Learning Curves Redux: A New Use for a Familiar Tool

Evin Stump
Senior Systems Engineer
Galorath Incorporated
30584 Bay Hill Drive
Temecula, CA 92592
o: 951.676.7804
f: 951.694.6824
estump@galorath.com

Alexandra Minevich
Systems Engineer/Cost Analyst
Galorath Incorporated
100 N. Sepulveda Blvd. Ste. 801
El Segundo, CA 90245
o: 310.414.3222
f: 310.414.3220
aminevich@galorath.com

Abstract

This paper proposes another use for learning curves, namely the scheduling of production operations. This is not entirely a new idea, but it is usually not formally implemented in the planning of production operations. Learning curves alone can't do this planning job effectively, but when combined with other appropriate, relatively simple logic, the result can be an automated scheduling process that predicts not only the cost of each produced item, but also the dates when production of each item will start and finish, plus a spread of labor hours and material costs by month, or even by week, if desired.

The method can be applied to multiple production lots of arbitrary size, making it useful for many situations, including block production, simultaneous production at multiple facilities, and LRIP/FRP situations. In this paper, we demonstrate the process by showing methods for use of learning curves to schedule production in three planning modes:

- Constant effort, with the same labor hours expended in each time period
- Uniform rate, with the same production quantity in each time period
- Ramp-up, beginning at zero rate and accelerating linearly to a maximum rate.

User input is minimal. For each production lot, the user specifies production quantity, start date, finish date, learning curve theory and slope, mode of scheduling, and information about transfer of learning from a previous lot, if appropriate.

Introduction

Currently, Galorath's SEER-H hardware model requires users to enter production quantities by year. While this provides a rough idea of the timing of production costs, some users would like to see cost flow more accurately defined. That is the principal motivation for the methods described in this paper. They will appear this year (2007) in the release of SEER-H version 7.1.

Traditional learning curve theory underlies the methods to be described. Some ancillary calculations extend the basic capability provided by the learning curve math. This is done without requiring exhaustive (and exhausting!) user input. The methods automatically adjust production concurrency to account for user imposed overall duration of production, whether that be long or short.

The current SEER-H model treats all of production as a single lot, albeit it can be partitioned into annual quantities. In the new approach, the user can create any number of production lots, each with its own learning slope, average labor rate, start date and finish date. Lot to lot transfer of learning, full or partial, is accommodated. The multi-lot structure readily deals with low rate initial production/full rate productions situations, production in more than one factory, and block production situations.

In creating these improvements to SEER-H, several issues were encountered and dealt with. One is the issue of production concurrency, that is, the number of units being produced at one time. Concurrency has considerable importance because it affects not only the demand for labor but also the demand for tooling, facilities, and capital equipment.

Other issues relate to material purchases. Two common industry practices, currently unaccounted for in SEER-H, are the application of learning to material purchases, and the advance procurement of material, usually known as long lead procurement.

Another issue dealt with is breaks in learning. Whiled not a major problem in many projects, it happens often enough that we decided to address it.

Production Concurrency

In the revisions being created, the model user is required to set a date for start and finish of each lot, as well as a learning slope and a production quantity. In doing this, the user may, unaware, establish a concurrency of production (e.g., multiple production lines, or labor resource overload) that the factory cannot accommodate. To guard against this possibility, we have developed an empirical equation that SEER-H will use to help the user set start and finish dates realistically. The equation is:

$$D = (1 + Q - N)MQ^{B}$$

In this equation:

- D = approximate duration of production, months
- Q = user defined production quantity
- N = user estimated average number of units in production at one time (i.e., concurrency)

- M = user estimated time in months to produce unit #1
- B = learning curve natural slope

The concurrency issue is important because it bears on the peak touch labor staffing requirement, tooling required, factory floor space required, and related matters. An estimate of the concurrency required can be obtained by solving the above equation for N:

$$N = 1 + Q - \frac{D}{MQ^B}$$

<u>Example:</u> Suppose that you want to build Q= 15 units of an item in D = 12 months, and the learning slope is assumed to be 87% (B = -0.2 by the usual formula).[1] You estimate the time required to build the first unit as M = 1.5 months. Thus:

$$N = 1 + 15 - \frac{12}{(1.5)(15)^{-0.2}} = 2.25 \text{ units in production concurrently}$$

A reasonable conclusion from this result is that the staffing needs to support 2.25 units on average in production, but the facilities and tooling may need to support building three units concurrently.

