8 for which we have chosen the parameter $\omega\Delta t = 0.38$. Figs.5 and 6, on the other hand, were obtained with $\omega\Delta t = 1$ as stated earlier. The ratio of 1 to 0.38 happens to match the ratio 21 to 8 of floating point operations needed to execute (13), versus (19). This, in turn, assures equal computation time for both formulas with (19) providing 2.6 times as many time increments.

Fig. 8: Long term behaviour of Newton and Heun with positional feedback.

A debate over the relative accuracy of one or the other formula is irrelevant since both appear to match the Newton waveform except for the angular frequencies as seen from the graphs. For the Heun case defined in (20), the fractional deviation of Ω from ω is established as $\left(\tan^{-1}\left[\omega\Delta t / \sqrt{1-\omega^2\Delta t^2/4}\right] - 1\right)/\omega\Delta t$ and is plotted in Fig.9 versus $\omega\Delta t$. In the Appendix AP2 we have also found an analytic expression for the angular frequency Ω of the Runge-Kutta case and its relevant deviation from the nominal ω as $\left(\tan^{-1}\left[\omega\Delta t\sqrt{1-\omega^2\Delta t^2/12} / (1-\omega^2\Delta t^2/3)\right] - 1\right)/\omega\Delta t$. This is also plotted in Fig.9 as R..

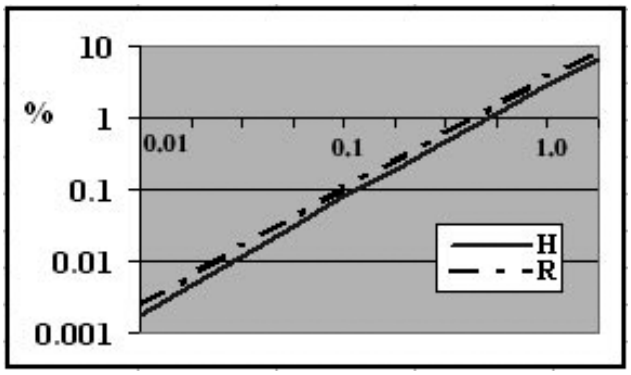

Fig. 9: Fractional error of FORK and Heun simulated frequency.

There is a slight difference in favor of Heun in terms of angular frequencies but more significant is the fact that for same expenditure of computer cycles Heun delivers more computed points with shorter time increments. Both of these are desirable for dynamic simulations, which demand displays of velocities and positions of objects.

3. Conclusion

In summary the Heun algorithm has been found to numerically integrate Newton's laws of motion with the same accuracy as the FORK formula in tests from the highest frequencies down to the Nyquist limit. The superiority of Heun formula over the FORK algorithm is its ability to deliver 260% more computed points at proportionately reduced time increment for same expenditure of computer cycles. Both of these are criteria important to simulation practitioners. Furthermore, its simplicity of implementation leaves the designer in control of the last stage of his dynamic simulation code. i.e. of computing the new velocities $v(t+\Delta t)$ and the new positions $s(t+\Delta t)$ of his objects from forces $F(t+\Delta t)$ currently acting on them. When we consider our starting force (11) we can present the preferred result of our analysis for general force functions as

$$\begin{aligned} v(t+\Delta t) &= v(t) + \frac{\Delta t}{2m}\left[F(t+\Delta t) + F(t)\right] \\ s(t+\Delta t) &= s(t) + \frac{v(t+\Delta t) + v(t)}{2}\Delta t \end{aligned} \tag{21}$$

A simulation program in which (21) was implemented has been tested with 100 spherical particles with randomly assigned initial velocities. They were allowed to interact through mutual gravitation, through three-dimensional collisions and individually by bouncing off the box walls. The gravitational masses of particles were adjusted so that their naturally formed clusters were able to disperse as their rotational velocity increased. This assured a continuous activity of all 100 particles. While there is no method available to evaluate the accuracy of observed simulated behaviour of this many objects, the physics comes to the rescue. Unless a loss mechanism or a source of energy is coded into the system the total energy must remain constant. Therefore we have monitored the overall energy through one million passes, during which each of the 100 particles received an update of position and velocity from the Heun algorithm from formula (21). During the whole observation time of one hour and 23 minutes that was needed for the experiment, the total energy of particles remained unchanged.

