Session Number 3553

Application of Object-Centered Scaffolding to Introductory MatLab

Jon Sticklen, Marilyn Amey,
Taner Eskil, Timothy Hinds, Mark Urban-Lurain

Michigan State University

Introduction

In this report, an extension to the standard sense of scaffolding as used in the learning literature is proposed. *Object scaffolding* is proposed as a pedagogical technique in which current student learning is anchored by a conceptual map resultant from previous learning. This report focuses specifically on the use of object scaffolding for freshman level instruction in MatLab.

MatLab is rapidly gaining acceptance if not defacto default status as the computational tool of choice in many engineering colleges. In the majority of universities in which MatLab plays the role of preferred computational tool in the undergraduate engineering curriculum, MatLab is typically introduced in freshman or sophomore courses. There are a number of choices for textbooks which introduce MatLab. But all can be characterized as having a "steep learning curve."

Current experience in directly applying the proposed expanded version of scaffolding to introductory MatLab is described. The conceptual starting point is the observation that freshman engineering students do not arrive in introductory MatLab courses with significant experience with array/matrix formulation of problem solutions, nor with computational operations such as matrix multiplication. Given this observation, rapid introduction of MatLab starting with array/matrix operations from the outset is ill advised.

However, beginning engineering students do have background in scalar operations. Starting with "MatLab as Scientific Calculator" and focusing on scalar expressions and becoming familiar with the MatLab environment, then moving systematically to "Vector Operations," and then finally to "Array and Matrix Operations" enables student linkages from the outset with past learning. Once student conceptual organization for MatLab scalar operations solidifies, understanding of vector operations is a small step, with the most significant new concept being one-dimensional indexing into a vector. From there, two dimensional indexing also becomes a relatively small step, and this opens the way for introduction of the full range of array and matrix operations that is linked to past student learning.

Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition

In this report, a description of a MatLab learning progression Scalars => Vectors => Arrays/ Matrices will be given, and the expected impact of introducing MatLab by following this progression will be described. We intend to test this expected impact by comparing student performance on MatLab programing problems in spring semester, 2003, to performance on like difficulty problems in spring semester, 2004.

Background – Concept of Scaffolding

As a concept, *scaffolding* is known in the engineering education community. But, as many times true across diverse technical areas, *scaffolding* means different concepts depending on the commentator and the context for use.

When discussed in a pedagogical context, scaffolding is usually linked to the learning theories of Vygotsky. (Vygotsky 1962; Vygotsky 1978) Vygotsky's viewpoints are rooted in the belief that a learner's cognitive development is enabled by interaction with more capable members of the same culture – usually teachers or other students. As a student learns, she from time to time encounters difficulty, and at such times appropriate assistance will help the learner break through. The assistance can be of almost any form including the posing of Socratic-type questions to lead the learner, the revealing of unknown (to the learner) relevant facts that the leaner needs to know, the description of apt analogies to help the learner, … The act of providing this assistance to the learner is termed *scaffolding*.

In Vygotsky's view, many times a learner will be on the verge of understanding a concept or solving a problem, and will need just a "nudge" from a more capable member of the same culture. That point of need for specific and pointed help is known as the state of *proximal development*. In the context of a formal class, the role of the teacher is to either supply the needed assistance, or to facilitate the learner getting the assistance from someone else. It is important to understand that in Vygotsky's theory, scaffolding is undertaken only when a learner is in need of it, and that, perhaps more important to the learning theory, that scaffolding is stopped as soon as possible, that is, as soon as the learner can solve exemplars of the target problem on her own.

There is a somewhat subtle point in the above paragraph that often leads to confusion of what scaffolding (in the Vygotsky sense) means. Following Vygotsky, scaffolding is not an informational or procedural "fact" of stand alone character. Rather scaffolding is the *act* of providing such facts that are currently needed by a learner but unknown to that learner. This follows from an implication of the importance in the Vygotsky theory that scaffolding be removed once the learner no longer requires it. A fact like "the MatLab plot function has optional arguments for plot line color" is not something that can be withdrawn from a learner. It is the act of offering the "nudge" that can be stopped once the learner if fully capable of independent work on problems of the target variety.

