Time handling and its problems

Leif Gustafsson © File: LAB-5_Time_Handling.docx 2020-04-23

Contents: Time handling is the method to update the model over time. In CSS languages such as e.g. StochSD the ***Time Slicing*** method is used. These exercises deal with practical problems that might arise because of this.

The purposes of the exercise are:

- *To understand the problems that might appear because continuous-time development is handled by updating the model in discrete time steps.*
- *To understand how to handle such problems.*

"A discrete description of time is never perfect. "

Name:	**Date:**
Course:	**Approved:**

I. TIME HANDLING FOR A DETERMINISTIC MODEL

1. Introduction

Time handling is about methods to update a simulation model over time. In a CSS language (e.g. StochSD) the ***Time Slicing*** method is used. Time slicing means that the model 'equations' are updated, where after Time is incremented by a small time-step (slice), DT. The procedure is repeated until the final point in time to be studied is reached. This is analogous to *integrating* a differential equation over time, why the word *'integration*' often is used also for the numerical methods that update the difference equations over time.

Numerical updating of the model

The dynamic equation

In Continuous System Simulation (CSS) the dynamic development is generated by Flows filling and draining a Stock. It is the Stock equation (called the *state equation*) that updates the dynamics by relating its increase or decrease to its in and out flows, according to:

Stock(t+DT) := Stock(DT) + DT·InFlow(during DT) - DT·OutFlow(during DT)

However, for a deterministic model, we can simplify this to:

Stock(t+DT) := Stock(DT) + DT·NetFlow(during DT), see Figure 1.

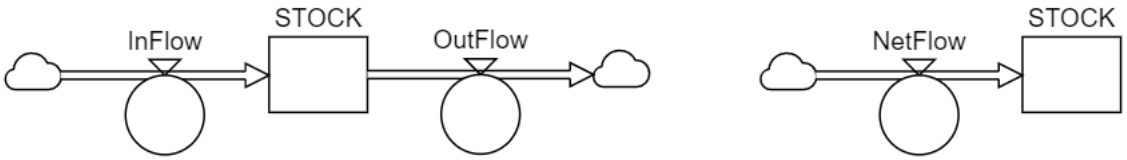


Figure 1. The basic dynamic relation between Stock and Flows. To the right, a simplification where NetFlow = InFlow - OutFlow. When the NetFlow is positive – the level in the 'bathtub' (Stock) will rise, and when it is negative it will sink.

Although there can be many Stocks and Flows in a CSS model, the model can always be reduced to a system of Stock-NetFlow equations, why we will illustrate the important principles with a single Stock-NetFlow equation. (In a CSS model we will also have a number of *algebraic* equations, i.e. equations that only contain + - * and /, but no dynamic complications. Such equations are handled by Auxiliaries and can be left out here.)

Mathematically, the Stock-Netflow equation is formulated as:

dx/dt =f(x, t); where x stands for Stock and f for NetFlow. dx/dt stands for the derivative (change) of x over time, so f(x, t) only points out that the NetFlow, f, in its turn depends on the value of the stock x and of time. (In fact, f can be affected by several Stocks, which mathematically makes x to a vector. But we will here disregard this complication.)

The derivative dx/dt is mathematically defined as:

$$\frac{dX}{dt} = \lim_{\Delta t \to 0} \frac{X(t+\Delta t) - X(t)}{\Delta t} = f(X(t), t).$$

From this equation, we see that the model in Figure 1can be mathematically written as:

$$\lim_{\Delta t \to 0} \frac{X(t+\Delta t) - X(t)}{\Delta t} = f(X(t), t).$$

The only problem is that a computer cannot handle a time-step, Δt, that is infinitely small. Therefore, we have to settle with a *small* (but not infinitively small) time-step that we denote DT. Then we get:

$\frac{X(t+DT)-X(t)}{DT} \approx f(X(t),t)$, which can be rewritten as: $X(t+DT) - X(t) \approx DT \cdot f(X(t),t)$, or:

$$X(t+DT) \approx X(t) + DT \cdot f(X(t),t). \qquad \text{(Euler's method)}$$

[Alternatively: $\hat{X}\ (t+DT) = X(t) + DT \cdot f(X(t),t)$,
where $\hat{X}$ stands for *the estimate of* X.]

This last form is called **Euler's method**. (After the Swiss mathematician Leonhard Euler, 1707-1783.) It can be graphically illustrated, because the derivative dX/dt gives the tangent of the solution X at time t, and f = dX/dt, why we get the tangent direction from f. So, we shoot from X(t) in the f(t) direction the distance DT forwards in time to get the new *estimate of* X (denoted $\hat{X}$) at time t+DT. See Figure 2.

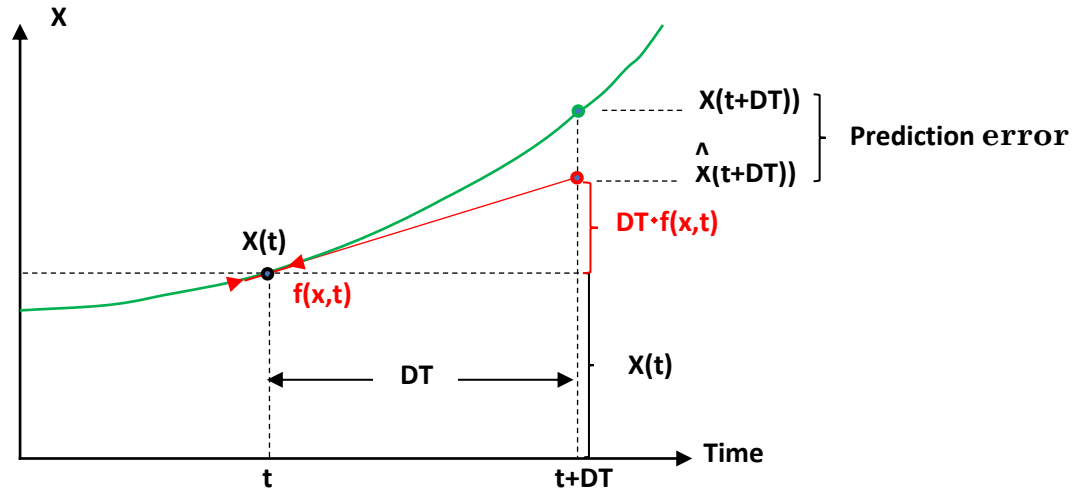


Figure 2. Difference approximation using Euler's method. Note that the derivative of x is dx(t)/dt=f(x, t), which graphically means that f(x, t) is the tangent to the curve x at time t. So Euler's method means that you are at point x(t) and shoots in the direction f(x, t) the distance DT forwards in time. This gives the new point: $\hat{x}(t+DT) := x(t) + DT \cdot f(x,t)$. However, $\hat{x}(t+DT)$ deviates from the true value $x(t+DT)$, creating a prediction error.

As seen from Figure 2, the **prediction error**, $x(t+DT) - \hat{x}(t+DT)$, comes from the necessary approximation when going from mathematics to numerics. (Theoretically there is also a **rounding error** because of the precision of the computer. However, with modern computers using 64 or 128 bits, this is usually negligible.)

