ARMA/USRMS 06- 1012

Keynote Lecture: SEVEN LESSONS OF GEOMECHANICS SOFTWARE DEVELOPMENT

Curran, J.H.
Lassonde Institute, University of Toronto, Toronto, Ontario, Canada
and Rocscience Inc., Toronto, Ontario, Canada

Hammah, R.E.
Rocscience Inc., Toronto, Ontario, Canada

ABSTRACT: The paper outlines 7 key lessons learned from developing software for geomechanics practice. It describes the gap between the state of geomechanics research and real-world practice of the discipline. The paper argues that the development of software is essential to narrowing this gap, and outlines reasons why such development should receive greater recognition and funding. The reasons for this position include the fact that software is a much more active knowledge medium than written knowledge, and software development constitutes articulation of theory, an important component of science. The paper asserts that software development generates new knowledge and insights in two ways: during algorithm development and during application to problems. It illustrates different lessons through three examples.

1. INTRODUCTION

One of the meanings of the word lesson is "an experience, example, or observation that imparts beneficial new knowledge or wisdom. The knowledge or wisdom so acquired." [1]. Our involvement in academic research and the development of software for geomechanics purposes has offered many important lessons we would like to share. This paper will outline seven of the most important lessons and perspectives gained from over 25 years of such work.

It is the view of the authors that there is currently a considerable gap between the state of the art of geomechanics research and the state of professional practice. A survey of research findings published in academic journals and conference proceedings, accompanied by a comparison with current geomechanics practices quickly establishes this state of affairs. Alternatively, one can study the history of geomechanics techniques and knowledge routinely applied today. It is quickly discovered that whereas several of the techniques used in geomechanics practice today originated decades ago, research knowledge exists today that could substantially improve design practices if adopted.

Through the seven lessons to be outlined, the paper will argue that the development of active computational algorithms and software can help narrow the gap between geomechanics research state of the art and professional practice. (The term active computational algorithm will be defined in the next section.) The paper will also attempt to persuade readers that software development is as important a contributor to 'new' knowledge and insights as any other component of scientific research.

The authors believe that unquestioning adherence to traditional views of academic research constitutes a major reason for the gap. Grant applications for geomechanics research projects, which primarily involve software and/or computational algorithm development, are seldom adjudged to be 'original' or 'new' enough. The same attitudes persist in the assessment of doctoral theses on geomechanics software development. With the help of Lesson #2 especially, the authors hope to convince funding agencies and the academic establishment of the

many scientific merits of software development, attributes which call for a re-evaluation of how this important enterprise is judged.

2. A FEW DEFINITIONS

To avoid ambiguity, we will define a few key terms and the sense in which they are used in this paper. We define software as the complete suite of procedures, such as graphical interface, calculation routines, etc., which constitute a stand-alone computer program. The term computational algorithm refers to a step-by-step procedure for solving a numerical problem based on some scientific or engineering principle(s) developed in the form of pseudo-code, flow chart, or concept map. Whenever we refer to an active computational algorithm, we mean an algorithm that has been implemented in detail in a computer programming language such as C++ or FORTRAN, and has been well tested and debugged so that it is readily incorporated into a program.

We refer to engineering specialists who practice geomechanics by applying the knowledge they have acquired to the solution of real-world problems, and by translating ideas into reality as practitioners. They earn their living by solving the problems of clients as efficiently as they can. Under the banner of academics, we group specialists whose primary activity is to systematically investigate issues in order to increase knowledge, or use of this knowledge to create new applications. A primary characteristic of academic work is originality.

3. LESSONS OF SOFTWARE DEVELOPMENT

The lessons learned from the development of software from geomechanics research principles are many. However, we have attempted to identify seven of the most important. They are as follow:

- Lesson #1: Software is an active medium of knowledge; consider it as such.
- Lesson #2: Software development is integral to science; it is hard, creative work that generates new knowledge.
- Lesson #3: Parsimony is invaluable; avoid complexity, embrace clarity.
- Lesson #4: Interface design must not be an afterthought; it is fundamental to practical engineering.
- Lesson #5: Users have practical constraints; keep things as simple as possible.
- Lesson #6: Uncertainty is king; make room for it.
- Lesson #7: Legacy codes may be outdated; but think carefully before rewriting.