The goal of scaffolding in the Vygotsky sense is that the learner gradually attain mastery in solving problems of the target variety. Moreover, such mastery should be demonstrable by the learner actually solving problems with no help. In essence, the theory of proximal development is a road map for how to achieve this goal.

In the computer science community, *scaffolding* in the sense described above has most often surfaced in the human factors subcommunity or in the intelligent tutors subcommunity. (Chalk 2001; Heffernan 1998; Jackson, Krajcik et al. 1998; Quintana, Krajcik et al. 2002) These subcommunities have taken the sense of scaffolding more or less intact from the original meaning delineated by Vygotsky.

In contrast, the term *scaffolding* has been used in the software engineering subcommunity in more direct analogy to the scaffolding that a construction crew uses to erect a new building: as an actual object used to build some final structure. (Sellink and Verhoef 2000) In software engineering scaffolding has most often been used to mean software put in place whose sole purpose is to facilitate debugging. The important high level distinction between *scaffolding* as used in learning theory and *scaffolding* as used in software engineering is that in learning theory usage scaffolding refers to *process*, while in software engineering scaffolding refers to *object*.

Major Thesis of this Work

The major thesis of work reported here is that broadening the pedagogical usage of *scaffolding* to include the object sense (as used in software engineering) can shed light on learning difficulties faced by many engineering students when they are first introduced to the MatLab computational environment, and by extension to other modern computational tools for technical problem solving.

If an educational scaffold is viewed as a conceptual map that a student must have to enable further student progress towards computational tool mastery, then a high level shift in instructor viewpoint is prompted. Scaffolding in the process sense is still an available pedagogical tool to facilitate targeted student learning for specific tool-use skills. But in the large grain perspective of becoming familiar with and competent in the use of an entire computational environment, scaffolding in the object sense lays the ground work for *systematic construction of a piece wise instructional presentation* of the computational environment in such a way that student learning is enhanced.

In a nutshell, the idea is that a current conceptual map is an enabling starting point for learner mastery of new items, and further, that the current conceptual map provides *anchors* for the learner as new topics are introduced. The current conceptual map then becomes the scaffold (in the object sense) for anchoring new student understanding.

MatLab and Current Textbooks Supporting MatLab

MatLab is very complete computational environment supporting technical problem solving in general, and via a number of "toolboxes" supporting problem solving in target domain areas such as embedded controller applications and image processing. MatLab was originally developed to directly support high level array and matrix operations; the name *MatLab* derives from *Matrix Laboratory.*

As is true for all modern computing environments supporting technical problem solving (MatLab, Mathematica, Macsyma, MathCad, …) novice learners face a step learning curve in going from zero capability to minimal capability. In addition to such matters as becoming familiar with window layouts and help facility use, learning MatLab entails becoming facile in problem formulation in array/matrix form whenever possible.

There are *many* textbooks in the academic marketplace currently supporting MatLab. A common assumption underpinning the presentation in these textbooks is that students have a working grasp of array and matrix operations. A good exemplar is Introduction to MatLab 6 for Engineers, by William Palm. (Palm 2001), an often used textbook for beginning MatLab instruction. Coverage in the Palm book of the basics in MatLab is both thorough and well presented *provided the learner is already aware of and somewhat proficient in the concepts of arrays and matrices*. The Palm book is thus very appropriate for the professional engineer who is trying to "pick up" MatLab, or for the graduate student in a technical area.

However, for freshman level students, the assumption of base-line familiarity with array/matrix operations is not typically met. Freshman students faced with learning both how to manage a complex computer environment *and* learn from scratch concepts associated with array representations and operations most often face knowledge overload sufficient to mute crisp learning that would lead to proficiency in MatLab use.

Highlighting the difficulty faced by freshman students, in Chapter 2 of the Palm book, arrays are introduced as the first topic. Vectors are introduced by Palm in two pages and as a special case of arrays. In rapid succession then first array operations are introduced, then matrix operations are introduced. For a freshman student, who likely in his high school experience has used a graphing calculator as a tool, and performed in general only scalar operations, the very rapid pace of the Palm book leaves little time for concept understanding and assimilation, and often results in a high degree of learner frustration.