The prediction error is a **local error** that arises when estimating one time-step ahead. However, the simulation continuous for many steps, cumulating the error. This cumulated error is called the **global error**.

For the Euler algorithm used here, the prediction error decreases (about quadratically) when DT is made smaller. This means that if we reduce DT by a factor 10, the prediction error becomes about 100 times smaller. However, we will then need 10 such time-steps to reach

the same point in time, so the gain in precision will be *about* 10/100 = 1/10 (somewhat depending on the model).

Euler's method is of **order 1** since the *global error* is reduced by about a factor of 10 when DT is reduced to a 10:th of its size. We will later on discuss higher order methods.

Exercise 1

Now we will test different time-steps, DT, on a simple linear system containing a Stock with a constant InFlow and an OutFlow that drains the stock with a constant fraction per time unit. This constant fraction, c, we Invert to get the time constant, T, so c = 1/T. This system is described In Figure 3.

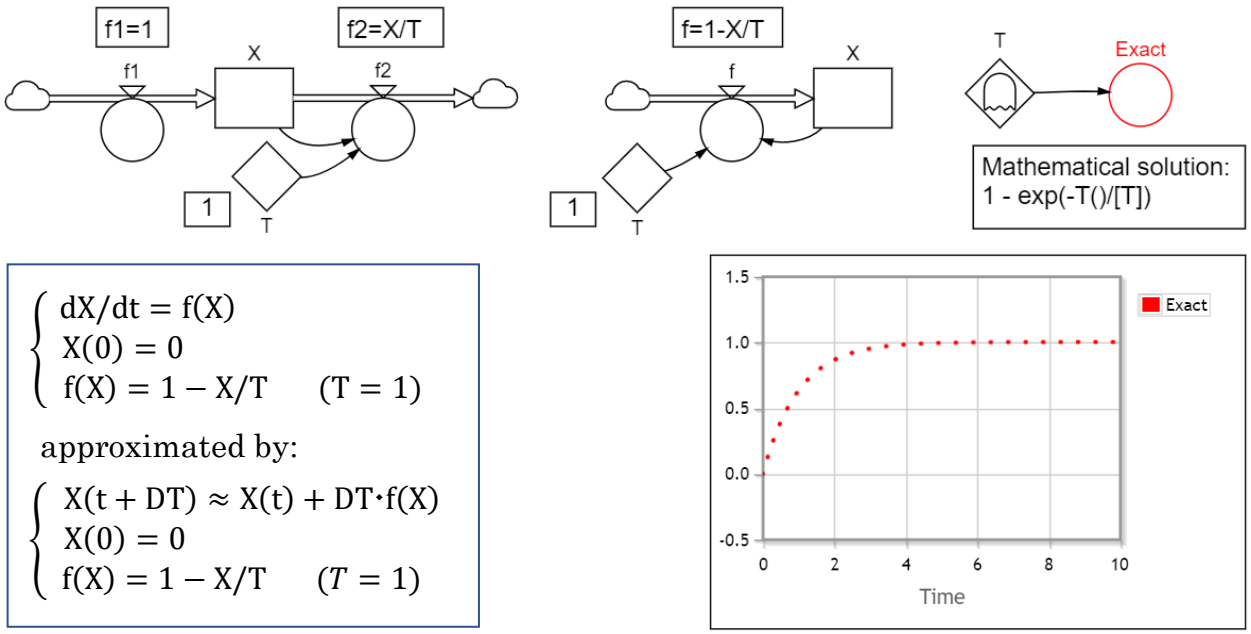


Figure 3. The studied system described by Stock, Inflow and OutFlow or by Stock and NetFlow. The mathematical description must be replaced by a numerical approximation – here using Euler's method. Further, the mathematically exact solution is shown in the Time Plot.

Mathematically, this system is described by:

$$\begin{cases} \frac{dX}{dt} = f;\ where\ X(0) = 0 \\ f = 1 - X/T \end{cases}$$

and has the exact solution: $X(t) = 1 - e^{-t/T}$, which is a *smooth, stable, non-oscillating* curve shown in Figure 3, above.

However, computers understand numerics but not mathematics, why we have to translate the mathematics into a numerical form using e.g. Euler's method:

$$\begin{cases} X(t + DT) = X(t) + DT \cdot f(X);\ \ where\ X(0) = 0 \\ f(X) = 1 - X(t)/T \end{cases}$$

The calculations proceed according to:

Time	X(t)	f(x, t)=1-X/T
0	X(0) = 0 (Initial Value)	f(0) = 1-X(0)/T
DT	X(DT) = X(0)+DT·f(0)	f(DT) = 1-X(DT)/T
2·DT	X(2·DT) = X(DT)+DT·f(DT)	f(2·DT) = 1-X(2·DT)
etc.	…	…

You see the pattern; X(0) is given. Then calculate f(0).

Advance time by DT, update X(DT) from X(0) and f(0), and then calculate f(DT).

Advance time by DT again, update X(2DT) from X(DT) and f(DT), and then calculate f(2DT).

etc. (These calculations can easily be done in a spreadsheet.)

Task: Build one of the versions of the model in Figure 3. Also make an auxiliary for calculation of the mathematically exact value: Exact = 1 – exp(-t/T). (Note that time, t, is the function T() and the time constant is the primitive [T].)

Set DT=0.1, 0.2 and 0.5 and simulate for 10 time units and compare X with the Exact solution in a Time Plot. Comment what you see in Table 1 about accuracy, oscillations and lost stability.

Table 1. Simulation of the model: X(t+DT) = X(t) + f(x), f(t)=1-X/T, where T=1 and DT is varied.

Time constant T	Time-step DT	Max X error	**Comment** (Compare with the Exact solution)
1	**0.1**		
1	**0.2**		
1	**0.5**		
1	**1.0**		
1	**1.5**		
1	**2.0**		
1	**2.5**		

Now, set the time constant to 0.15 time units. What DT do you now need to get a good precision? About how much smaller than T should DT be?

Answer: ……

…….. ■

2. Different types of algorithms to update the model over time

Euler's method is easy to understand from the definition of derivative (dx/dt=f) as shown in Figure 2, above. To understand the ideas behind higher order methods we must introduce the Taylor series that tells how a well-behaved ('analytical') function develops over time if you know all the derivatives. (Brook Taylor: English mathematician, 1685-1731.) Don't panic – it can be described in plain English.

The Taylor series assumes that you are at X(t) and want to estimate the value of X at t+DT for a well-behaved model. It then states that if you also can calculate all derivatives of X (X', X'', X''', etc.) then you can exactly calculate the value X(t+DT). The Taylor series states:

$$X(t+DT) = \underbrace{X(t)}_{Value} + \frac{DT}{1}\underbrace{X'(t)}_{Slope} + \frac{DT^2}{1\cdot 2}\underbrace{X''}_{Curvature} + \frac{DT^3}{1\cdot 2\cdot 3}\underbrace{X'''(t)}_{Change\ in\ curvature} + \frac{DT^4}{1\cdot 2\cdot 3\cdot 4}\underbrace{X''''(t)}_{\dots} + \cdots$$

So, the next point X(t+DT) can be exactly predicted if you know the Value, Slope (1:st derivative=f), Curvature (2:nd derivative), Change in curvature (3:rd derivative), etc. of X at time t. Also note that that the terms with higher derivatives are less and less important since they are successively scaled down. For example, with DT=0.1, *Value* has the weight 1, *Slope* has the weight 0.1, *Curvature* has the weight 0.05, *Change in curvature* has the weight 0.00017, etc.