3.1. Lesson #1: Software is an active medium of knowledge; consider it as such.

An inadequate understanding of the nature of software development is a major problem in geomechanics. Too many specialists view software only as a product, a commercial product. But this view is inaccurate [2].

Software is primarily knowledge that has been encoded into a computer program. The most fundamental attribute of this form of knowledge is that it is active [3]. Because software can be executed, it is much more readily applied to problem-solving than written knowledge. Knowledge in written form, on the other hand, is inert or passive.

As will be detailed under the next lesson, the transformation of knowledge into working, robust computational algorithms is the hardest component of geomechanics software development. It requires extra-theoretical work, and extensive validation and verification effort. It is also a pursuit that yields substantial scientific rewards, including the creation of new knowledge. Because software is an active knowledge medium, it allows specialists to obtain deeper and better understanding of parameter interrelationships and interdependencies in ways written knowledge cannot. It makes it easier to perform thought experiments in order to answer a broad variety of questions.

3.2. Lesson #2: Software development is integral to science; it is hard, creative work that generates new knowledge.

The development of computational algorithms and software constitutes a necessary and important component of scientific research. In the seminal book, “The Structure of Scientific Revolutions,” [4] Thomas Kuhn, the renowned philosopher of

science, describes an activity he calls articulation of theory. This is the empirical work researchers do to resolve residual ambiguities in a theory, refine the theory, and apply it to problems to which it had previously only alluded [4]. In our opinion software development is an embodiment of articulation of theory.

The development of active computer algorithms from theoretical concepts is far more challenging than is generally recognized in the geomechanics community. To be successful, it requires great skill, and some trial-and-error, to overcome numerical difficulties and instabilities that occur during computations [5].

It is important to note that computational algorithms are developed in the course of many geomechanics research projects. However, most of these are still not active – they have not been converted into well-tested and ready-to-be-incorporated-into-a-program format.

Why is Software Necessary to Geomechanics?

Aside from the fact that computers perform calculations much faster than humans, there are several other reasons for developing active computational algorithms out of theoretical concepts. Often theoretical principles are not easily applied to specific instances. In other cases, even though governing principles may be well understood, the local behaviours generated by specific sets of conditions may not be known. In yet other instances, direct application of governing principles lead to equations, solvable only with the aid of computers.

Each year also, many research papers submitted for publication receive reviews that, although the ideas they express are interesting, they offer little practical value. The development of active computational algorithms and software can bring such ideas closer to real-world application. At the least it can help better assess how practical or impractical a concept is.

The Intellectual Challenges of Software Development

Creating computational algorithms out of theory is not a straightforward process. It frequently relies on simplifications, approximations, idealizations, and falsifications (assumptions that contradict theory) to make things work [5]. These actions can be classified under the single banner of extra-theoretical work. The development of software must also consider limitations such as computer memory and speed [5].

Due to the potential for errors (primarily from the required extra-theoretical work), the process of transforming theory into computational algorithms requires great care. The sources of these errors include:

1. Discretization – the use of discretization inevitably results in the likelihood of truncation errors. It can also cause instabilities in the solution process, which in turn can introduce artifacts into the solution that may go undetected [5].
2. Assumptions, simplifications, falsifications – any assumptions made during the creation of an algorithm can have unintended consequences.
3. Use of knowledge and routines drawn from several different sources – software development inevitably relies on numerical procedures provided in books and other sources. The algorithms provided in Numerical Recipes for C [6] are a typical example. These algorithms may however have errors or difficulties the software developer may not be aware of.
4. Programming errors or bugs – these are mistakes made during the writing of software codes. They either produce unintended consequences, wrong results, or crashes.

As a result of the potential of errors in active computer algorithms, software results are not automatically trustworthy and therefore cannot be taken for granted [5]. Results are verified or validated through comparisons with known (often closed-form) solutions. Validation also involves checks of the different routines and algorithms used in a program to ensure they perform as prescribed. Software and computational algorithms must also be checked for robustness. Robust programs work well not only under anticipated conditions, but also under unusual circumstances that push algorithms to their limits.