Figure 1

Duration vs. Concurrency (10 Units Produced; 87% Learning Slope)

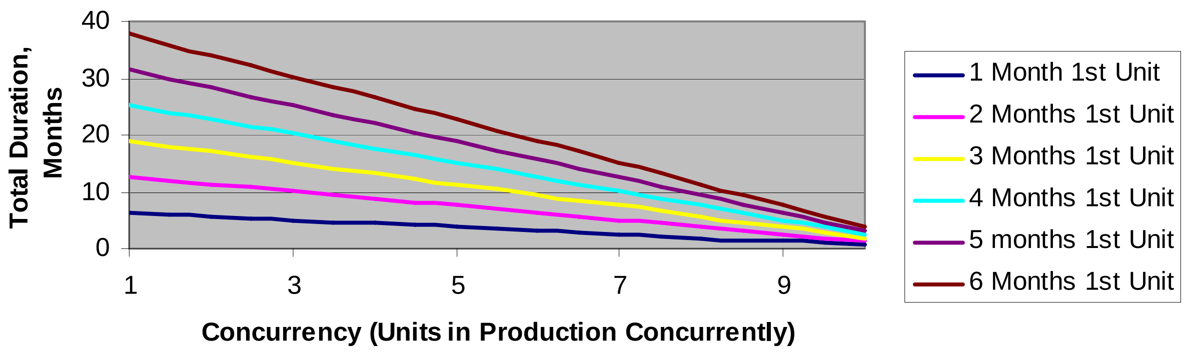


Figure 1 illustrates a use of this math. The plot shows total duration of production in months versus concurrency for production of ten units. The time required to build the first unit ranges for one to six months with an 87% learning slope.

[1] B = log(S/100)/log(2) where S is the slope in percent.

Modes of Scheduling Production

The best way to schedule production can change with circumstances. Currently, we plan to provide for three "modes" of scheduling, although we are still considering inclusion of other modes. The three modes currently planned are:

- Uniform rate
- Constant effort
- Linear ramp-up

As the name implies, in the uniform rate mode, units come off the production line at equal intervals of time. The labor effort required per unit is highest for the first unit and gradually decreases as learning occurs.

In the constant effort mode, the same number of labor hours is worked each month. Due to learning, the production rate gradually increases.

In the linear ramp-up mode, the production rate is initially zero, but it accelerates sufficiently in a linear manner to achieve the required production quantity in the allotted time. The linear ramp-up scheduling approach is sometimes used in low rate initial production (LRIP).

Conventional learning curve mathematics underlies all three scheduling modes. Specialized ancillary math differentiates them, as will be demonstrated shortly.

The demonstrations will be based on the unit theory of learning. Later, the difference in using the cumulative average theory will be explained.

Uniform Rate Mode

For the uniform rate mode, we want units to finish production at equal intervals of time. Specific inputs required of the model user for each lot are:

- Start date, TS
- Finish date, TF
- Production quantity, Q
- Labor learning slope, S%
- Average unit makespan, UM (average months required to build each unit)

Following are the steps we use to calculate the production scheduling and cost flow details:

Step 1: SEER-H calculates first unit labor hours, H_1, based on user input parameters.

Step 2: Calculate the production overall duration in calendar days:

$$DD = TF - TS$$

Step 3: Convert unit makespan from months to calendar days. (See the Appendix.)

$$UM = UM * 30.4375$$

(round to nearest day – 30.4375 is the average number of days in a solar month)

Step 4: Calculate the natural learning slope, B.

$$B = \frac{\log(S\%/100)}{\log(2)}$$

(both logarithms must be to the same base, but it can be any base)

Step 5: Calculate the labor hours for each unit produced, H_q, from q = 1 to Q.

$$H_q = H_1 q^B$$

Step 6: Calculate the total labor hours for all units to be produced, HT:

$$HT = \sum_{q=1}^{Q} H_q$$

Step 7: Calculate the finish time of each unit, T_q, also the beginning time, t_q, and the labor hours spread by months. (Note: there can be concurrency of production, with more than one unit being produced at the same time. The algorithm establishes the necessary concurrency automatically.)

Finish time of each unit:

$$T_1 = TS + UM$$

$$T_q = T_{q-1} + \frac{DD - UM}{Q - 1} \quad (\mathrm{q} \neq 1)$$

Start time of each unit:

$$t_1 = TS$$

$$t_q = T_q - UM \quad (\mathrm{q} \neq 1)$$

Note that T_q and t_q generally are not integers. For details of calculation of labor hours spread by month, we present now a fully worked example.

Example: A company wants to produce a lot of 6 units of a product, with starting date November 15, 2006 and finishing date May 20, 2007. The first unit labor hours estimated by SEER-H are 3,975. The planned learning slope is 87%. The unit makespan of production for each unit is estimated at 1.5 months. Produce a production schedule and labor spread by the method described above.

Step 1: Obtain the first unit hours from SEER-H, H_1. (3,975)

Step 2: Calculate the production duration in calendar days, DD.

$$DD = TF - TS$$

Using the method in the Appendix, the two dates are separated by 186 calendar days; DD = 186.