If we truncate the series after two terms we get Euler's method: $X(t+DT) = X(t) + DT \cdot f(t) = X(t) + \frac{DT}{1}X'(t); where\ X' = dt/dx = f$, which only uses knowledge of *Value* and *Slope*. But if we <u>also</u> include knowledge about *Curvature* and *Change of curvature*, etc. (we have to truncate the series somewhere), then we can obtain much better precision for well-behaved models.

Higher order approximations incorporate additional terms from the Taylor series expansion. The higher the order the better the approximation for a given step-size dt. But more important, the faster the prediction error will approach zero when the step-size is reduced.

However, and still better! You know the *Value* of X at time t. For two points anywhere between t and t+DT you can draw a line and get the *Slope*. For three points you can adjust a circle with of proper size so you have the *Curvature*, with four points you get the *Change in curvature*, etc.

Based on these ideas, a differential equation can be approximated by a numerical equation in many different ways. Some examples are shown in Figure 4.

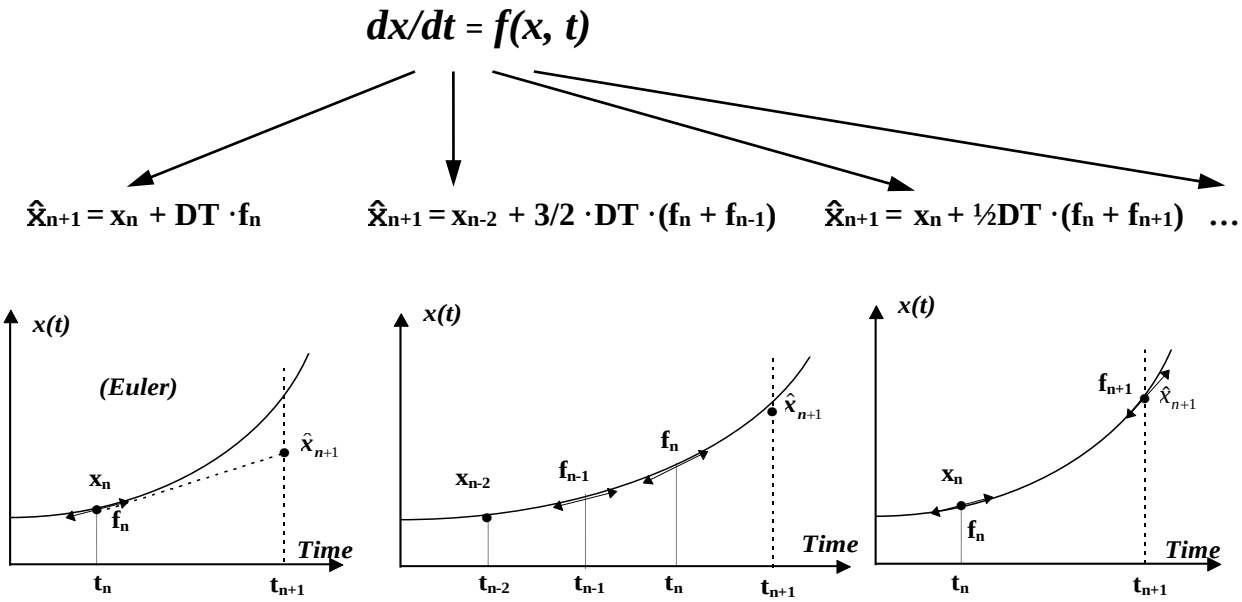


Figure 4. A differential equation can be approximated by a difference equation in many ways. (Here the current time-step is denoted t_n, the next t_{n+1}, and past time-steps t_{n-1}, t_{n-2}, etc.) The left diagram is Euler that shoots in the direction of *f*. The second uses old (already calculated) estimates and current to regard the curvature. The third first uses Euler to get a preliminary idea of $\hat{X}_{n+1(prel)}$. Then $f_{n+1(prel)}$ is calculated. Finally, a new $\hat{X}_{n+1}$ is calculated based on the average of f_n and $f_{n+1(prel)}$.

Of special interest are the so-called Runge-Kutta methods (after two German mathematicians, published 1900), which are usually abbreviated to RK1 (= Euler), RK2, etc. depending on the order of the method. In StochSD there are Euler and RK4 methods. The classical RK4 method use f_n makes two preliminary calculations of $f_{n+½}$ and one at f_n. Then a weighted average of these four *f*-estimates between *t* and *t+DT* are used. See Figure 5 and text below.

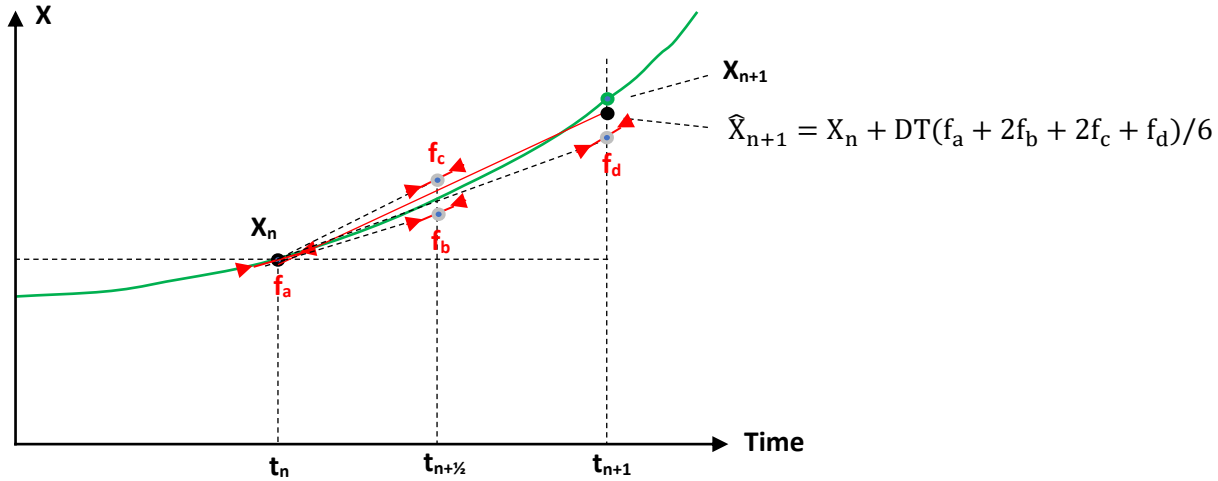


Figure 5. The working principle of a 4:th order Runge-Kutta algorithm is to estimate the next point $\hat{X}_{n+1}$ by weighting together the function values in four different points. See text below.