Software Development Produces New Knowledge

The process of developing computational algorithms from theoretical principles can produce new knowledge and insights. The new knowledge stems from questions that arise from the attempts to

apply governing principles to a wide range of particular initial conditions. These questions force software developers to examine aspects of theoretical concepts and their implications much more closely than was originally done during development of the theories.

New knowledge can also result from observations of the local behaviours arising from application of theory to specific conditions, and from thought experiments possible with software. The insights gained can lead to refinements or improvements. At other times the questions raised can demand newer answers.

A specific geomechanics example of how software development produced fresh insights will be described in the Examples section of this paper. It involves the development of an algorithm for calculating stresses in multi-layered materials.

3.3. Lesson #3: Parsimony is invaluable; avoid complexity, embrace clarity.

In software development, by parsimony we mean taking great care to develop computational algorithms that require the smallest possible number of parameters in order to explain or model behaviour. It also means avoiding unnecessary complexity. It encourages approaches that are as straightforward as possible, and discourages the use of solutions that may be very clever but may not be robust or general enough, i.e. solutions that may not be capable of handling unusual cases. Lastly, parsimony demands that code be written in the simplest, self-explanatory manner.

There are many reasons for adhering to the principle of parsimony in the development of computational algorithms. Every parameter included in an algorithm introduces additional uncertainty. Keeping parameters therefore to a minimum reduces uncertainty in the solution process. Simple algorithms are also much easier to understand and explain than complicated ones. In addition, by keeping code as simple as possible, it becomes much easier to find errors and thus minimizes their occurrence.

The last point cannot be stressed enough. Very often the details of computer codes are poorly documented. Computer languages evolve with time. Programmers and researchers move on, taking with them important knowledge on the inner workings of algorithms. If parsimony is neglected therefore during software development, with time it becomes virtually impossible to update or fix subroutines.

Adhering to parsimony does not imply use of computational algorithms that are so simple as to ignore key aspects of the problem being solved. Like Albert Einstein once said, “Things should be made as simple as possible, but not any simpler.” This underlines some of the challenges facing today’s practice of geomechanics. In many cases we are analyzing problems in the field with overly simple approaches, even though more accurate techniques exist, which, to the user, are just as simple to apply.

A case in point is the analysis of rock falls using stereomechanical (particle) models. These models do not include the influence of the size of a falling rock, and do not properly account for its mass. It will be shown later in the paper that the behaviour of rock falls can be better simulated with rigid body impact mechanics (RBIM). Even though the framework for RBIM is more involved than the stereomechanical approach, RBIM captures the physics of the problem much better. At the same time, for the user of an RBIM algorithm, it is not any more complicated than a stereomechanical model. In fact it requires input parameters, which are more intuitive to the problem and easier to obtain than those used in particle analysis.

3.4. Lesson #4: Interface design must not be an afterthought; it is fundamental to practical engineering.

If software is to be a useful tool to geomechanics specialists, especially to practitioners, then its user interface – the options and tools through which users interact with the program – must be well executed. It is a tragedy that many engineering programs hardly pay attention to good interface design, completely ignoring the resulting burden imposed on users. Many excellent research findings that could have benefited the geomechanics community have been consigned to library shelves, because users find the software, which embodies these findings, too cumbersome or difficult to understand or use. Academia stands especially guilty. We seem to stick to a bad formula that simply says: “All that matters is it works!"

Geomechanics research findings would have much greater impact on the state of professional practice,

if developers of software resulting from research would consider that the purpose of any program is to allow users greater freedom to concentrate on solving the problem at hand, freedom to express problem-solving skills. This is accomplished when a program removes tedium and frustration from the problem-solving process by assuming the less skillful chores.

A user-friendly interface must enable users to efficiently solve their problems, and to be satisfied with the whole process. It must allow them to quickly see available options and to understand how to use these options to achieve their goals. It must not unnecessarily burden users with the inner workings of code. A well-designed program should perform a majority of the work, while requiring a minimum of information and input from users. Unless the use of a program is far less burdensome than the problem being solved, the program's application becomes questionable.