Step 3: Convert the unit makespan from calendar months to calendar days:

$$UM = 1.5 * 30.4375 = 46 \text{ days}$$

(rounded to nearest day)

Step 4: Calculate the natural learning slope, B.

$$B = \frac{\log(S\%/100)}{\log(2)} = \frac{\log(87/100)}{\log(2)} = -0.2$$

Step 5: Calculate the labor hours for each unit produced, H_q, from q = 1 to Q.

$$H_q = H_1 q^B$$

Unit	Hours
1	3975
2	3460
3	3191
4	3012
5	2881
6	2778

Step 6: Calculate the total hours for all units to be produced, HT:

$$HT = \sum_{q=1}^{Q} H_q$$

Summing the table above yields HT = 19,297 hours.

Step 7: Calculate the finish time of each unit, T_q, also the start time, t_q, and the hours spread by months.

Finish time of each unit:

$$T_1 = TS + UM$$

$$T_q = T_{q-1} + \frac{DD - UM}{Q - 1} \quad (q \neq 1)$$

Start time of each unit:

$$t_1 = TS$$

$$t_q = T_q - UM \quad (q \neq 1)$$

Finish time of first unit:

November 15, 2006 + (186-46)/(6-1) = December 31, 2006

Finish time of second unit:

T_2 = December 31, 2006 + 28 = January 28, 2007

Finish time of third unit:

T_3 = January 28, 2007 + 28 = February 25, 2007

Finish time of fourth unit:

T_4 = February 25, 2007 + 28 = March 25, 2007

Finish time of fifth unit:

T_5 = March 25th, 2007 + 28 = April 22, 2007

Finish time of sixth unit:

T_6 = April 22, 2007 + 28 = May 20, 2007

Now we continue by finding the begin dates of each unit.

Begin time of first unit:

t_1 = November 15, 2006

Begin time of second unit:

t_2 = January 28, 2007 – 46 = December 13, 2006

Begin time of third unit:

t_3 = February 25, 2007 – 46 = January 10, 2007

Begin time of fourth unit:

t_4 = March 25, 2007 – 46 = February 7, 2007

Begin time of fifth unit:

April 22, 2007 – 46 = March 7, 2007

Begin time of sixth unit:

May 20, 2007 – 46 = April 4, 2007

Here is a complete tabulation of the start and finish dates:

Unit	Start	Finish
1	11/15/2006	12/31/2006
2	12/13/2006	1/28/2007
3	1/10/2007	2/25/2007
4	2/7/2007	3/25/2007
5	3/7/2007	4/22/2007
6	4/4/2007	5/20/2007

(A report capability similar to the above is planned for SEER-H v7.1.)

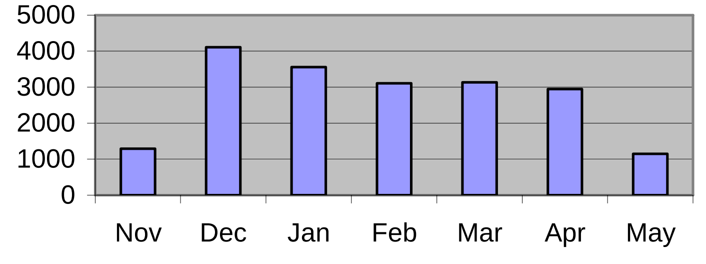
The 3,975 hours for unit #1 are allocated to the months in which that unit is produced. Production of this unit occurs in November and December, 2006. Between these two dates there are 46 days, of which 31 are in December and 15 are in November. The allocation to November is (15/46)(3975) = 1,296. The remainder is allocated to December: 2,679. Proceeding in this manner, we can build the following allocation table (note the concurrency of production – on average about 1.17 units are in production at one time according to the concurrency algorithm previously given, but the peak is two units):

Unit	NOV	DEC	JAN	FEB	MAR	APR	MAY	Totals
1	1296	2679						**3975**
2		1429	2031					**3460**
3			1526	1665				**3191**
4				1441	1571			**3012**
5					1566	1315		**2881**
6						1631	1147	**2778**
Totals	**1296**	**4108**	**3557**	**3106**	**3137**	**2946**	**1147**	**19297**

(A report capability similar to the above is planned for SEER-H v7.1.)

Plotting the above hours versus months yields the following:

Figure 2 -- Example Labor Hours Spread by Month Constant Rate



Constant Effort Mode

In the constant effort approach to production scheduling, the same number of labor hours is expended in each month. The problem is to determine how many units can be built in each month for those hours, given that the workers become more efficient each month due to learning. Each month, the number of units built is larger than the month before.

When dealing with cumulative hours for a number of production units, the unit learning theory is non-linear and complex, and exact closed form calculations are not feasible. Accordingly, we will use an approach called the Hungry Algorithm, plus some reasonable approximations, to arrive at allocations of labor hours by month and start and finish dates for each production item.