<u>The RK4 algorithm:</u>

1. From x_n calculate $f_a = f(x_n,t_n)$.
2. Shoot *DT/2* in the f_a direction to get x_b and calculate $f_b = f(x_b,t_{n+½})$.
3. Go back to (x_n,t_n) and shoot *DT/2* in the f_b direction, to get x_c and calculate $f_c = f(x_c,t_{n+½})$.
4. Go back to (x_n,t_n) and shoot *DT* in the f_c direction, to get x_d and calculate $f_d = f(x_d,t_n)$.

5. Now, take the actual step from time t_n to t_{n+1}, by going from (x_n,t_n) in the direction given by a weighted sum of the computed f_a, f_b, f_c and f_d (where you give double weights to the central f's): $\hat{X}_{n+1} = X_n + DT(f_a + 2f_b + 2f_c + f_d)/6$. □

The local error for the fourth order Runge-Kutta method is proportional to DT^5 why the global error is proportional to DT^4. So, if you reduce DT to a tenth, the local error is reduced by a factor 0.00001, and the global error by a factor 0.0001.

Next, we will test the precision of Euler and RK4 for a well-behaved model.

Exercise 2

Build the harmonic oscillator and include a Time Plot and an XY Plot according to Figure 6. (For the XY Plot set the 'Line Width:' to Thin!)

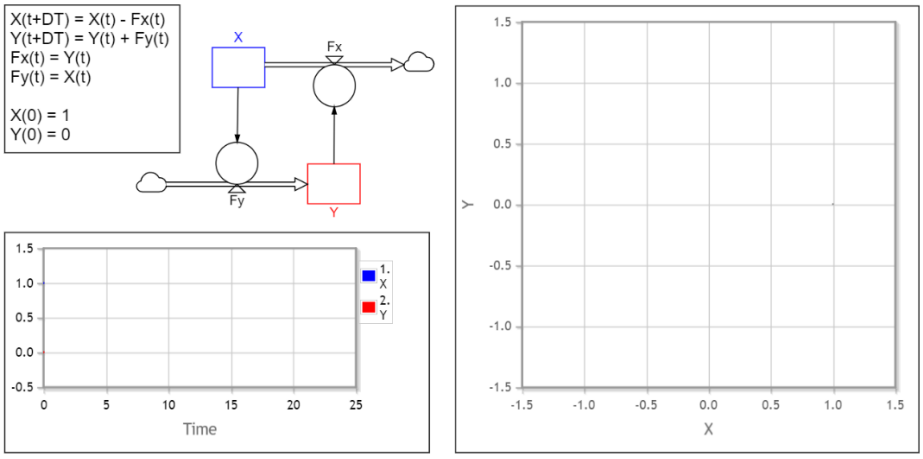


Figure 6. A harmonic oscillator.

Perform the simulations over 25 time units using the DT indicated in Table 2. Comment on the results in terms of 'Precision', 'Execution time' and 'Stability'.

Table 2. Simulation of the harmonic oscillator with Euler and RK4 for various DT.

Euler	RK4	Comment
DT = 1	DT = 1	Euler: RK4:_ _
DT = 0.1	DT = 0.1	Euler: RK4:_ _
DT = 0.01	[redacted]	Euler:
DT = 0.001	[redacted]	Euler:

[redacted]

3. Theoretical and practical rules for DT

From the above exercises, it is clear that step-size is an balance between *precision requirements* and *execution speed*. Further, for well-behaved deterministic models, a higher order method, e.g. RK4, could drastically reduce the execution time while preserving a high precision.

The need for *execution speed* does not manifest for small models to be run once. However, models can be very large, and for stochastic models you may perform 10 000 executions to obtain accurate statistical estimates and confidence intervals. Furthermore, an optimization requires many simulations when seeking an optimum.

Theoretically, it is often argued that the time-step should be about 10 times shorter than the smallest time constant (of course also depending on the demand for accuracy). However, this can be complicated, except for very simple models.

> ***The practical way to find an appropriate step-size*** is to test the model with different DTs by increasing or decreasing DT by a factor of e.g. 2 or 10. (You may also consider a more efficient integration algorithm.)

Only in exceptional cases you have a mathematical reference for the behaviour (of the *differential* equation model!) However, you can observe how a deterministic model *converges* when DT is decreased. So you decrease DT until further improvements are negligible (or not necessary).

Unless your interest of accuracy not only refers to a single number, you should use a plot for the selection of an appropriate DT (and method). In StochSD you preferably use the Compare Simulations Plot so you can see the results of different DTs in the same plot.

Exercise 3

The task of this exercise is to find an appropriate DT for Euler and for RK4 so that the maximal error (at any time) is less than 5% of X's value at that time for the model shown in Figure 7.

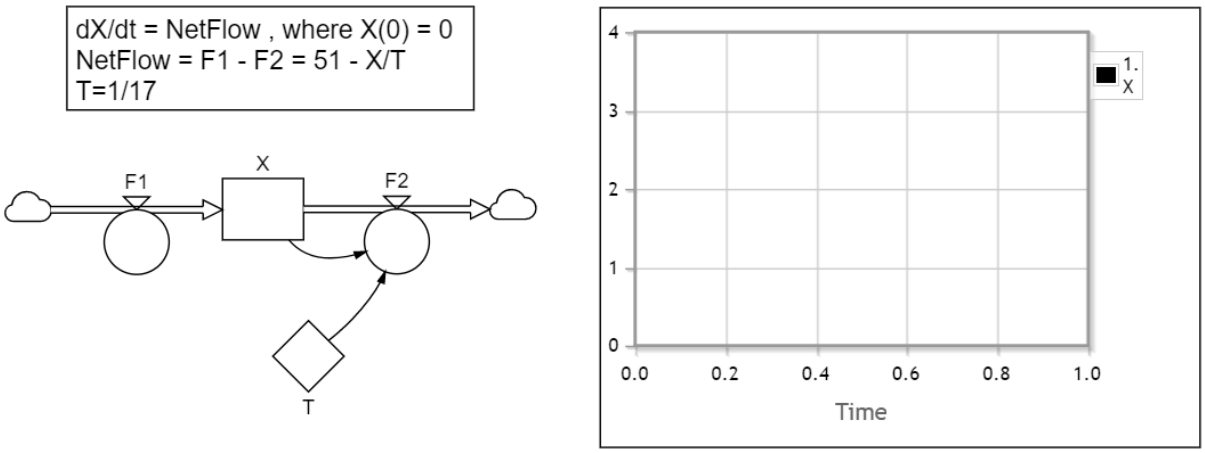


Figure 7. Find an appropriate DT for this model.

What is the value of the time constant T with two significant decimals.

Answer: ……

The model In Figure 7 is a first order linear system with a negative feedback. It has a smooth, stable, non-oscillating behaviour described by an exponential curve that approaches its final value.

If there exist a equilibrium value, it means that there are no further changes of X. Then the change-of-X term is there equal to zero (dX/dt=0). Mathematically you should therefore solve the trivial equation: $0 = -17 \cdot X + 51$.

What Is the equilibrium value of X? **Answer:** $X_{equilibrium}$ = ……

Now, build the model In Figure 7 and use a *Compare Simulation Plot* where you check the 'Keep Results' box so that you can compare the results from simulations with different DT.

