

XANALOG.COM Presents:

How much simulation speed do you need? The answer might surprise you.



Imagine that you are implementing a proportional-integral (PI) controller with a real time digital processor. The following is known:

1. The analog input is converted for digital processing by an 8-bit A/D converter.
2. The integrator of the PI is computed with the Euler integration rule.
3. The controller's analog output is generated by an 8-bit D/A converter.
4. The highest frequency that the PI controller will have to process is 10 Hz.

At what frequency will the controller need to operate ? Choose: A or B

- A. It needs to update at more than 2 times the highest frequency of the A/Ds input or greater than 20 Hz.
- B. It needs to update at more than 21 times the highest frequency that the integrator needs to integrate or greater than 210 Hz.

Whether one updates at 2 times a given frequency or at 21 times a given frequency can make a major difference in the investment in real-time hardware and software needed to solve a given problem. This paper will discuss the issues surrounding which update rate is needed in a given situation, with the goal of providing the user with information they need to make the most economical software and hardware choice.

INTRODUCTION

The period of transition from analog to digital controllers and from and from analog computers to digital real-time hardware-in-the-loop simulation systems was decades of struggling for sufficient computation power. Today engineers put a 333 MEGAFLOP computer (twice the power of a CRAY I supercomputer and six times the power of the Texas Instruments C-40 DSP chips) on their desk when they install a 333 MHz Pentium®. There are still problems that demand this level of power and more, ***but there is a vast range of controller prototyping and hardware-in-the-loop simulation problems that can be readily and economically solved on garden-variety PCs.*** By taking advantage of block-diagram programming software such as SIMULINK interfaced to real-time I/O modules with a product such as XANALOG Corporation's REALoop, users are quickly and easily carrying out real-time testing.

However, there remains a feeling among users and among vendors (probably originating with the people that still bear the scars of decades of struggling for sufficient arithmetic power) that real-time simulation is a huge problem and that solutions can only come from super-scale RISC or paralleled DSP chips running highly-optimized C code. To avoid falling into this trap, the user needs to understand the level of performance that is needed for their problem.

Consider the many simulation and control problems encountered in the automotive field. Real-time simulation and rapid controller prototyping have proven to be extremely valuable through the worldwide automotive industry. The speed of automotive dynamics varies over several orders of magnitude. The following lists testing being successfully carried out by XANALOG users and the update rates they are using:

- Power Steering Controller Prototyping - 4 times/sec
- Diesel Engine Dynamics Simulation - 200 times/sec
- Vehicle and Tire/Road Interaction Simulation - 2000 times/sec

Clearly the level of performance and the investment that must be made to achieve it can vary over a large range. XANALOG has written this paper to help the real-time-controller-prototyping and hardware-in-the-loop-simulation user with sizing their problem and arriving at an adequate and economical solution to their own real-time testing needs.

The correct answer to the quiz is B. The reasons will be discussed in the following text.

SIMULATING & CONTROLLING DYNAMIC SYSTEMS

Dynamics means differentiation and integration—The dynamics of interest in simulation and control are often those that can be mathematically modeled with integrators and differentiators. That is these dynamics can be modeled with differential/integral equations.

Controllers such as the widely used proportional-integral-differential (PID) or proportional-integral (PI) controllers both use an integration operation. More complex controllers (compensators) such as those represented by ratios of polynomials in S can also be implemented with integrators. One common way of converting continuous controllers ("S-domain") to digital controllers ("Z-domain") is to rearrange the S polynomials into (1/S) factors and then inserting a digital integration rule such as Euler or Tustin for the (1/S) factors (Franklin 1980).

Plant dynamics simulations are usually implemented using integrators to represent the dynamics. This is done simply because the integration operation is more stable and integration tends to reduce noise. Differentiation, on the other hand, tends to enhance noise and to increase the possibility of instability. Usually the differential equations modeling a given plant's dynamics can be converted to equivalent integral equations.

Integration is key—An essential issue in implementing both digital controllers and digital simulations of plant dynamics is how to integrate in a digital environment (Gelb 1974). The problem of integrating data taken at discrete intervals is centuries old. Scientists have long needed to deal with experimental data recorded at discrete intervals. For example, consider Galileo's telescopic measurements of planetary positions made on successive nights. Newton invented calculus to deal with the calculation of planetary body dynamics in the 17th century.