User inputs required for each production lot are as follows. Note that the Unit Makespan required for the Uniform Rate method of planning is not required here, because the unit makespan for constant effort is variable. .

- Start Date, TS
- Finish Date, TF
- Production Quantity, N
- Learning Slope % (Labor), S%

Step 1: SEER-H calculates first unit labor hours, H_1, based on user input parameters.

Step 2: Calculate the production overall duration in calendar days, DD.

$$DD = TF - TS$$

Step 3: Calculate the number of months production will run, to the nearest tenth of a month, DM.

$$DM = \frac{DD}{30.4375}$$

Step 4: Calculate the natural learning slope, B.

$$B = \frac{\log(S\%/100)}{\log(2)}$$

Step 5: Calculate the labor hours for each unit produced, H_q, from q = 1 to Q.

$$H_q = H_1 q^B$$

Step 6: Calculate the total labor hours for all units to be produced, HT:

$$HT = \sum_{q=1}^{Q} H_q$$

Step 7: Using the "Hungry Algorithm," allocate hours for production of each unit.[2] From the result determine the begin and completion times of each unit, t_q and T_q. This is best illustrated by example.

Note: there can be concurrency of production, with more than one unit in production at the same time. The Hungry Algorithm establishes the necessary concurrency automatically. The less total time allowed by the model user for production, the more the concurrency. For many products, a high level of concurrency requires multiple production lines or stations, or multiple work shifts.

Example: To fully illustrate the method, the following problem is solved in detail. A company wants to produce 10 units of a product in the period TS = January 5, 2008 to TF = June 14, 2008. First unit hours are H_1 = 1,150, and the learning slope is S% = 87%. Allocate the hours by month using the Hungry Algorithm and determine a start and a finish time for each unit.

Step 1: SEER-H calculates the first unit hours (1,150)

Step 2: Calculate the production overall duration in calendar days

$$DD = TF - TS$$

[2] See worked example below.

The dates given are separated by 161 days. DD = 161. (See the Appendix.)

Step 3: Calculate the number of months production will run, to the nearest tenth of a month.

$$DM = DD/30.4375 = 161/30.4375 = 5.3 \text{ months}$$

Step 4: Calculate the natural learning slope.

$$B = \frac{\log(S\%/100)}{\log(2)} = \frac{\log(87/100)}{\log(2)} = -0.2$$

Step 5: Calculate the labor hours for each unit produced.

$$H_q = H_1 q^B$$

Unit	Hours
1	1150
2	1001
3	923
4	872
5	833
6	804
7	779
8	759
9	741
10	726

Step 6: Calculate the total labor hours for all units to be produced.

$$HT = \sum_{q=1}^{Q} H_q$$

The summation of the hours in the above table is 8,588.

Hungry Algorithm		**Months & Month Fractions**						
		January	**February**	**March**	**April**	**May**	**June**	
Unit	**Hours**	**0.8407**	**1**	**1**	**1**	**1**	**0.46**	Totals
1	1150	1150						1150
2	1001	212	789					1001
3	923		831	92				923
4	872			872				872
5	833			656	177			833
6	804				804			804
7	779				639	140		779
8	759					759		759
9	741					721	20	741
10	726						726	726
Totals	8588	1362	1620	1620	1620	1620	746	8588

Step 7: Using the "Hungry Algorithm," allocate hours for production of each unit. From the result determine the begin and completion times of each unit, t_q and T_q. This is best illustrated by example (see table above, discussion below).

The constant burn rate is 8,588 hours / 5.3 months = 1,620 labor hours per month. We build the table shown above, appropriately accounting for fractional months.

The working of the Hungry Algorithm is fairly obvious. Starting with unit #1 in January, we enter into each cell as many hours as possible, without violating either the row or the column constraints. For example, in the unit #1 January cell, we can enter the total hours required to do unit #1. But 212 hours are still available for unit #2, so those are also entered. But unit #2 requires 789 additional hours, so we put those in February. Continuing in this manner, we spread all of the required hours across the 5.3 month period of performance.

Although the Hungry Algorithm defines the <u>month</u> each unit begins and ends, it does not define the <u>day</u> in the month. Therefore there are options as to how to do this. What we decided to do is assume that each unit, in numerical order, occupies the full labor force, until it is completed. Then the labor force moves on to the next unit. This approach is computationally simple and reasonably realistic.

Unit #	**Hours**
3	92
4	872
5	656
Total	1,620

For example, in March we have the situation shown at left. March has 31 days, We assume that unit #3, which started in February, requires 92/1620 = 5.7% or 1.8 of those 31 days. It is therefore completed on March 2.