Appendix

AP1 Closed form solution of equations (15) and (16)

They are respectively

$$s(t+\Delta t) = s(t) + [bv(t+\Delta t) + (1-b)v(t)]\Delta t \quad \text{(APP1)}$$

$$v(t+\Delta t) = v(t) - \omega^2\left[as(t+\Delta t) + (1-a)s(t)\right]\Delta t \quad \text{(APP2)}$$

Take the first forward difference of (APP1)

$$s(t+2\Delta t) - s(t+\Delta t) = s(t+\Delta t) - s(t) + b\Delta t[v(t+2\Delta t) -$$

$$- v(t+\Delta t)] + (1-b)\Delta t[v(t+\Delta t) - v(t)]$$

Note that the first bracketed term arises directly from (APP2) as $-\omega^2\Delta t^2\,[as(t+2\Delta t) + (1-a)s(t+\Delta t)]$ and the second one as $-\omega^2\Delta t^2[as(t+\Delta t) + (1-a)\,s(t)]$

Substitute these into the above equation and collect contemporaneous terms to obtain

$$s(t+2\Delta t)(1+ab\omega^2\Delta t^2) - 2s(t+\Delta t)[1-(a+b-2ab)\omega^2\Delta t^2/2] +$$

$$+ s(t)[1+(1-a)(1-b)\omega^2\Delta t^2] = 0$$

This equation is equivalent to

$s(t+2\Delta t) - 2A\,s(t+\Delta t) + B\,s(t) = 0$ with

$$A = \frac{1-(a+b-2ab)\omega^2\Delta t^2/2}{1+ab\omega^2\Delta t^2}$$

$$B = \frac{1+(1-a)(1-b)\omega^2\Delta t^2}{1+ab\omega^2\Delta t^2} \quad \text{(APP3)}$$

(APP3) has two closed form solutions dependent on the relative magnitude of *A* and *B*. The case *B*>*A* is of interest to us. Application of the Z-transform /8/ results in the following Z-domain equation for s(*t*) in which the transformed variable is defined as $S(z) = Z[s(t)]$

$$S(z)z^2 - s(0)z^2 - s(\Delta t)z - 2AS(z)z + 2As(0)z + BS(z) = 0$$

From this follows immediately

$$S(z) = \frac{s(0)z^2 + s(\Delta t)z - 2As(0)z}{z^2 - 2Az + B} = \frac{s(0)z^2 + s(\Delta t)z - 2As(0)z}{(z-z_1)(z-z_2)}$$

where

$$z_{1,2} = A \pm j\sqrt{B-A^2} = |z|e^{\pm j\varphi} = \sqrt{B}e^{\pm j\tan^{-1}\frac{\sqrt{B-A^2}}{A}} \quad \text{(APP4)}$$

The inversion is done by the sum of residui since we are dealing with an analytic function /9/. We have discretized time as $t = n\Delta t$ in the following expressions

$$s(n\Delta t) = S(z)(z-z_1)z^{n-1}\big|_{z=z_1} + S(z)(z-z_2)z^{n-1}\big|_{z=z_2}$$

$$= \frac{s(0)z_1 + s(\Delta t) - 2s(0)A}{z_1 - z_2}z_1^n + \frac{s(0)z_2 + s(\Delta t) - 2s(0)A}{z_2 - z_1}z_2^n$$

Because $z_1 - z_2 = 2j\sqrt{B-A^2}$ we get

$$s(n\Delta t) = \frac{s(0)\left(A + j\sqrt{B-A^2}\right)}{2j\sqrt{B-A^2}}z_1^n - \frac{s(0)\left(A - j\sqrt{B-A^2}\right)}{2j\sqrt{B-A^2}}z_2^n +$$

$$+ \frac{s(\Delta t) - 2s(0)A}{2j\sqrt{B-A^2}}(z_1^n - z_2^n)$$

or

$$s(n\Delta t) = \frac{s(0)A}{2j\sqrt{B-A^2}}(z_1^n - z_2^n) + \frac{s(0)}{2}(z_1^n + z_2^n) +$$