What is a proper DT for Euler's method? **Answer:** DT = ……

What is a proper DT for the RK4 method? **Answer:** DT = …… ■

4. Stiff systems

A system of differential equation is said to be ***stiff*** if usual numerical methods for solving the equations introduce numerically instabilities unless the step-size is extremely small. This problem depends on the model, the initial conditions, and the numerical method used.

A dynamic system can simultaneously display several ***modes*** of behaviour. For example, a ship moves up and down through the waves with one frequency, rolls sideways with another and at the same time there might be motor vibrations in the ship with a much higher frequency. So, the system exhibits *both slow and fast changes*. The "typical time" for an oscillation or an exponential change is called the ***time constant***. It the size of the time constants are *very* different; the model is denoted stiff.

For example, the system $dX/dt = X/T$ has the solution $X = X(0) \cdot e^{-t/T}$ where T is the time constant. For higher order systems you might have an oscillatory behaviour like: $sin(t/T)$, where T is the time constant.

Stiffness may also arise for a system where the changes are slow during some part of the solution but very rapidly during other parts. Then stiffness becomes a problem because the time constant changes drastically.

So, not **to introduce numerical instability, the time-step must be small enough to handle the smallest time constant in a numerical model.**

Example: A system of masses and springs is shown in Figure 8a.

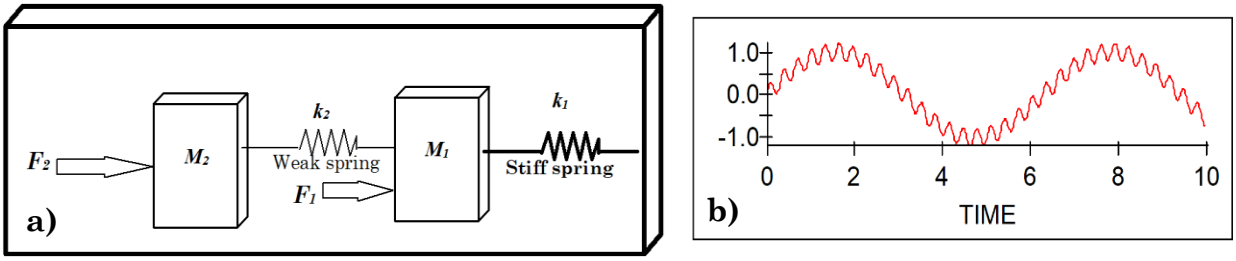


Figure 8. a) A system of masses and springs. If the string k_1 is is much *stiffer* than k_2 then M_1 will oscillate much faster than the mass M_2. **b)** The system displaying two modes of oscillations, one slow and one faster. In this example the time constants are differing in size about a factor of 20.

This system can easily be described by Newton's equations relating force to mass and acceleration. If the spring constant k_1 is much larger (*stiffer*) than k_2, then the mass M_1 will oscillate with a a much higher frequency (but with a much smaller amplitude) than mass M_2. See Figure 8b.

Although the ***stiffness*** concept comes from mechanical systems it is valid for all kind of systems.

Assume that you use a model of a boat, which describes both slow changes from the waves and fast changes from the engine vibrations. But now you use the model only to study the impact from the waves when the engine is turned off. However, the numerical calculations may still excite vibrations from the engine part of the model when the DT is not small enough – and the calculations may 'explode'! □

The problem with *modelling stiff systems* is that a very small step-size must be used to calculate the fast behaviour, not to lose the stability. *At the same time*, we must have a simulation time long enough to describe the slow variations. An integration method that otherwise would perform with great accuracy may even become unstable when applied to a stiff system. So, stiffness is an efficiency issue. It forces us to use very small time-steps or to use a more stable integration method.

In simulation modelling, *the best is often to avoid stiffness whenever possible*. If the purpose, in the mechanical example above (Figure 8), is to study the motion of M_2, we can perhaps replace M_1 with a rigid wall. If we are more interested in the motion of M_1, we can instead keep the simulation time period so short that M_2 does not have time to move noticeably during the simulation. M_1 and M_2 are then studied by different models.

In some CSS languages (not in StochSD) there are special algorithms for handling stiff models. In principle, such methods build on updating X_n by f_{n+1} rather than by f_n. These methods are called *implicit* methods because you must first estimate an $\hat{X}_{n+1}$ from which you calculate f_{n+1}. The simplest of these methods is *the implicit Euler method*., shown in Figure 9.

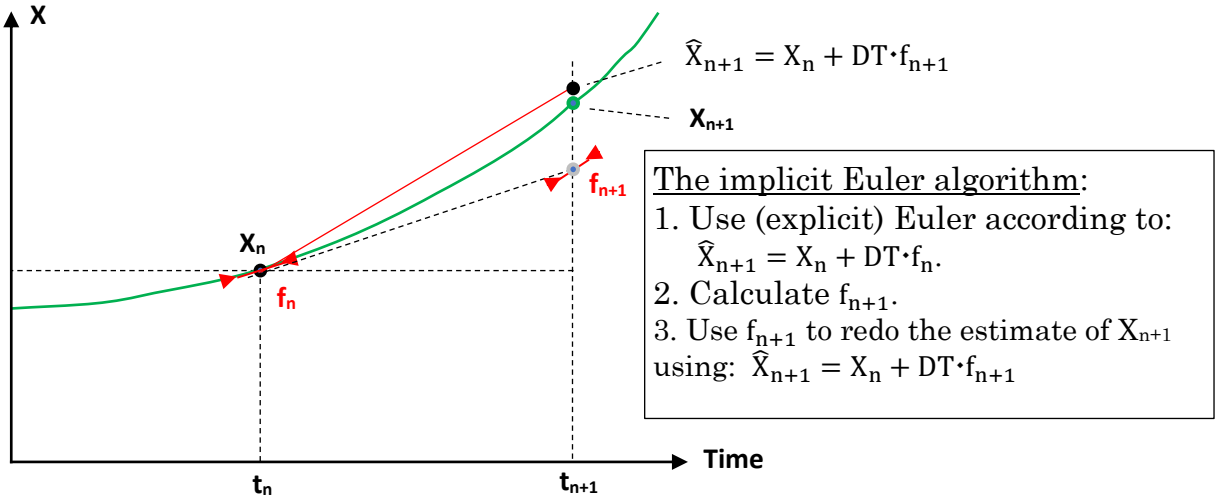


Figure 9. The implicit Euler method. This method is significantly slower than the (explicit) Euler method, but can handle stiff problems much better.

Metaphorically, when you push a boat or car from behind (f_n) the motion becomes more unstable than when you stand in front of it (f_{n+1}) and pull it.

The advantage of implicit methods is that they can be used for simulation of stiff models. The implicit methods allow much larger steps than similar explicit methods. However, the numerical costs for calculating a step are larger than for explicit methods, which gives a longer execution time.

Exercise 4

The mathematical model (where we are only interested in X_1)

$$\begin{cases} dX_1/dt = X_2 \\ dX_2/dt = -101 \cdot X_2 - 100 \cdot X_1 \quad (\text{where } X_1(0) = 1 \text{ and } X_2(0) = 0) \end{cases}$$

has the exact solution: $X_1 1(t) = \frac{100}{99} e^{-t} - \frac{1}{99} e^{-100t} \approx e^{-t}$.