Applying the "Object Version" of Scaffolding for MatLab Instruction

To apply the object version of scaffolding, the first task is to decompose a broad learning objective into a number of identifiable "stepping stones." To illustrate, let us set as the broad learning goal the development of student facility in formulating and solving problems in an array/

matrix format. In addition a freshman engineering course focused on MatLab use would extend to working with polynomials in MatLab, to 2-D and 3-D graphing, to optimization problem solving, and so on. But for purposes of clarity of exposition here, we will limit the goal to base line facility in using arrays and matrices.

Noting again that freshman students typically have at best a theory-only comprehension of arrays, but that they do typically have solid working understanding of scalar operations, a first "stepping stone" would naturally lie in "scalar computations using MatLab." Students know very well how to use a calculator to solve such relationships as $D = \frac{2s_0^2 \sin(\phi)\cos(\phi)}{g}$ for the horizontal distance D traveled by a projectile fired with initial speed s_0 at an angle ϕ from the horizontal, and given the acceleration due to gravity g. They do *not* know coming in the scalar operator syntax of MatLab, nor with any depth the concept of stored variables, nor a number of other facets required to perform scalar calculations using MatLab. But by isolating to one educational target (scalar operations) and setting it up as one substantial unit of study, student prior knowledge of scalar computations becomes the conceptual anchor for students to leverage in assimilating new knowledge.

After the unit (stepping stone) on scalar computations in MatLab is completed, including substantial exercise by learners on practice problems, vector operations is a relatively small step conceptually. Vectors in MatLab are linear, ordered sets of scalars. The scalar operations in MatLab carry over directly to the vector analogues. For example, if A and B are two MatLab vectors, then the vector C = A + B is found by adding the corresponding elements in A and B. Thus C(2) (the second element of the C vector) is found by C(2) = A(2) + B(2), and similarly for all other elements. In MatLab such operations are termed cell-by-cell operations. The pedagogically important point is that following a unit on scalar operations with a well circumscribed unit on vector operations (a) builds cleanly on student understanding of scalar computations while (b) introducing broadly one new type of computation: cell-by-cell computations.

Table 1: Values for Total Horizontal Distance (initial speed = 50 feet per second)

ϕ *(degrees)*	*D (feet)*
0	0.0000
10	26.7203
20	50.2178
30	67.6582
40	76.9381
50	76.9381
60	67.6582
70	50.2178
80	26.7203
90	0.0000

A sample of the type of problem students are given in the unit on MatLab vector computations is to revisit the equation first given as a scalar relationship for the horizontal distance traveled by a projectile. For a fixed value of the initial projectile speed and a known value for g, students can

now produce a table (Table 1 above) of values for D that result from a range of values for ϕ. ϕ is represented as a vector of values, then the same mathematical relationship is used but implemented now by using vector operators instead of the scalar operators used before. The result is a vector of corresponding values of D. From the table results students can then approximately determine the firing angle that will *optimize* the horizontal distance the projectile covers.

As a learner incorporates knowledge of vector operations into her conceptual map of MatLab use, natural generalizations of the scalar operators are made to corresponding vector operators. The top level new knowledge to be learned is the concept of cell by cell operations on vectors. Once that is mastered the entire conceptual map for MatLab use of vector computations can be understood as a generalization of the conceptual map for scalar computations.

So far, the learner has evolved two versions of a conceptual map for MatLab use from a starting conceptual map for the use of a calculator to perform scalar computations. The first evolved conceptual map, capturing student understanding of using MatLab for scalar operations depended strongly on the starting conceptual map. The second evolved conceptual map capturing student understanding of using MatLab for vector operations depended strongly on the first evolved conceptual map.

In like manner once student understanding of MatLab use for vector operations is in place, the next stepping stone that should be introduced is array and vector computations in MatLab. As in the case for student learning of MatLab vector computations based on generalizing what had previously been learned on MatLab scalar computations, student learning of MatLab array/matrix computations is presented as high level generalization of vector operations. The progression in student learning is depicted in Figure 1 as "layering of an onion."