Unit #4 requires 872/1620 = 53.8% of those days, with a cumulative of 59.5%. It starts on the same day that unit #2 is completed, and finishes on March 19. Unit #5 starts the same day that unit #4 finishes, and is completed in April.[3]

Linear Ramp-Up Mode

In the linear ramp-up mode of scheduling, the production rate increases linearly from zero at t = 0 to some R_{max} at t = D. D is the user specified duration for lot production in months. The user also specifies Q, the total quantity to be produced in the period D. We have *R* = *at*, a linear function of time, t, with the constant *a* to be determined.

$$Q = \int_0^D R dt = \int_0^D at dt = \frac{aD^2}{2}$$

Hence, $a = 2Q/D^2$ and $R = 2Qt/D^2$. When t = D, $R = R_{max} = 2Q/D$, so if (say) Q = 10 units and D = 10 months, $R_{max} = 2$ units/month.

At any intermediate time t, the quantity produced is q:

$$q = \int_0^t \frac{2Qt}{D^2} dt = \frac{Qt^2}{D^2}$$

The cumulative time to produce the qth unit is:

$$t_q = D\sqrt{\frac{q}{Q}}$$

Let H_1 = labor hours to build unit #1, as estimated by SEER-H. Hours to build any subsequent unit q are:

$$H_q = H_1 q^B$$

where

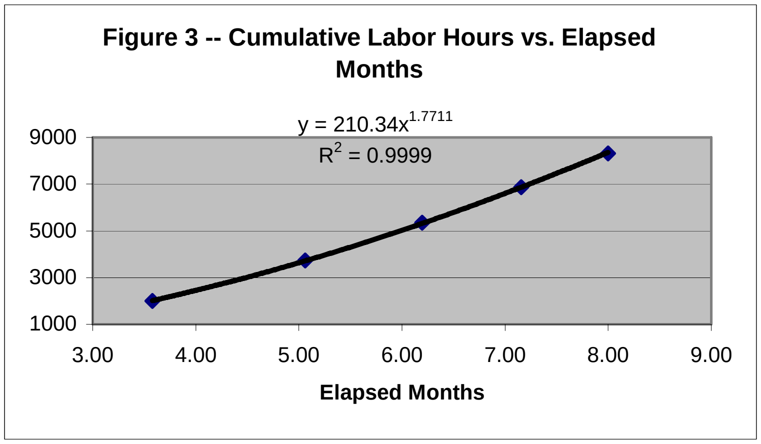
$$B = \frac{\log(S\%/100)}{\log(2)}$$

[3] We decided as a design ground rule not to take into account weekends and holidays. To do this would complicate the model and would require additional features that permit users to set the days when work is not done in their organizations. This means that our results are approximate dates, but we deemed this to be a reasonable limitation for a parametric cost model, where timing of cost flow is the main objective, as opposed to a detailed production scheduling model..

and S% = learning slope in percent. We can pair up the cumulative hours to build units 1 through n, and also the cumulative time required to build them. At this point, an example problem is useful to illustrate the method.

Unit, n	Months	Hours	Cum Hrs
1	3.58	2000	2000
2	5.06	1741	3741
3	6.20	1605	5347
4	7.16	1516	6862
5	8.00	1450	8312

Example: A company wants to build five units of their GIZMO in low rate initial production. The period of performance is eight months. The first unit hours are estimated at 2,000. They want to use the ramp-up scheduling mode. Work out the production schedule and the labor hours distribution by month (procedures for determining start and finish dates have been demonstrated in the previous two examples and will not be repeated here.)



From previous examples we know that B = -0.2 for an 87% slope, a slope we will again assume. Also, D = 8 months, Q = 5 units, and H_1 = 2,000 hours. A table nearby indicates labor hours and cumulative labor hours by unit. These were calculated using the unit theory equation $H_q = H_1q^B$.

The table also shows the elapsed months to complete each unit using the equation t_q = D*sqrt(q/Q), starting at t = 0, and also includes the cumulative hours. A cross plot of the cumulative hours and the elapsed months is shown above.

The cross plot can easily be regressed to find a relationship between elapsed months and cumulative hours. Because the underlying mathematics is all power law, the regression should seek a power law fit. The regression equation for this particular data is shown in the plot. The regression is automated in SEER-H v7.1.

Month	Cum Hours	Cum Norm Hrs
1	210	209
2	718	714
3	1472	1463
4	2450	2435
5	3638	3616
6	5025	4994
7	6602	6561
8	8363	8312

At left is a table of labor hours allocated by month using this equation. To correct small curve fit errors, the data has been normalized to have the correct total (8,312 hours).

Note that with the 87% learning rate we have used, the decrease in labor usage due to learning is more than offset by the increase in production rate, so that the build up in labor required per month is fairly steep. The labor build up is of course less severe if the production makespan is increased beyond eight months.

Note also the rather long period of time to build the first unit (3.58 months). This is much longer than the time required for any other unit. The 5th unit requires only 0.84 months. This is characteristic of ramp schedules. The initial rate is zero, and the labor force is initially small, so the initial unit takes a while to get organized.