There is a considerable body of literature concerned with the numerical integration of differential equations. Much of this work was carried out around 1900 (Fröberg 1985). It was then that Runge, Heun, and Kutta published numerical integration techniques, such as the Runge-Kutta method, which are in wide use today. In those days they calculated by hand or with a

mechanical adding machine. Therefore they were very interested in finding ways of getting stable calculations while computing as infrequently as possible along the traces that they were integrating. The frequency analysis of integration rules that forms the basis of much of what follows was introduced by Hamming (Hamming 1962).

THE DIFFERENTIAL RATES OF REAL-TIME DIGITAL SYSTEMS AND THE RULES THAT GOVERN EACH

Consider the block diagram of Figure 1 which represents a typical real-time digital system. Even the simplest of this class of system has several rates associated with it. First there is the rate at which the A/D converter must run to acquire the data without introducing aliasing errors. Then there is the rate at which the simulation or the controller must run to achieve a stable simulation of the plant dynamics or the control law. Finally there is the rate at which the D/As must run to generate an acceptably smooth output signal. Much more complex situations are common, such as when one part of a controller or plant simulation operates in a frequency range that is quite different from that of another part.

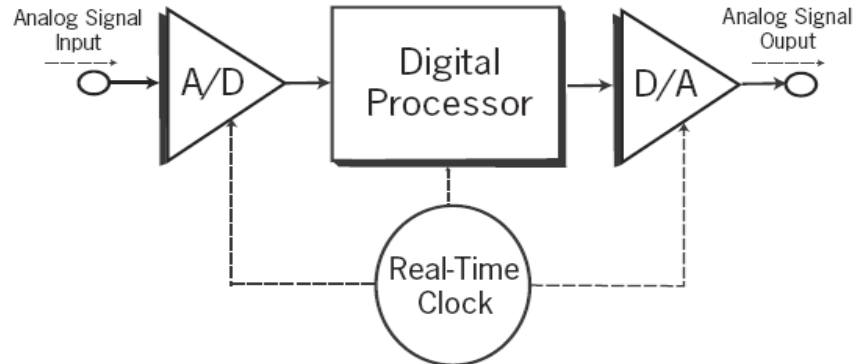


Figure 1. A block diagram of a typical real-time system.

For example, there may be multiple control loops such as a position loop and a velocity loop that are running at different rates. A plant simulation might have widely differing rates. For example, consider the simulation of a high speed turbine pump with natural frequencies in the range of 100 Hz that is pumping a liquid whose temperature changes with a time constant of hours that must also be modeled.

The rule for sampled data—Now returning to Figure 1, the A/D converter and the D/A converter are concerned with sampled data. Sampling of data is governed by the Nyquist theorem that calls for samples to be acquired (or generated) at a frequency that is at least twice that of the highest frequency to be sampled or generated. It is from this theorem that the "A" answer to the opening quiz is suggested. Namely, if 10 Hz is the highest frequency that will reach the A/D converter, then it must digitize with at least 20 Hz. But what about the real-time computation and, in particular, what about the integration operations that it contains?

A rule for integration—The rules governing an integrator are much more severe than those governing sampling of analog signals. If frequencies up to 10 Hz are to be integrated by a *digital integrator implemented with the Euler integration rule*, the integrator might need to update at a rate of 16 times the highest frequency. This is based on the requirement that the amplitude of the integrated sinusoids emerging from the integration operation be within 0.06 db of their correct amplitude. Thus in the case of a 10 Hz highest frequency that needs to be integrated, the integrator must update at a 160 Hz rate. More complex integration rules such as the widely-used *4th Order Runge Kutta (RK-4) rule* can achieve a 0.02 db amplitude error at 4 times the highest frequency, or at 40 Hz in this case.

The Ratio of the frequency of integration to the highest frequency to be integrated.	The Euler Rule's Deviation from an Ideal Integrator.	The 4th-Order RK-4 Rule's Deviation from an Ideal Integrator.
64:1	0.004 db	0.0000003 db
32:1	0.01 db	0.000004 db
16:1	0.06 db	0.00007 db
8:1	0.2 db	0.001 db
4:1	0.9 db	0.02 db
2:1	3.9 db	0.4 db

Table 1. Euler and 4th Order Runge-Kutta Integration Rule's deviation from an ideal integrator as a function of the ratio of the update frequency of the integration to the highest frequency being integrated.

The procedure used to compute Table 1 is explained in the paper in Appendix A. There the frequency response of an Euler integration is computed. The same computation was carried out on Fourth- Order Runge-Kutta.

Now let's apply this to the popular PI controller. A block diagram model of a typical PI controller is shown in Figure 2. Let's suppose that an 8-bit A/D and an 8-bit D/A are connected to the controller. Assume that 8-bit resolution, one-part-in-256, is adequate for the plant that is to be controlled by the PI controller.