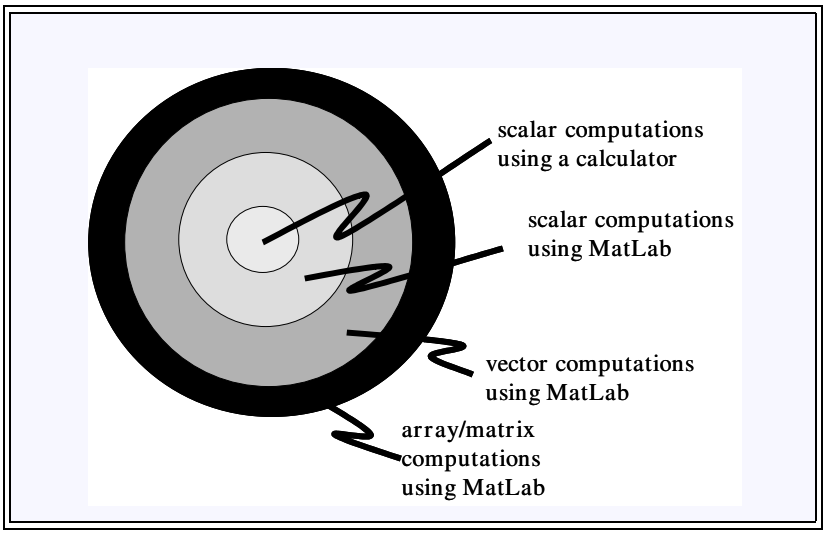

Figure 1: Layers of an onion view of evolving conceptual understanding of MatLab

It is important to emphasize that the view of evolving student learning of MatLab in Figure 1 is predicated on well circumscribed "units" on (a) scalar computations in MatLab, (b) vector operations in MatLab, and finally (c) array/matrix computations in MatLab. Without the circumscription of each stepping stone in the sequence, or to put it more operationally in terms of course layout, without each unit having a stand-alone character, systematic student generalization from what they know to what you intend for them to learn is impeded.

Conclusion

The proposal put forward in this paper addresses the need to ameliorate the steep learning curve of MatLab as perceived by freshman engineering students. The proposal can be boiled down to

1. decompose the initial major goal for MatLab learning (proficiency in scalar, vector, and array/matrix computations) into three self contained "units"
2. in a first unit aimed at proficiency in scalar computations, make connections to student understanding of scalar computations using a calculator,
3. in a second unit aimed at proficiency in vector computations, portray vector computations as a generalization of scalar MatLab computations,
4. in a third unit aimed at proficiency in array/matrix computations, portray array/matrix computations as a generalization of vector MatLab computations

This "recipe" for MatLab instructional design may appear to be straight forward, and in some ways trivial. In fact, it is not self evident as indicated by the fact that the most currently used MatLab textbooks blur the three "units" into one introductory chapter.

This general layout for freshman instruction in MatLab has been used for two terms in CSE 131 at Michigan State University: Spring Semester 2003, and Fall Semester 2003. CSE 131 serves approximately 400 students per term, and acts as a gateway course to most of the engineering majors at Michigan State. To date, anecdotal evidence only is available to judge the efficacy of the proposal. However this anecdotal evidence supports the view that freshman level instruction in MatLab will profit if this very general view is followed. We plan to analyze student examination data from prior semesters to student examination data for Spring, 2004, to obtain more tangible comparisons.

This report is a response to the student-perceived steep learning curve of freshman instruction in MatLab. But the idea proposed to extend *scaffolding to support learning* to include both *process scaffolding* (the "traditional sense" of scaffolding in learning theory) and *object scaffolding* (as proposed in this report) may find application beyond MatLab instruction. This broadening for what the community calls scaffolding is prompted by one key observation.

Both process scaffolding and object scaffolding can be viewed as folding in the concept of *proximal development*, as described by Vygotsky. When a learner is in a state of proximal development, she is learning something new, something that is not understood, but close to being understood. In a process scaffolding setting the small bit of missing information can be filled in by "nudges" from an instructor. It is the process of nudging that is termed scaffolding. In an object scaffolding setting, the small distance between the "new material" what the student already knows is bridged by a generalization of what the student has already assimilated. The point is that in both process and object scaffolding, a small conceptual "distance" exists between what the student knows and what we hope the student will learn: the zone of proximal development is a common concept to both variants of scaffolding.