Using another equation from above, the maximum production rate reached at the end of the ramp is:

$$R_{max} = 2Q/D = (2)(5)/(8) = 1.25 \text{ units/month}$$

The ramp approach will assume minimal concurrency, that is, unit #2 will start when unit #1 is finished, unit #3 will start when unit #2 is finished, etc. The rationale is that the rapid build up and possible limitations of tooling, floor space, etc. will often make any significant concurrency infeasible in linear ramp up production.

Revisions Needed for Cumulative Average Theory

The analysis for the cumulative average theory is the same as in the previous sections with the exception that the equation $H_q = H_1q^B$ is replaced by the equation $H_q = H_1[q^{1+B} – (q-1)^{1+B}]$.

Treatment of Production Material Costs

No changes are contemplated in the way SEER-H estimates production material cost, but we do have in mind two changes in the way material costs are handled. One change is to allow users to apply learning theory to material costs, a common industry practice. Another is to allow users more realistic ways to spread material costs.

Learning slopes for material are usually on the order of 95%, and the savings realized typically does not represent nearly as large a part of the project funding as is the case for labor hours learning. For that reason, we will use only one theory of learning for material. For reasons of mathematical convenience, we will use the cumulative average theory only. If the user universally elects to use the unit theory, this introduces some error, but it is small.

Material for a work element will be considered as (possibly) having two parts. One part is bought more or less concurrently as production proceeds, on essentially a “just in time” basis. The other part is long lead-time material that must be ordered well in advance of production, perhaps as early as the Detail Design phase of development. These two parts will be designated, respectively, concurrent material and long-lead material.

Only concurrent material will have learning applied. Long-lead material is assumed not to benefit from learning savings.

By default, all material costs calculated by SEER-H for a work element are assumed to be concurrent with production. Long-lead material is assumed to be zero in cost unless the user makes specific provision it. To make provision for it, the user must specify, of the total units to be produced, how many units will require long lead material. For example, if 10 units are to be produced, and the user specifies that three will require long lead procurement, then SEER-H will calculate the material cost for the total build of 10 units, then will take 3/10th of that amount, and designate it as long-lead material. By default, that buy will be placed in a month midway between Preliminary Design Review (PDR) and Critical Design Review (CDR).[4] However, the user can change the timing of this advance buy.

After adjustment for learning, assuming that the user has specified that material learning takes place, the remaining material cost will be allocated to concurrent material. Assume that the material cost as calculated by SEER-H is M, and that the amount allocated to long-lead is m. Then the total concurrent material for production quantity Q is C_Q = M-m.

According to the cumulative average theory, the cumulative material cost at any intermediate quantity q is given by:

$$C_q = C_Q \left(\frac{q}{Q} \right)^{1+B}$$

As usual, B represents the natural slope of the user assigned learning curve.

The timing rule for cost flow of concurrent material is as follows: For each work element built, its concurrent material cost is charged in the same month the unit starts production.

Loss of Learning

If one production lot follows another, all prior learning may apply to the successor lot, none of it may apply, or part of it may apply. If all of it applies, the learning calculations continue from one lot to the next as though they were all one lot, although in the second lot the learning slope could be different. If none of it applies, the successor lot learning simply starts over with unit one.

By default, SEER-H assumes that each lot begins with no prior learning. This assumption will hold unless the user specifies that there has been prior learning. A named predecessor lot is assumed to be the source of that learning. Once that is

[4] In addition to the changes described in this paper, SEER-H v7.1 will also estimate the points in time for each work element when Preliminary Design Review and Critical Design Review can occur.

done, an issue that immediately arises is whether all of the prior learning is to be credited to the successor lot, or only part of it. The user must make this choice.

Normally, if the successor lot follows closely in time behind the predecessor lot, or perhaps slightly overlaps it, and both lots are being built at the same site by the same people, it is reasonable for the user to declare a full transfer of learning from one lot to the next. Otherwise, it is likely that some learning has been lost, and that will increase the costs (and may modify the timing) of the successor lot. If the user declares that any learning at all has been lost, the user shall be presented with what is called an Anderlohr analysis, so named after the industrial engineer who invented it.

First, we examine the case where the user specifies that no learning is lost. In that case, all production units simply follow the learning curve already established by the predecessor lot. However, the user may change the slope in the successor lot, but may not switch from one learning theory to the other.[5] The user may, however, change from one production planning mode to another. For example, the predecessor lot may follow the ramp-up mode, while its successor lot follows the constant effort mode. And of course, each lot may have a different production quantity and even a different average labor rate.

If the user states that some of the learning is lost between the predecessor and the successor, then the Anderlohr process is invoked to determine just how large the loss is. The increase in cost of the successor is a strong function of the amount of the learning loss, and the loss can be considerable.