This is not to advocate the use of black boxes. We will explain this through an example of the application of the finite element method (FEM) to slope stability analysis. Although it is absolutely essential that users have good understanding of the FEM, its capabilities, limitations and caveats, it would be crippling if users had to know full details of meshing and matrix solution algorithms before they could actually create models and solve problems.

The design of a user interface must consider the productivity of users. It must ensure a short, gentle sloping learning curve. Practitioners are keenly aware that people cost a lot more money than computers and software. These costs constitute a primary reason why the state of geomechanics practice lags behind state-of-the-art research. Industry finds it hard to justify the costs and delays inherent in using knowledge stored in written form, or in poorly written software; the expected value of the insights to be gained does not match the cost of people time.

At this stage we would like to stress the importance of including visualization tools in geomechanics programs. Such tools are not merely means of transferring facts. Most computations result in a bunch of numbers, sometimes huge amounts of them. As such visualization is a powerful aid to understanding, allowing users to gain precious insights into problems. It makes comparisons between alternative solutions much easier to make. Software developers must understand that visualization is by far one of the most effective means of communicating to humans.

3.5. Lesson #5: Users have practical constraints; keep things as simple as possible.

Successful transfer of geomechanics research advances and knowledge to practice of the discipline through development of software will be realized only by considering the environment within which the discipline is practiced. We have to fully understand the constraints and limitations faced by practitioners.

Typically, geomechanics practitioners work with limited budgets, tight project schedules and very limited knowledge on material properties and subsurface conditions. Whether or not they have the tools to model problems, practitioners have no choice but to make decisions. Given the option, however, most practitioners would use software to aid their decision making, since computational and modelling tools enable logical use of available information.

The time constraints practitioners face can be quite severe. The situation of a rock mechanics engineer in a mine is a typical example. He/she has several tasks to fulfill each work day in different parts of the operation. This leaves hardly any time for carrying out numerical simulations.

The conditions we have described demand software and computational tools that are as simple as possible. These tools must not require more skill than most users are expected to possess. Generally, specialists in the field, although knowledgeable and experienced with practical engineering, are not specialists in the details of numerical analysis methods. Therefore programs must minimize the need of users to be well schooled in the intricacies (rules of thumb, exceptional situations, etc.) of numerical methods.

We have also learned that under many of the conditions confronting users, well-designed, simple software that capture the fundamental physics of behaviour are empowering. In the world of practitioners, easy-to-understand and easy-to-use tools are much more useful than intricate ones, which may be more accurate. The simpler tools make it easier to think through problems. For

practitioners, any preoccupation with the details of computational algorithms (details which they cannot get right unless they have the relevant numerical methods expertise) is a barrier, and actually produces less understanding, and poorer prediction.

3.6. Lesson #6: Uncertainty is king; make room for it.

Because geological materials are formed under a broad variety of complex, physical conditions, the history of which is not known, geomechanics involves large uncertainties. Single-point predictions of quantities have therefore practically zero likelihood of ever being realized in such a world. If room is therefore not made in geomechanics software analysis to accommodate uncertainty, any conclusions reached will be open to question.

In the application of geomechanics software to real-world problems, uncertainty can be dealt with in different ways. Popular ways of handling uncertainty include parametric and scenario analysis – the assessment of possible ranges of behaviours through variation of input properties and consideration of different conditions. Such analyses are very difficult to conduct, especially given the constraints described above, if geomechanics software and computational routines are difficult to modify or take too long to compute. For all practical purposes, it is nearly impossible if the situation is compounded by poor interface design.

Although parametric and scenario analyses are very useful, statistical simulation is an even more powerful approach. This is because it enables uncertainty to be quantified and the probabilities (likelihoods) of outcomes to be estimated.

We believe that in many situations, the combination of statistical simulation with simple models produces more realistic capturing of the true nature of geomechanics behaviour than the use of sophisticated single-point procedures.