$$+ \frac{s(\Delta t) - 2s(0)A}{2j\sqrt{B-A^2}}(z_1^n - z_2^n)$$

From (APP4) it follows

$$z_1^n - z_2^n = \sqrt{B^n}e^{jn\tan^{-1}\frac{\sqrt{B-A^2}}{A}} - \sqrt{B^n}e^{-jn\tan^{-1}\frac{\sqrt{B-A^2}}{A}} =$$

$$= 2j\sqrt{B^n}\sin n\tan^{-1}\frac{\sqrt{B-A^2}}{A}$$

$$z_1^n + z_2^n = \sqrt{B^n}\left[\varepsilon^{jn\tan^{-1}\frac{\sqrt{B-A^2}}{A}} + e^{-jn\tan^{-1}\frac{\sqrt{B-A^2}}{A}}\right] =$$

$$= 2\sqrt{B^n}\cos n\,\tan^{-1}\frac{\sqrt{B-A^2}}{A}$$

Then

$$s(n\Delta t) = s(0)\sqrt{B^n}\cos n\Omega\Delta t + \frac{s(\Delta t) - s(0)A}{\sqrt{B-A^2}}\sqrt{B^n}\sin n\,\Omega\Delta t \quad \text{(APP5)}$$

where $\Omega\Delta t = \tan^{-1}\frac{\sqrt{B-A^2}}{A}$

The denominator of the second right hand term of (APP5) can be found with straightforward but time consuming algebraic manipulations as

$$\sqrt{B-A^2} = \omega\Delta t\frac{\sqrt{1-\omega^2\Delta t^2(a-b)^2/4}}{1+ab\omega^2\Delta t^2} \quad \text{(APP6)}$$

$s(\Delta t)$ is extracted directly from (APP1) as

$$s(\Delta t) = s(0) + b\Delta t v(\Delta t) + (1-b)\Delta t v(0)$$

while $v(\Delta t) = v(0) - \omega^2 \Delta t a s(\Delta t) - \omega^2 \Delta t (1-a) s(0)$ is obtained from (APP2).

From expressions (APP5) and (APP6) and from the definition of *A* in (APP3) we obtain via tedious but elementary algebraic manipulation

$$\frac{s(\Delta t) - s(0)A}{\sqrt{B - A^2}} = s(0) \frac{\omega \Delta t (a-b)/2}{\sqrt{1 - \omega^2 \Delta t^2 (a-b)^2 / 4]}} +$$

$$+ v(0) \frac{1/\omega}{\sqrt{1 - \omega^2 \Delta t^2 (a-b)^2 / 4]}}$$

The argument of Ω is evaluated as follows

$$\frac{\sqrt{B - A^2}}{A} = \omega \Delta t \frac{\sqrt{1 - \omega^2 \Delta t^2 (a-b)^2 / 4}}{1 + ab\omega^2 \Delta t^2}$$

$$\frac{1 + ab\omega^2 \Delta t^2}{1 - (a + b - 2ab)\omega^2 \Delta t^2 / 2} = \omega \Delta t \frac{\sqrt{1 - \omega^2 \Delta t^2 (a-b)^2 / 4}}{1 - (a + b - 2ab)\omega^2 \Delta t^2 / 2}$$

With these values our solution for *s(t)* becomes

$$s(n\Delta t) = s(0) B^{n/2} \cos \Omega t +$$

$$+ \frac{s(0)\omega \Delta t (b-a)/2 + v(0)/\omega}{\sqrt{1 - \omega^2 \Delta t^2 (a-b)^2 / 4}} B^{n/2} \sin \Omega t$$

$$B = \frac{1 + (1-a)(1-b)\omega^2 \Delta t^2}{1 + ab\omega^2 \Delta t^2} \qquad \text{(APP7)}$$

$$\Omega \Delta t = \tan^{-1} \omega \Delta t \frac{\sqrt{1 - \omega^2 \Delta t^2 (a-b)^2 / 4}}{1 - (a + b - 2ab)\omega^2 \Delta t^2 / 2}$$

AP2. Closed Form solution of FORK equation (13)