However, on a numerical form we get:

$$\begin{cases} X1(t+DT) = X1(t) + DT \cdot F1 \\ X2(t+DT) = X2(t) + DT \cdot F2 \\ F1 = X2 \\ F2 = -100 \cdot X1 - 101 \cdot X2 \\ \text{where } X1(0) = 1 \text{ and } X2(0) = 0 \end{cases}$$

This is a moderately stiff model where the time constant of X1 is T1=1 which is much larger that T2=0.01 for X2. Then, when translated into a numerical form, the presence of the small and fast vanishing e^{-100t} term is enough to disturb the numerical computations unless a very small step-size is used.

Task: Make a numerical model of this system according to Figure 10.

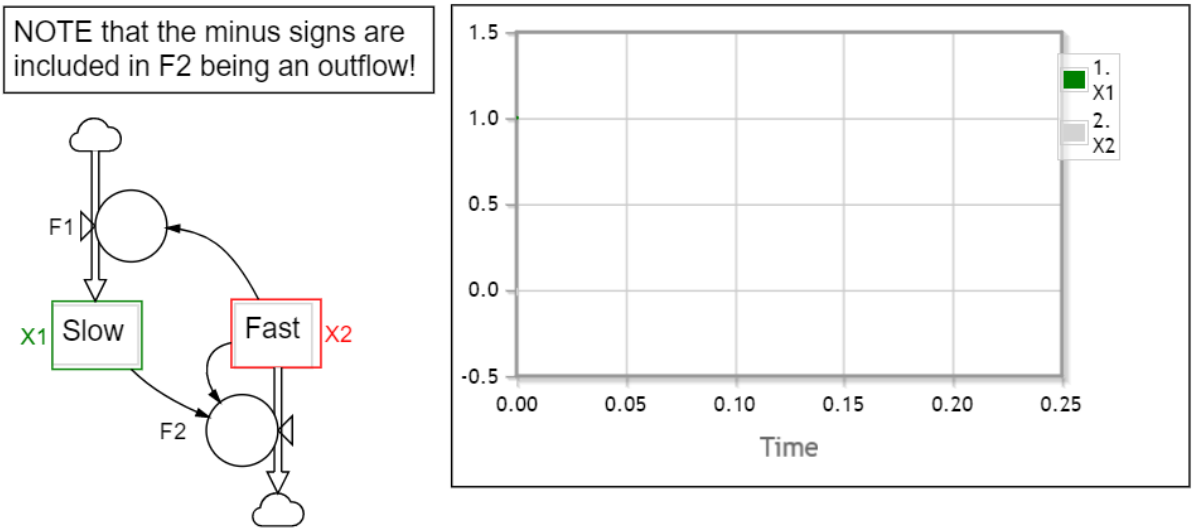


Figure 10. A moderately stiff system to be simulated with DT=0.025.

Now, we will study what will happen when we use DT=0.025. To see the effects, we will start with a small simulation period (Length) and then successively increase the Length in order to see what happens, when using the same DT. Study what happens in a Time Plot where X1 and X2 are displayed – but don't lock the scaling (Let it be 'Auto')!

Task: Follow the Instructions and Comment, in Table 3, on what you see using terms such as 'accuracy', 'oscillations' and 'overflow' (i.e. larger numbers than the computer can handle).

Table 3. Simulation of a stiff model with Euler and DT=0.25 time units, where the Length is successively increased (not to be confused of the large changes of the Y-axis).

DT=0.025	Instructions	Comments on X1 and X2
Length = 0.25	X1 and X2 on Left Axis	
Length = 1	– " –	
Length = 5	X1 Left & X2 Right Axis	
Length = 50	– " –	

This means that a model with a very smooth and slowly converging mathematical solution (e.g. $\approx e^{-t}$ as in this case) can fool you to use a too large DT where there is a hidden mode with a much smaller time constant. The *numerical* calculations (that are never exact) may then make the calculation process unstable until the computer can't handle it and stops because of *overflow*.

For (stiff) models, you will get into trouble if you don't use a very small step-size. You may for this model try a smaller DT to see that it works, but it consumes more execution time! ■

5. Problematic model structures

In CSS, artefacts can also be generated that are not directly about the step-size or integration method.

5.1 Lanchester's model

Lanchester's model is an example where you may over-empty a Stock that by nature cannot be negative. (We met this model already in Lab-3, Section 5)

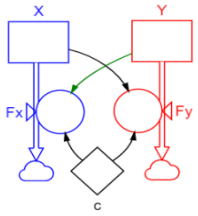
Lanchester's model is a simple model of a battle between two fighting forces (soldiers, aircrafts, battle ships, etc.). Force X is characterised by its initial number of entities, $X(0)$, and hitting power, a, and Force Y by its initial number, $Y(0)$, and hitting power, b. The hitting power is the number of eliminated entities per time unit that each member can inflict on the enemy. Here we let the hitting powers be the same, c (i.e. $c = a = b$).

The combat is won when the last entity of the losing force is eliminated.

Exercise 5

Build the model in Figure 12 and simulate it for 30 time units with a proper step-size. You will then see something like in Figure 12.

$$\begin{cases} dX/dt = Fx, \text{ where } X(0) = 5 \\ dY/dt = Fy, \text{ where } Y(0) = 4 \\ Fx = c \cdot Y \\ Fy = c \cdot X \\ c = 0.1 \end{cases}$$



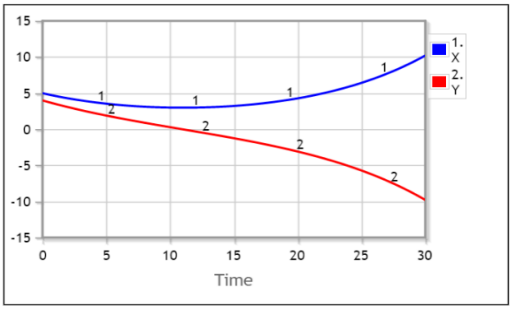


Figure 12. Lanchester's model. The battle is over at about 11 time units – but what happens then?

Task: Explain what happens after force Y is eliminated!

Answer: Force Y ……

and Force X …… ■

In most models, the births and deaths, for example, of a population are proportional to the size of the population. However, in Lanchester's model, the reduction of e.g. Y is *not* proportional to Y, but to X, and vice versa. So, the reduction of Y will not cease when Y vanishes – but continues to negative numbers.

<u>Ways to handle this problem</u>

There are several ways to handle the problem of negative numbers (of soldiers, vessels, aircrafts, etc.) which we here call zombies. Since Y becomes negative the Fx=c⋆Y changes sign and the fight with the zombies will create more units of X! If you create zombies in the model, various processes may start to run backwards.

On way to handle this is to terminate the simulation when one force gets extinct, as was done in Lab-3, exercises 6, 7 and 8. There, an IF-statement was used that checked the condition of X and Y and stopped the simulation when a force got extinct.

However, the battle model may just be a part of something bigger like a war. So, then we want to continue the simulation without introducing zombies. Also, here you can use IF-statements. In StochSD more powerful mechanisms are already at place in form of "☐ Restrict to non-negative values" that exists for Stocks and Flows.