3.7. Lesson #7: Legacy codes may be outdated; but think carefully before rewriting.

Often in software and computational algorithm development, it becomes necessary to modify older (legacy) pieces of code or include them in a new routine. These program routines may be obsolescent due to the age of the programming languages in which they were developed.

Such codes often contain a wealth of knowledge, acquired over the history of the codes' development with huge time investments. They embody all sorts of workarounds and modifications to ensure they perform as required. Unfortunately, like most codes, they are often inadequately documented. In addition, personnel such as post-docs and students, instrumental in the development of such codes may have moved on to other places.

The issue then is how to modernize such codes, while keeping their functionality intact. The task can be hugely challenging. The topic of how best to deal with legacy codes is beyond the scope of this paper. The lesson we would like to share though is that any undertaking to rewrite such legacy codes should be well thought out. It is a risky task that often requires more resources and time than most developers budget for. The key is to understand that program codes contain a lot of expert knowledge, some of which is not formal. They may seem poorly written or ugly, but great care has to be exercised in modifying or rewriting them.

4. THE LESSONS AT WORK – THREE EXAMPLES

We will now look at three concrete examples of the lessons outlined above at work. The examples involve the development of active computational algorithms for

1. Calculating stresses in elastic half-spaces comprising multiple parallel layers of material
2. Simulating the behaviour of rock falls, and
3. Analyzing slope stability with the finite element method.

4.1. Elastic Stress Analysis of Multilayered Material

There is a class of geomechanics problems for which the prediction of settlement is very important. In the design of shallow foundations for buildings on soils, for example, it is often the case that settlement (especially differential settlement), rather than bearing capacity, is of the greatest concern. In most of these cases, such as the example of the surface loading due to three buildings shown on

Fig. 1, three-dimensional behaviour is very dominant and cannot be meaningfully approximated with two-dimensional analyses.

The first step in the process of computing settlements is the determination of the stress distribution as a result of surface or sub-surface loadings. Despite all the advances of today, elastic stresses used in settlement calculations are routinely obtained from classical solutions such as those of Boussinesq and Westergaard. The problem with this is that these solutions make very simplifying assumptions, which as will be shown later, can distort results. For example, they assume the half-space of soil material to be homogeneous.

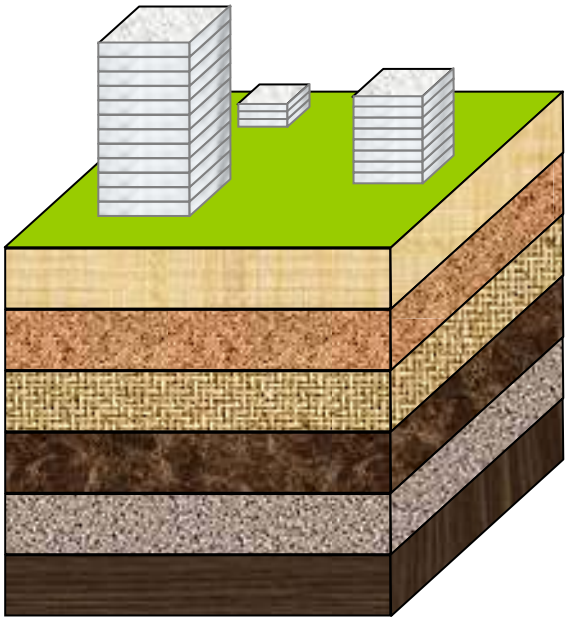

Fig.1: An illustration of three different-sized buildings on a site comprising layered materials. Due to the asymmetry of the surface loadings, it is clear that the stresses induced in the material strata will have a complex, three-dimensional nature.