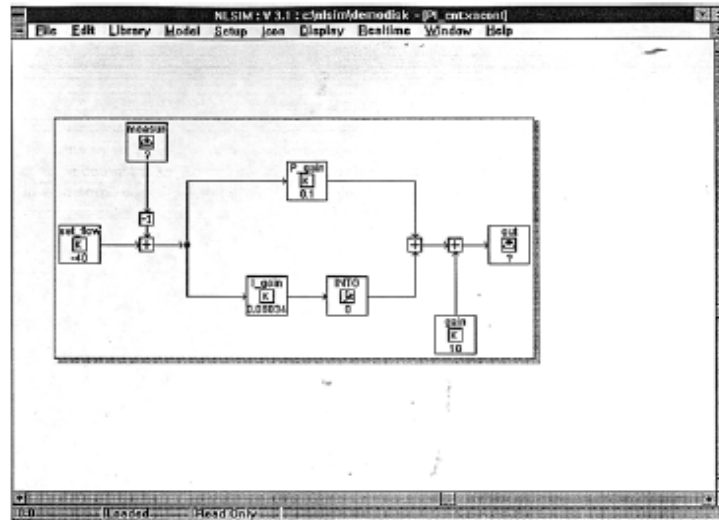


Figure 2. A NL-SIM block diagram of a PI controller.

If we look at Table 2 below, we see that the integrator needs to be at 3.5 times the highest frequency present for RK-4 integration. Let's suppose that the plant is a hydraulic power steering mechanism and the highest frequency in the mechanism's response is about 5 Hz. If we use RK-4 integration, Table 2 tells us that we will need to update the integrator at 17.5 Hz (3.5 x 5 Hz) to get an accuracy of one-part-in 256. However, if we wish to use Euler, then an update rate of 105 Hz (21 x 5 Hz) will be needed. Given the huge difference between RK-4 and Euler, why would one ever want to use Euler?

Number of Bits	Resolution 1 part in	To Integrate with Equivalent Accuracy	
		with RK-4 $F_{update}/F_{highest}$	with Euler $F_{update}/F_{highest}$
6	64	2.6	10
8	256	3.5	21
10	1024	4.9	41
12	4096	6.9	83
14	16384	9.7	170
16	65536	14.0	360

Table 2. The Update-Rates needed to achieve equivalent levels of accuracy for Euler and for 4th-Order Runge-Kutta Integration Techniques.

WHY NOT ALWAYS USE 4th ORDER RUNGE KUTTA?

Given that RK-4 performs so much better, why ever bother with Euler? The answer is that RK-4 requires a lot more computation: Four evaluations of the simulation (or controller) and also some complex formulae for combining the results of the various evaluations. Euler requires only one evaluation of the equations and a simple formula for combining the result (Fröberg 1985). The bottom line is that RK-4 requires on the order of *three times* as much computation as Euler. Also, by using RK-4, *it may be impossible to complete the necessary computations within the real-time frametime*. The clock beats at a regular interval and all computation and data I/O *must* be completed between beats of the clock.

Although Euler works in many situations, there are systems for which Euler integration will be unable to compute the dynamics. A simple example is undamped oscillations. Try simulating a second order system with zero damping using Euler and one finds that even infinitesimal time steps will yield a surprisingly unstable simulation, while RK-4 will have no difficulty at reasonable time steps. Another area in which Euler falls down is phase response.

A Few words on the phase shift of integrators—An ideal integrator has a -90° phase shift over the band of frequency of interest. Forth Order Runge-Kutta 4th does this beautifully. Euler on the other hand starts with a -90° shift at DC, but moves away from -90° in a linear phase fashion. Thus if phase shift is an issue, Euler may be unsuitable.

A Real Case—Several years ago XANALOG user was emulating in software a hardwired digital controller. The hardwired controller ran at a 2000 Hz update frequency and thus the emulation needed to run at 2000 Hz. The controller was complex but with Euler integration it was possible to compute the block diagram of the controller in 400 μ s on the 133 MHz Pentium® available for the job. Running the dozen I/O channels required additional time that brought the total time close to 500 μ s, meaning 2000 Hz real-time operation was possible. Unfortunately, subsequent testing at 2000 Hz showed that stable and accurate operation was *impossible* with Euler integration. Testing the individual submodels that made up the block diagram model showed that the instabilities were due to a notch centered at 400 Hz. With that removed, everything else ran fine with Euler at 2000 Hz. A look at Table 2 shows that this makes sense.