Task: Check '☐ Restrict to non-negative values' for Stock Y, repeat the simulation and describe what happens.

Answer:

……...

If you are uncertain of who will win, you can of course check the restriction for both the X and Y Stocks. (However, **do not** use this specific facility without a clear reason. In many cases (economy, temperature etc. a negative value may be legitimate. And in other cases, you may just hide an error that show up for a different reason.) ■

5.2 A specific problem

There are some models that shows a very odd behaviour when a Stock has a content smaller than the drain DT·F_{out}. We will first present such a model and study it before we discuss this further.

Exercise 6

We will now study the emptying of an open water tank with vertical sides.

According to Torricelli's law, the speed, v, of the outlet water from the bottom of an open water tank filled with h meters of water is $v = \sqrt{2gh}$, where g=9.81 m/s^2 is the acceleration due to gravity constant. This means that the change in water level, h, is $\frac{dh}{dt} = \frac{a}{A}v = \frac{a}{A}\sqrt{2gh}$, where a is the area of the outlet and A of the barrel (chosen so that $\frac{a}{A} = 1/\sqrt{2gh} \approx 0.226$ in order to simplify), see Figure 13.

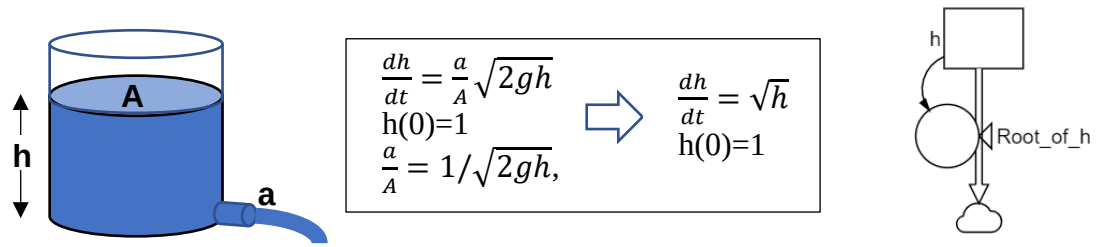


Figure 13. Emptying an open cylindrical water tank when the higth of water is **h**. (The otlet area a=0.226 of A is very large, but here we want the simpliest possible model when focusing on the problem!)

Task: Build the model, initiate it with h(0)=1 meter and simulate it for 1 seconds, and then for 2 seconds with DT=0.1.

What happened? **Answer:** ……

Does it help to use a smaller DT, e.g. DT=0.01? **Answer:** ……

Explanation: When the remaining content (here represented by the height h) becomes smaller than DT·Root_of_h, then h becomes negative. In the next time-step the square root of a negative number is calculated, which makes the program to crash.

Independently of how small time-step you will use, it comes to a point where $DT \cdot \sqrt{h} > h$, because for h<1, $\sqrt{h} > h$, and more and more so the closer to zero you get. (for h=0.01, $\sqrt{h} = 0.1$, and for h=0.0001, $\sqrt{h} = 0.01$). So, a smaller DT takes you somewhat longer, but never all the way to h=0.

The remedy to this problem
Also, here you can use an IF-statement in the outflow that restricts the drain to at most what is in the barrel. However, in StochSD you can check "☐ Restrict to non-negative values" in the Stock.

Task: Check '☐ Restrict to non-negative values' for Stock h, repeat the simulation and describe what happens.

Answer: ……...

As already told, **do not** use this specific facility without a clear reason. ■

Note: For the common structure to the right in Figure 13, this will not happen when the outflow is proportional to what is in the stock. But if a specific function (like the root function) increases its value relative to the content in the drained compartment when you get closer to an empty Stoch, you may have this problem.

II. TIME HANDLING FOR STOCHASTIC MODELS

6. Finding an appropriate time-step for a stochastic model

For deterministic models the algorithms are based on well-behaved (so-called analytical) functions that can be described by the stock value, its first derivative, second derivative, etc. according to the Taylor series, see Section 2.

For stochastic models, on the other hand, the random numbers will destroy the predictions obtained by the Taylor's series, and thereby the power of the higher order algorithms that are so successful for deterministic models. Using a higher order algorithm, e.g. RK4, will then just increase the computation time without creating accuracy. Therefore, Euler's method should be the standard method for stochastic models in StochSD.

Unfortunately, it is much harder to find an appropriate DT for a stochastic model. Even if you could specify the sequence of random numbers, a changed DT would not place the random sequence at the same places over simulated time. The only way is to simulate the stochastic model for a given DT, say 1000 times and then change the time-step and do it for another 1000 times to see if there is a statistically significant difference.

A heuristic method this author often uses is to find a proper time-step, DT, for a corresponding deterministic model (when possible), and then use DT/2 for the stochastic model. But if this is a good and secure strategy is still to prove. At least it is simple and can be used before the 1000 + 1000 replications just suggested, to guess a DT to start the search from. ■

7. The transition stochasticity problem

Already for the updating a *deterministic*, dynamic equation by Euler's method [X(t+DT) = X(t)+DT·(F_{in}(t)-F_{out}(t))], you include an error by keeping DT fixed during the whole time-step.

In the *stochastic case*, the corresponding algorithm: X(t+DT) = X(t)+DT·(F_{in}(t)-F_{out}(t)), where F_{in}(t) and F_{out}(t) include Poisson distributed random numbers, there is also another complication.

Assume that you have a Stock with a single radioactive atom that has the probability c of decaying per time unit modelled by the output Decay = Poisson(DT·c·1)/DT, then Poisson(DT·c·1) most often gives you the outcome of 0 and sometimes 1, but it could also give 2, 3, … etc. although occurring more seldom. Then the Stock will be *over-drained* and becomes negative. For smaller DT, e.g. DT=0.1, 0.01, etc. this problem becomes less frequent, but may still occur.

Often you can accept that it happens a few times without significantly affecting the statistics of the study. However, if a negative Stock may drive other processes backward, this can be a serious problem. Then, checking "☐ Restrict to non-negative values" in the Stock may sometimes handle this specific problem.

However, assume that the outflow Flow from Stock1, also is input to Stock2. Then, when DT·Flow gets the value 2, only one atom is removed from Stock1 (because of the restriction), but 2 atoms are added to Stock2. This would increase the total 'population' in Stock1+Stock2 as an unwanted side effect. (In this case, Decay = MIN(Poisson(DT*c*1), Stock1)/DT will solve the problem (without checking any restriction box). ■

Note: Off the record, **never use net-flows** in a stochastic model.

For example, assume that we have:

X(t+DT) = X(t)+DT·(F_{in}(t)-F_{out}(t))
F_{in} = Poisson(5)
F_{out} = Poisson(4)

Then, sometimes the random outcome from Poisson(5) may be 3, and the random outcome of Fout may be 6. The stock X will then decrease with 3 units. However, a net-flow of Poisson(5-4) = Poisson(1), can never have a negative outcome. So, a net-flow would have a very different impact on X and can as an example never decrease the value of X in this model. ☐

8. A queuing model

Now we will study a simple *queuing model* (of e.g. a shop) where the customers arrive randomly over the day (8 hours of 60 minutes = 480 minutes) at an expected rate, λ, of one

customer each 6 minutes (λ=1/6 customers per minute), and the average rate of serving customers is μ=1/5 customers per minute. (Of course, it is necessary that λ<μ for the system to be stable.) When both the number of arrivals λ and the number of served customers during a time period are Poisson distributed, which we assume here, this model is denoted an M/M/1 model.