We reproduce the relevant expressions

$$\begin{aligned} m_0 &= -\Delta t \omega^2 s(t) \\ m_1 &= m_2 = -\Delta t \omega^2 s(t + \Delta t)/2 \\ m_3 &= -\Delta t \omega^2 s(t) + \Delta t \end{aligned} \qquad \text{(APP8)}$$

and

$$s(t + \Delta t) = s(t) + \Delta t v(t) + \Delta t (m_0 + m_1 + m_2)/6 \qquad \text{(APP9)}$$

$$v(t + \Delta t) = v(t) + (m_0 + 2m_1 + 2m_2 + m_3)/6 \qquad \text{(APP10)}$$

When we substitute the *m*-factors from (APP8) into (APP9) we obtain

$$s(t + \Delta t) = s(t) + v(t)\Delta t - \frac{\omega^2 \Delta t^2}{6} [s(t) + 2s(t + \Delta t/2)]$$

In scientific simulations the stepping time increment Δt is commonly maintained at a fixed value and consequently the $s(t + \Delta t/2)$ quantity is simply not available. In the time period when the FORK algorithm was developed, the relevant slopes were extracted from measurements or observations so there was no problem to satisfy the above formulas. But in simulations we are forced to either run at halved increments and use only every second result, do some fancy iteration to extract the fractional time values or approximate the half time quantity. Only a linear interpolation is available in the second order systems so we will make the following substitution:

$s(t + \Delta t/2) = [s(t) + s(t + \Delta t)]/2$ This leads to

$s(t + \Delta t) = s(t) + v(t)\Delta t - \omega^2 \Delta t^2 [2s(t) + s(t + \Delta t)]/6$ A simple collection of terms yields

$$s(t + \Delta t) = s(t)(1 - \omega^2 \Delta t^2 / 3)/(1 + \omega^2 \Delta t^2 / 6) +$$

$$+ v(t)\Delta t / (1 + \omega^2 \Delta t^2 / 6) \qquad \text{(APP11)}$$

Take the first difference, which yields

$$s(t + 2\Delta t) - s(t + \Delta t) = [s(t + \Delta t) - s(t)]$$

$$\frac{1 - \omega^2 \Delta t^2 / 3}{1 + \omega^2 \Delta t^2 / 6} + \frac{[v(t + \Delta t) - v(t)]\Delta t}{1 + \omega^2 \Delta t^2 / 6} \qquad \text{(APP12)}$$

We need the velocity term in (APP12). To this end we insert the *m*-factors from (APP8) into velocity equation (APP10) to get

$$v(t + \Delta t) = v(t) - \frac{\Delta t \omega^2}{6} [s(t) + 4s(t + \Delta t/2) + s(t + \Delta t)],$$

After making the same approximation to the half increment term we obtain

$$v(t + \Delta t) - v(t) = -\frac{\omega^2 \Delta t}{6} [3s(t) + 3s(t + \Delta t)] \qquad \text{(APP13)}$$

Insertion of this into (APP12) yields

$$s(t + 2\Delta t) - s(t + \Delta t) \left[1 + \frac{1 - \omega^2 \Delta t^2 / 3}{1 + \omega^2 \Delta t^2 / 6} - \frac{\omega^2 \Delta t^2 / 2}{1 + \omega^2 \Delta t^2 / 6} \right] +$$

$$+ s(t) \left[\frac{1 - \omega^2 \Delta t^2 / 3}{1 + \omega^2 \Delta t^2 / 6} + \frac{\omega^2 \Delta t^2 / 2}{1 + \omega^2 \Delta t^2 / 6} \right] = 0$$

This equation can be easily rewritten into the form $s(t + 2\Delta t) - 2A s(t + \Delta t) + B s(t) = 0$ with

$$A = \frac{1 - \omega^2 \Delta t^2 / 3}{1 + \omega^2 \Delta t^2 / 6} \qquad B = 1 \qquad \text{(APP14)}$$