Note that if we ask for the integrations to be accurate to one part in 1000 we would have the following situation:

$$\begin{aligned} F_{\text{update}} &= (\text{Integration rule constant from Table 2}) \times F_{\text{highest}} \\ &\quad \text{Assume } F_{\text{highest}} = 400 \text{ Hz, Then:} \\ \text{for Euler: } F_{\text{update}} &= 41 \times 400 \text{ Hz} = 16,400 \text{ Hz} \\ \text{for RK-4: } F_{\text{update}} &= 4.9 \times 400 \text{ Hz} = 1960 \text{ Hz} \end{aligned}$$

Thus according to the results of the integration rule frequency response analysis presented in Table 2, RK-4 should be able to compute the 400 Hz notch filter running at 2000 Hz. This turned out to be the case for our user. The user got the emulation working on the 133 MHz Pentium by running all of the integrators in the model at 2000 Hz, but the integrators in the notch filter submodel were implemented with RK-4 and the rest of the model's integrators were implemented with Euler.

Conclusion

The frequency analysis of Euler and 4th Order Runge-Kutta (RK-4) integration methods yielded guidelines for choosing the update frequency needed for the operation of digital controllers and hardware-in-the-loop plant simulations. For example, if the simulation must integrate frequencies of F_{highest} with an accuracy of one part in 1000, then Table 2 reveals that the update frequency of the system must be: $4.9 \times F_{\text{highest}}$ for the 4th Order Runge-Kutta integration rule and $41 \times F_{\text{highest}}$ for the Euler

rule. This means that XANALOG's REALoop product operating at its maximum update frequency of 1000 Hz can integrate frequencies of up to 200 Hz with a one part in 1000 accuracies using RK-4. This is adequate for either controlling or simulating the dynamics of a wide variety of plants that one encounters today, such as the powersteering controller or the diesel engine dynamics discussed in the Introduction.

References

1. Hamming, R., "Numerical Methods for Scientists and Engineers", McGraw-Hill, New York, NY, 1962
2. Gelb, A., "Applied Optimal Estimation", The MIT Press, Cambridge, MA, 1974, pp. 294-303
3. Franklin, G., "Digital Control of Dynamic Systems", Addison-Wesley, Reading, MA, 1980, pp.54-59
4. Fröberg, C., "Numerical Mathematics", Addison-Wesley, Redwood City, CA, 1985, pp. 281-327
5. Schrage, M.H., "A New Criteria for the Update Speed Needed for the Correct Real-Time Simulation of the Continuous Dynamics of Automotive Systems", Proceedings Advanced Vehicle Control Conference, Yokohama, Japan, 1994, pp. 433-434

Specifications subject to change without notice.

All trademarks are the property of their respective owners

© XANALOG Corporation, 1998, 2006 WHITE98,WPD

APPENDIX A

DERIVATION FOR SAMPLING FREQUENCY

The computation of the frequency response of an Euler integrator is presented here in detail. This will be helpful for those who would like to carry out this analysis on other integration rules.

The Euler approximation to the area, $Y(N)$, under the sampled curve $x(t)$ is given by:

$$Y(N) = \sum_{n=1}^N T_s x(n)$$

where T_s is the time between the samples taken of $x(t)$ and N is the number of samples.

$$Y(N+1) = T_s x(N+1) + \sum_{n=1}^N T_s x(n)$$

$$Y(N+1) = T_s x(N+1) + Y(N)$$

or letting $n=N+1$:

$$Y(n) = T_s x(n) + Y(n-1)$$

Now transforming to the Z domain:

$$Y(z) = T_s X(z) + z^{-1} Y(z) \quad \text{or}$$

$$\frac{Y(z)}{X(z)} = H_E(z) = \frac{T_s}{1 - z^{-1}}$$

Now let $z = e^{j\alpha}$ where $\alpha = 2\pi(\omega/\omega_s)$ and $\omega_s = 2\pi/T_s$:

$$H_E(e^{j\alpha}) = \frac{2\pi}{\omega_s (1 - e^{-j\alpha})}$$

using $1 - e^{j\alpha} = e^{j\alpha/2} (e^{-j\alpha/2} - e^{j\alpha/2})$
and then $-2j \sin(\theta) = e^{-j\theta} - e^{j\theta}$

$$H_E(e^{j\alpha}) = \frac{2\pi}{-(2j\omega_s) e^{-j\alpha/2} \text{SIN}\left(\frac{\alpha}{2}\right)}$$

note for small θ : $\sin\theta \approx \theta$ thus

$$H_E(\omega) = \frac{1}{|\omega|}$$