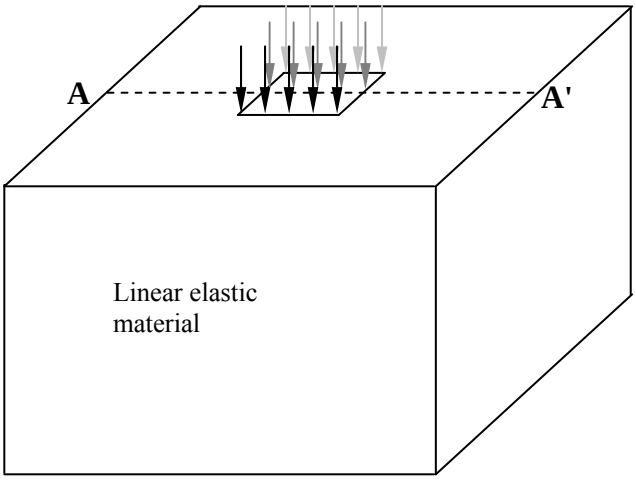

Fig 2: Problem of unit load applied over a rectangular area on the surface of an elastic half-space.

In theory, the problem can be tackled with numerical methods such as three-dimensional finite element or finite difference analysis. The difficulties however with these methods of solution are that they:

1. Require significant computing resources not (as yet) routinely available to practitioners
2. Substantial user expertise in ensuring adequate three-dimensional meshes.

Generally, three-dimensional meshing is quite challenging. It becomes even more challenging in problems involving thin seams (layers) of material. If thin layers are not properly discretized into elements, their elements will have poor aspect ratios, which in turn compromise the accuracy of results. In addition, if thin layers are not discretized with sufficient numbers of elements in the direction of their thickness, then rapid stress gradients in this direction may be missed. This aspect of finite element and finite difference modeling places the onus on users to ensure adequate meshes.

About twenty years ago, a theoretical method was developed for solving for the stresses and displacements in two bonded elastic half-spaces due to the application of a point load [7, 8]. Based on what is known as the method of images, the method used reflection and transmission matrices to calculate the required stresses and displacements. When the technique was originally published, it was deemed theoretically interesting, but of little practical value.

Recently, the geomechanics research group at the Lassonde Institute of the University of Toronto looked into developing an active, fast, three-dimensional computational algorithm for calculating stresses due to foundation loads. It was decided to develop one based on the method of images solution, especially since such an algorithm would be meshless – it would only require integration of loads over their area(s) of application.

Considerable extra-theoretical work had to be done in order to achieve the goal. The original two-material method had to be extended to any number of materials. Upon attaining this end, the contribution of the infinite images generated in the method had to be carefully studied. It was then discovered that, paired in a particular way, several terms cancelled out, meaning only very few terms were actually required. Next a robust integration

scheme [9] had to be employed to enable the new computational algorithm to produce accurate stress results for all loading configurations.

The validity of the new method was established through comparisons to known analytical solutions or to solutions obtained from three-dimensional, elastic finite element analysis. One such verification is described next.

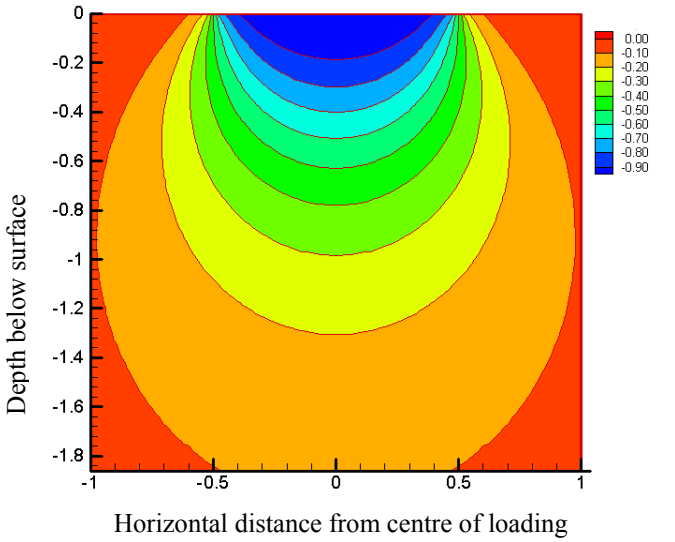

Fig. 3: Contours of the vertical stress distribution on the vertical plane through A-A' for a homogeneous half-space.