Taking advantage of previously recorded derivation between (APP3) and (APP5) we can write down the solution of our FORK equation defined by (APP8),APP(9) and (APP10)as

$$s(n\Delta t) = s(0)\sqrt{B^n} \cos n\Omega \Delta t +$$

$$+ \frac{s(\Delta t) - s(0)A}{\sqrt{B - A^2}} \sqrt{B^n} \sin n\Omega \Delta t$$

$$\Omega \Delta t = \tan^{-1} \frac{\sqrt{B - A^2}}{A}$$

The numerator of the second term is established from expression (APP14) as

$$B - A^2 = 1 - (1 - \omega^2 \Delta t^2 / 3)^2 / (1 + \omega^2 \Delta t^2 / 6)^2 =$$

$$= (1 - \omega^2 \Delta t^2 / 12) / (1 + \omega^2 \Delta t^2 / 6)^2$$

Just as easily we find

$$\sqrt{B - A^2} / A = \omega \Delta t \sqrt{1 - \omega^2 \Delta t^2 / 12} / (1 - \omega^2 \Delta t^2 / 3)$$

The target expression $s(n\Delta t)$ has an undefined numerator in the second term, part of which is obtained from (APP11) by setting the time equal to zero

$s(\Delta t) = s(0) \frac{1 - \omega^2 \Delta t^2 / 3}{1 + \omega^2 \Delta t^2 / 6} + v(0) \frac{\Delta t}{1 + \omega^2 \Delta t^2 / 6}$. Multiplier of the sinusoidal term is found as

$$\frac{s(\Delta t) - s(0) A}{\sqrt{B - A^2}} =$$

$$= \frac{s(0) \frac{1 - \omega^2 \Delta t^2 / 3}{1 + \omega^2 \Delta t^2 / 6} + \frac{v(0) \Delta t}{1 + \omega^2 \Delta t^2 / 6} - s(0) \frac{1 - \omega^2 \Delta t^2 / 3}{1 + \omega^2 \Delta t^2 / 6}}{\frac{\omega \Delta t \sqrt{1 - \omega^2 \Delta t^2 / 12}}{1 + \omega^2 \Delta t^2 / 6}} =$$

$$= \frac{v(0)}{\omega \sqrt{1 - \omega^2 \Delta t^2 / 12}}$$

The closed form solution of the FORK equation (13) is now

$$s(n\Delta t) = s(0) \cos \Omega \Delta t + \frac{v(0)}{\omega \sqrt{1 - \omega^2 \Delta t^2 / 12}} \sin \Omega \Delta t$$

$$\Omega \Delta t = \tan^{-1} \omega \Delta t \frac{\sqrt{1 - \omega^2 \Delta t^2 / 12}}{1 - \omega^2 \Delta t^2 / 3}$$

The fractional angular frequency deviation is

$$\frac{\Omega \Delta t - \omega \Delta t}{\omega \Delta t} = \frac{\tan^{-1} \omega \Delta t \sqrt{1 - \omega^2 \Delta t^2 / 12} / (1 - \omega^2 \Delta t^2 / 3)}{\omega \Delta t} - 1$$

References

/1/ J.S Newton "Philosophiae Naturalis Principia Mathematica", Translation by Andrew Motte, UC CAL Press, Berkeley, CA, 1946 p. 13

/2/ F.B.Hildebrand "Introduction to Numerical Analysis", Second Edition, Dover Publications, New York 1974, p.285-292

/3/ Ibid pp.5-10

/4/ Ibid p.290

/5/ Z.Fazarinc, "Getting Physics into the Bounce", IEEE Potentials Vol.14 No.1, Feb./March 1995, pp.21-25

/6/ Z. Fazarinc, "Potential Theory, Maxwell's Equations, Relativity, Radiation and Computers", CAEE, Vol.7, No.2, John Wiley&Sons, Inc 1999 pp.51-86

/7/ F.B.Hildebrand "Introduction to Numerical Analysis", Second Edition, Dover Publications, New York 1974, pp.291-292

/8/ John A. Aseltine, "Transform Method in Linear Systems Analysis", McGraw-Hill Book Co., Inc. 1958, p.260

/9/ Ibid, p.276.

Dr. Zvonko Fazarinc
Formerly director of R&D laboratory at Hewlett Packard Co in Palo Alto
and Consulting professor of EE at Stanford University, California
620 Sand Hill Road, 417D, Palo Alto, California 94304
Tel.: (001) (650) 330 0310; Fax: (001) (650) 330 1967
E-mail: z.fazarinc@comcast.net

Prispelo (Arrived): 26.06.2008 *Sprejeto (Accepted): 15.09.2008*