The Anderlohr method begins with an analysis of the learning content of the predecessor lot. It continues with analysis of how that affects the successor lot. We will give a worked example of how those analyses are done based on the assumption that the predecessor lot used the unit theory. That will be followed with an explanation of what is done differently if the predecessor lot used the cumulative average theory. (Recall that the user is required to use the same learning theory in both the predecessor and the successor lots.)

<u>Unit Theory Andelohr Example.</u> Suppose that the predecessor lot had production quantity Q = 20 units, first unit hours H_1 = 1000, and learning slope S% = 90%. Using the usual equation, we find that B = -0.152.[6] If we calculate the hours for the 20th (final) unit of the predecessor lot, we find:

$$H_{20} = 1000(20)^{-0.152} = 634$$

[5] Users who want to use a segmented learning curve can do so by declaring a new lot at the quantity where they want the slope to change.
[6] B = log(S$/100)/log(2)

We started with 1,000 hours for the first unit, and reduced that to 634 hours for the final unit. We define learning accomplished (LA) as the difference between these numbers:

$$LA = 1000 - 634 = 366 \text{ hours}$$

LA measures learning gained. To measure learning lost, Andelohr uses five loss factors, expressed as *subjectively* assigned percentages (here we arbitrarily assign percentages to each factor, to be used in the subsequent example):

- Personnel loss of learning – 14%
- Supervisory loss of learning – 20%
- Continuity loss of learning – 18%
- Methods loss of learning – 1%
- Tooling loss of learning – 1%

We average these subjectively assigned percentages to obtain 10.8%.[7]

Since LA = 366 hours and 10.8% of that has been deemed lost, the loss amount is (10.8%)(366) = 40 hours. Had no loss occurred, the hours for the next unit would have been:[8]

$$H_{21} = H_{20}(21/20)^{-0.152} = 634(21/20)^{-0.152} = 629.3 \text{ hours}$$

With no loss of learning, 629.3 hours would be the first unit hours for the successor lot. But due to the loss of learning, we add the lost 40 hours to this to obtain a starting position of 629.3 + 40 = 669.3 hours.

Somewhere back in the predecessor lot was a hypothetical "unit" that required 669.3 hours. This almost certainly is not an integer (whole) numbered unit. It is most likely a fractionally numbered unit with a unit number like 3.85, or maybe 4.12. In other words, it's some whole numbered unit, plus a fraction of the next unit.

[7] This is equivalent to assigning each of the five factors an equal weight. Anderlohr, in his original paper, suggested that it might be desirable to weight them differently if there was reason to do so. In SEER-H v7.1 we shall assume equal weights, as opposed to asking the user to assign possibly different weights for each percentage.

[8] Actually, "unit #21" is unit #1 of the successor lot. If one lot follows another with no loss of learning, in order to keep the math straight the units of the successor lot must be renumbered in the calculations, continuing the numbering sequence of the predecessor lot. We will refer to this practice as "pseudo numbering." As will be seen shortly, a slightly different form of pseudo numbering is required when there is loss of learning.

Actual	Pseudo	Unit Hours
1	NA	1000.00
2	NA	900.00
3	NA	846.21
4	NA	810.00
5	NA	782.99
6	NA	761.59
7	NA	743.95
8	NA	729.00
9	NA	716.06
10	NA	704.69
11	NA	694.55
12	NA	685.43
13	NA	677.14
14	NA	669.55
15	NA	662.57
16	NA	656.10
17	NA	650.08
18	NA	644.46
19	NA	639.18
20	NA	634.22
21	14.07114	669.04
22	15.07114	662.09
23	16.07114	655.66
24	17.07114	649.67
25	18.07114	644.07
26	19.07114	638.82
27	20.07114	633.88
28	21.07114	629.21
29	22.07114	624.79
30	23.07114	620.60

The reason why we want to find this unit number is because Anderlohr's method starts the learning in the successor lot all over again from that point. If the number is say, 3.85, the pseudo numbering in the successor lot would be 3.85, 4.85, 5.85, etc.

How do we find this unit number? Designate the unknown unit number as k. We know that:

$$H_k = H_1(k/1)^B$$

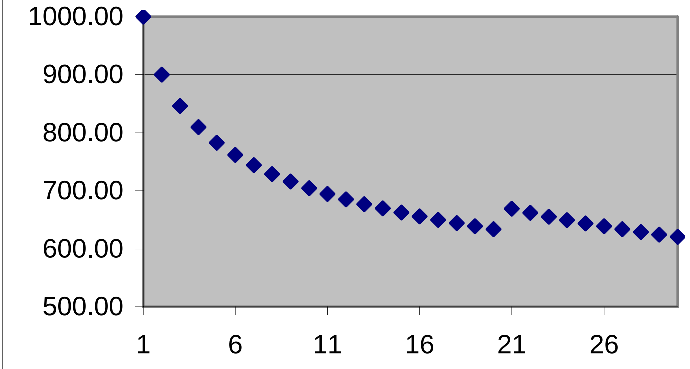
Solving this equation for k:

$$k = \left(\frac{H_k}{H_1}\right)^{1/B} = \left(\frac{669}{1000}\right)^{1/-0.152} = 14.07$$

Hence, whatever planning mode the user has applied to the successor lot, it is applied not to the original lot numbers 1, 2, 3, etc., but to the pseudo lot numbers 14.07, 15.07, 16.07, etc.