A *deterministic* model using these average values would never show a queue building up in waiting for service. Therefore, a queue model has to be *stochastic*.

A proper M/M/1 model together with measuring devices for waiting and queue length, etc. is shown in Figure 14.

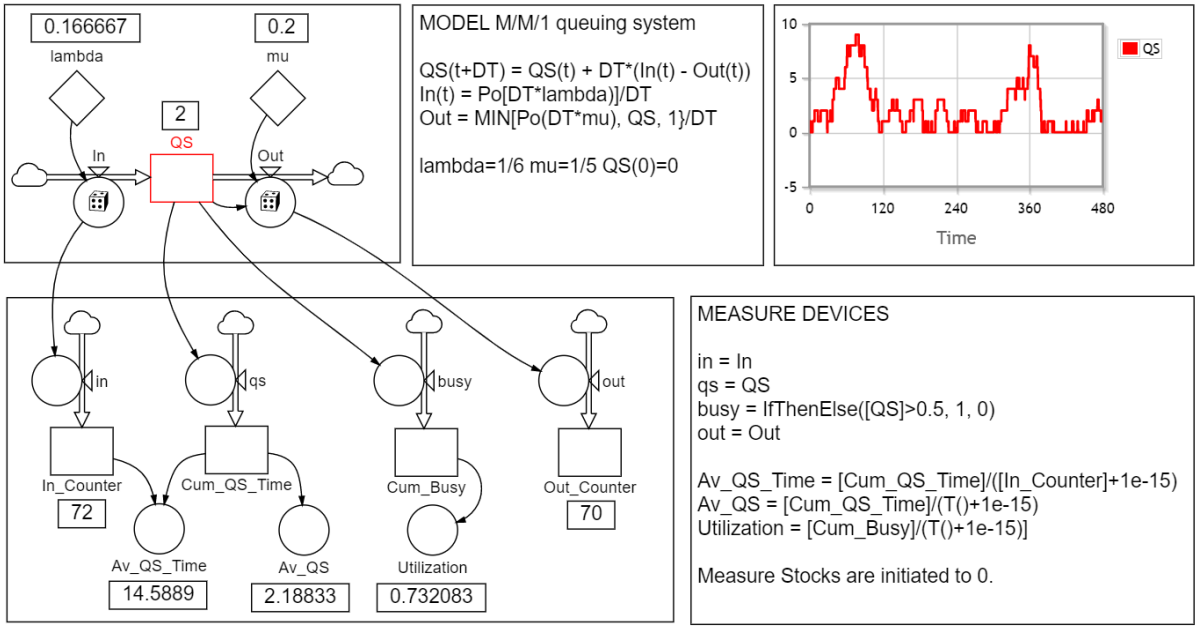


Figure 14. An M/M/1 queuing system. On the top row the model and below measuring devises. ('Av' stands for average.)

This M/M/1 algorithm joins the queue Q and the service S in a single stock denoted QS. (It is also possible to separate them into two stocks Q and S.)

The input to the stock QS in the M/M/1 formula In=Po(DT*λ)/DT is trivial, but the output Out=Min(Po(DT*μ), QS, 1)/DT requires an explanation. The kernel is Po(DT*μ)/DT. However, since μ>λ, QS would soon become negative if we only used this kernel. Therefore, we need a 'floor' in QS to prevent it from becoming negative. (DO NOT use the restriction option!) This is accomplished by surrounding the kernel with MIN(*kernel*, QS, 1) that selects the smallest of the three candidates, why the kernel Po(…) can never remove more than the existing number of QS persons in the stock. Further, even when Po(…) or QS are large, no more than one customer can be served during a DT.

Task: Build the stochastic model with the measuring devices shown in Figure 14, and then find a proper DT. (Note: 'Po(…) refers in StochSD to RandPoisson(…). You could use PoFlow(…), but then you have to modify the equations accordingly.)

Simulate the model for a working day of 480 minutes to see in a Time Plot how the queuing situation and the utilization of the service will vary drastically from day to day.

Exercise 7

For *deterministic* models, it is easy to find a proper DT, as discussed above. But for a *stochastic* model this is not so trivial, and it requires a much larger effort.

To find a proper DT, use StatRes to make, say, 500 replications of the model with DT = 1, 0.5, 0.2 and 0.1 minutes, and fill in **average values** and **confidence intervals** for: Average time, Average queue length (including the service), and Utilization of the server in Table 4, below.

Table 4. 500 replications for different values of DT. Fill in both averages and 95% confidence intervals according to: "*Average* (*low value – high value*)".

DT	Av_QS_Time	Av_QS	Utilization
1	(-)	(-)	(-)
0.5	(-)	(-)	(-)
0.2	(-)	(-)	(-)
0.1	(-)	(-)	(-)

When there are no significant improvements for a smaller DT, then keep the DT. (You can see the uncertainties of the average estimates in the confidence intervals.)

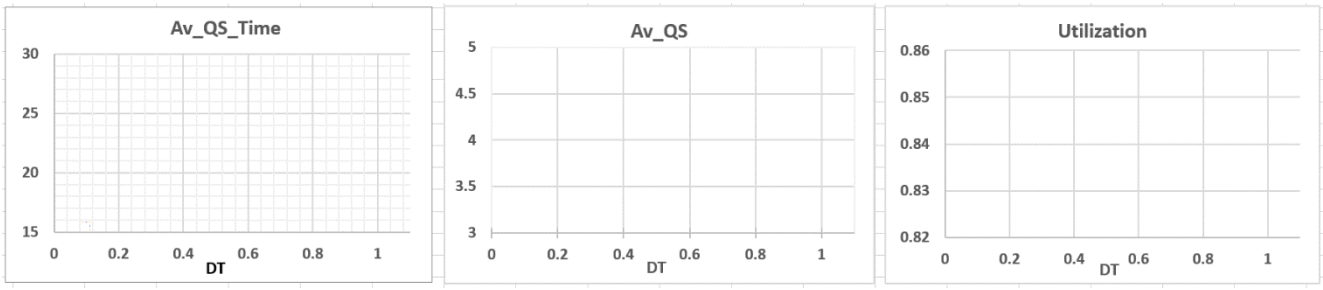


Figure 15. Fill in the values from Table 4 in the three diagrams with three curves each (Mean value, Low C.I. value, and High C.I. value).

What seems to be a proper step-size to get reasonable results for Av_QS_Time, Av_QS and Utilization?

Answer: DT = …… minutes.

Comment: Sometimes theoretical results for a model are known. For example, for the M/M/1 model the equilibrium values are known. Then you could initiate QS to its equilibrium value an make many long replications to find a proper DT. (Thereafter, you can reuse the model with the proper DT, initiated as an empty shop, and specify (in the model) how you should handle the customers that are there after the workday is over. (Probably, you don't want to lock them in until the next work day. But do you serve them after closing time, lock the in-door beforehand, or just ask the customers to return tomorrow?) ■