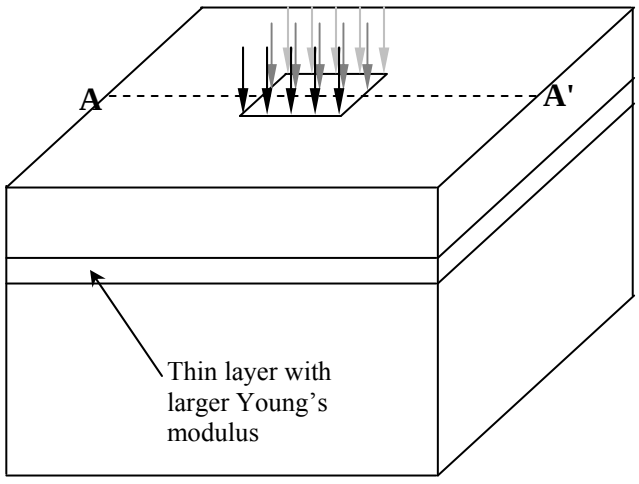

Fig 4: Problem of unit uniform load applied over a square area on the surface, but this time with a stiffer, thin layer at some depth below the surface.

Fig. 2 shows the application of a unit, uniform load to a square surface area of an elastic, homogeneous half-space. We will consider the stress distribution on a vertical plane passing through A-A' induced by the applied load. Contours of this stress distribution are shown on Fig. 3. These contours are identical to those predicted from the analytical Boussinesq solution to the problem.

An interesting thought experiment was performed with the new computational tool. A thin horizontal layer with a higher stiffness (larger Young's modulus) was inserted into the problem as shown on Fig. 4. Two cases of Young's modulus were considered: in the first case the thin layer was five times stiffer than the original material, and in the second case ten times stiffer.

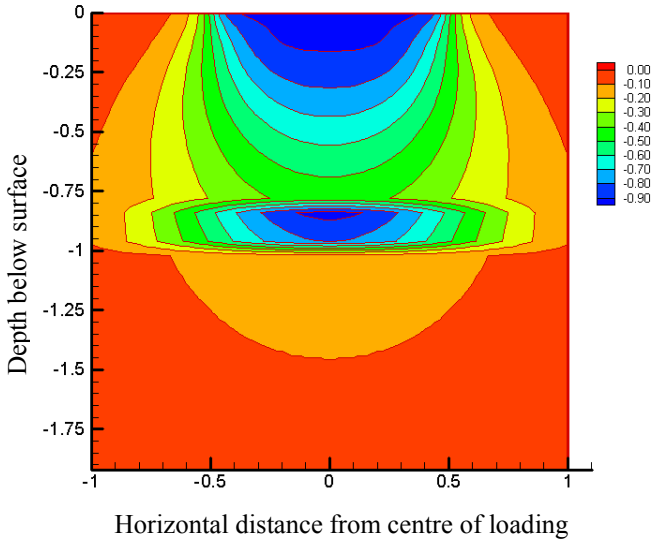

Fig. 5: Contours of the vertical stress distribution on the vertical plane through A-A' for the case when the thin middle layer is five times stiffer than the upper and lower layers.

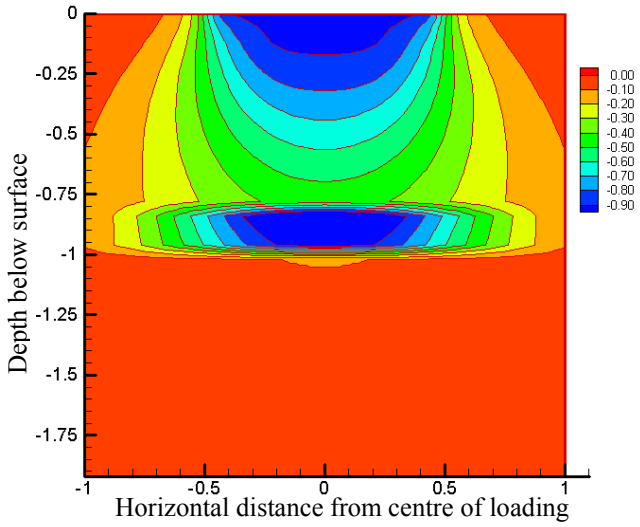

Fig. 6: Contours of the vertical stress distribution on the vertical plane through A-A' for the case in which the thin middle layer is a ten times stiffer than the upper and lower layers.