At left is a detailed, unit by unit calculation of this example. Note that for units 1 through 20, before the break, the actual unit numbers are used in the calculation. From Unit #21 on, after the break, the pseudo numbers are used. Figure 4 shows the effect of the break.

Figure 4 -- Example of Loss of Learning



<u>Cumulative Average Theory Anderlohr Example.</u> The cumulative average theory replaces $H_q = H_1q^B$ with $H_q = H_1[q^{1+b} - (q-1)^{1+B}]$. The calculation for k is messier. It requires an iterative solution of:

$$k = [\frac{H_k}{H_1} + (k-1)^{1+B}]^{1/1+B}$$

This is done by entering an initial guess for k on the right and solving for k on the left. That value is then entered on the right and a new solution is obtained. This continues until the answer is the same to a satisfactory precision. A good initial guess for k is the number of units in the predecessor lot divided by 2.

Appendix -- Rules for Converting Time Spans to Calendar Dates & Vice Versa

The SEER-H v7.1 scheduling system is not intended to be used for precise scheduling of work activities to include such features as weekends, holidays, multiple shifts, plant shutdown periods, irregular shifts, etc. When SEER-H estimates an activity duration of, say, 7.6 months, it means that the project is expected to last 7.6 *average* months, calendar time, without regard to peculiarities of the work days in a given month.

In the average solar year there are (very closely) 365.25 days, and exactly 12 months. The average month therefore has 30.4375 days, to a sufficently high level of precision. Accordingly, if SEER-H estimates 7.6 months, as an example, the number of calendar days will be (7.6)(30.4375) = 231, rounded to the nearest integer, using the usual arithmetic rounding convention. The end date corresponding to any stated begin date shall be counted off on the calendar, with each month given its usual number of days, with 28 days allowed for February. The first day of the count shall be the user stated start day.

Example: SEER-H estimates an activity as 7.6 months. The corresponding number of days is 231. The user has stated that January 15 is the start date. What is the end date?

We list here the number of days in each calendar month:

January -- 31
February -- 28 (leap year and end of century effects ignored)
March -- 31
April -- 30
May -- 31
June -- 30
July -- 31
August -- 31
September -- 30
October -- 31
November -- 30
December – 31

Month	Days	Cum Days
January	17	17
February	28	45
March	31	76
April	30	106
May	31	137
June	30	167
July	31	198
August	31	229
September	2	231

Of the 231 days duration, these are the numbers of days consumed by each calendar month, and the cumulative sum of these. The end date is therefore September 2, (at end of business).

Biography: Evin J. Stump

Evin Stump is a senior systems engineer for Galorath Incorporated. He has recently managed the Darwin project, aimed at developing an advanced cost model for estimating the effects on projects of “evolutionary acquisition.” Other recent and ongoing assignments include development of the Spyglass model for estimating costs and schedules of development and production of electro-optical sensors used in space, and updating the SEER-IC model for estimating costs and schedules of development and production of integrated circuits.

Mr. Stump has over 54 years of experience in the aerospace industry having worked as a test engineer, design engineer, systems engineer, project engineer, cost engineer, engineering manager, and independent consultant. Major projects to which he has made a significant contribution include Mercury, Gemini, Apollo, Tacit Rainbow, Brilliant Eyes, C-17, and RSA IIA. Mr. Stump is a former president of the Southern California chapter of the International Society of Parametric Analysts (ISPA), and also of the Southern California chapter of the Society of Cost Estimating and Analysis (SCEA).

Mr. Stump holds the BS in Engineering form Loyola University of Los Angeles, and the MS in Operations Research from the University of Texas at Austin. He has also earned the professional designation in government contract management from UCLA/NCMA.

Biography: Alexandra Minevich

Alexandra Minevich is a Systems Engineer/Cost Analyst for Galorath Incorporated. She has recently assisted in development of the labor effort and duration allocation schemes for SEER-SEM Client for MS Project. She also participated in the development process of SEER-H Client for MS Project, and has developed and tested a prototype of the algorithms which are the subject of this paper.

Ms. Minevich holds the BS in Computer Science from the University of California at Los Angeles. As a student at UCLA, Ms. Minevich participated in a project to develop a health alert and monitoring system. The project involved establishing system requirements, subsystem specifications, organizational arrangements, work breakdown structures, marketing approach, and risk analysis.