Process scaffolding is most appropriate at the small grain. That is process scaffolding is most applicable when a student is struggling with one relatively narrow facet of target learning. On the other hand, object scaffolding is aimed much more at the top level layout of instruction. Process scaffolding and object scaffolding can be viewed as complimentary learning enablers.

There is one strong sense in which process scaffolding and object scaffolding differ. In a context of process scaffolding, the concept of *fading* is very important. That is, once the learner is capable of problem solving without the "nudge" of process scaffolding, the scaffolding should be removed. But in object scaffolding, there is no fading or withdrawal of the scaffold. Indeed, to the extent that object scaffolding is effective, there is a constant evolution of the learner's conceptual map that retains conceptual map knowledge from earlier stages of learning while integrating new understanding.

Although the view that scaffolding should be broadened to include both process and object scaffolding is particularly well suited to act as pedagogical underpinning to support instructional design for freshman instruction in MatLab, the concept of the two complimentary types of scaffolding to support learning may find wider applicability.

Bibliography

Chalk, P. (2001). Scaffolding learning in virtual environments. Annual conference on Innovation and technology in computer science education, Candelabra, UK.

Heffernan, N. T. (1998). Intelligent tutoring systems have forgotten the tutor: adding a cognitive model of human tutors. Conference on Human Factors in Computing Systems, Los Angles, CA, USA.

Jackson, S. L., J. Krajcik, et al. (1998). The design of guided learner-adaptable scaffolding in interactive learning environments. SIGCHI conference on Human factors in computing systems.

Palm, W. (2001). Introduction to MatLab 6 for Engineers, McGraw Hill Higher Education.

Quintana, C., J. Krajcik, et al. (2002). A Case Study to Distill Structural Scaffolding Guidelines for Scaffolded Software Environments. SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves, Minneapolis, MN, USA.

Sellink, M. P. A. and C. Verhoef (2000). Scaffolding for Software Renovation. Fourth European Conference on Software Maintenance and Reengineering.

Vygotsky, L. S. (1962). Thought and Language. Cambridge, MA, MIT Press.

Vygotsky, L. S. (1978). Mind in Society. Cambridge, MA, Harvard University Press.

Wertsch, J. V. (1985). Cultural, Communication, and Cognition: Vygotskian Perspectives. Cambridge, UK, Cambridge University Press.

Author Biographies

Jon Sticklen is an Associate Professor in the Department of Computer Science and Engineering at Michigan State University. He has a strong research record in knowledge-based systems. His main contributions have been in the theory and application of task specific approaches and in model-based reasoning. Dr. Sticklen has led the effort to rejuvenate the MSU College of Engineering freshman gateway course in computational tools.

Marilyn Amey is Associate Professor in the Department of Educational Administration and Chair of the Higher, Adult, and Lifelong Education Program at Michigan State University. She was part of a research team studying best practices in Science, Math, Engineering and Technology Undergraduate Reform for SRI and NSF, and policy evaluator for an NSF Rural Systemic Reform project on math and science curriculum reform in the Navajo Nation.

Taner Eskil is a Ph.D. candidate in the Department of Computer Science and Engineering at Michigan State University. Mr. Eskil holds a M.Sc. in Mechanical Engineering and will soon complete his Ph.D. research in the area of internet agent support for electronic commerce. Mr. Eskil has been instrumental in developments in the College of Engineering freshman gateway course in computational tools.

Timothy Hinds is an academic specialist in the MSU Department of Mechanical Engineering. He teaches undergraduate courses in machine design and statics as well as advises senior engineering student teams working on industrially sponsored capstone design projects. He also teaches a senior-level undergraduate international design project course and has taught graduate-level courses in innovation and technology management.

Mark Urban-Lurain is Director of Instructional Technology Research and Development in the Division of Science and Mathematics Education at Michigan State University. He is responsible for providing vision, direction, planning and implementation for using technology mathematics and science education and developed several introductory computer science courses for non-computer science students serving 2000 students per semester.