The stress distribution on the vertical plane through A-A' are shown on Figs. 5 and 6, respectively. The results are very different from the homogeneous case. In both cases the stiffer thin layer essentially

shields the lower material zone from the applied load. The shielding is more pronounced in case 2.

This example illustrates several of the lessons outlined above. Passive knowledge deemed impractical 20 years ago was converted into an active software tool suitable for routine analysis. The process of articulating the theory required an extension to the theory, and extra-theoretical work to develop a fast algorithm. It yielded new insights into the use of reflection and transmission matrices for elasticity, including the observation of terms that negate one another. It also enabled the conducting of thought experiments, which allow a user to gain better intuition into the influence of different material stiffnesses.

It was also learned that although the new active algorithm for elastic stress analysis was restricted to parallel material layers, it offered significant improvements over the existing simple approaches used in current practice. By not requiring meshing, the new algorithm is more amenable to practical analysis – users do not have the additional burden of ensuring good meshing. Because it eliminates meshing, the method ensures that models can be set up quickly, and results obtained significantly faster than is possible with finite element or finite difference modeling.

4.2. Rigid Body Impact Mechanics Analysis of Rock Falls

Rock falls can pose significant hazards to infrastructure such as highways, buildings, and mine open pits and, sometimes, result in personal injury or death. Their prediction is a difficult task fraught with uncertainty. The geometries of natural slopes, including the location of the boundaries between different slope materials, can vary considerably from one cross-section to the other. The properties of a slope's materials can also vary widely, while the location and mass of rocks that may dislodge are also uncertain. Probabilistic simulation has proven very useful in analyzing this class of geomechanics problems [10].

The interactions of a falling rock with a slope surface mainly consist of bouncing, sliding and rolling. In some cases the rock may fracture into smaller pieces upon impact. The primary factors controlling impact interactions and trajectories of a falling block of rock are the:

1. Geometry of the slope
2. Shape and mass of the rock, and
3. Energy dissipated upon impact of the block on a slope segment.

To mitigate the effects of rock falls with measures such as restraining nets and ditches, engineers need to predict the velocity, frequency, height of bounce and run-out distance of potential falling rocks. Given the large uncertainties of the problem, it is best to obtain statistical distributions of these quantities in order to design effective remedial measures.

Most current rock fall simulation models [11, 12, 13, 14] are based on particle (stereomechanical) models. These models represent falling rocks with point masses. Two other input parameters, the normal and tangential coefficients of restitution (R_N, and R_T, respectively), are required in these models.

The normal coefficient of restitution is defined as the ratio of the normal component of the outgoing velocity of a particle after it collides with a surface, $V_{out,\,N}$ to the normal component of the velocity prior to collision (incident velocity), $V_{ini,\,N}$, i.e. $R_N = \frac{V_{out,\,N}}{V_{ini,\,N}}$.

The tangential coefficient, R_T, is similarly defined for the tangential components of outgoing and incoming velocities.

The primary merit of stereomechanical models is their simplicity and speed of computations. The latter attribute makes them very conducive for probabilistic analysis. They suffer though from three serious deficiencies:

1. Shape, which in reality has significant influence on trajectory, is ignored.
2. Generally, mass although incorporated, does not affect the total path of a falling block. It is not considered during impact interactions, which play a key role in determining the overall trajectory, but is only used to compute energies.
3. In reality, normal and tangential coefficients of restitution are not intrinsic parameters. It will be shown later that they depend on factors such as incident angle, frictional characteristics of the falling block-slope contact, and on the point on the falling object (for non-circular shapes) that collides with a surface [15].

Simply put, the particle models widely used in today's practice are too simplistic, ignoring important facets of the problem. However, in the absence of better active computational tools, they are the tools for practical application.

There are a number of approaches that better model the true physics of the rock fall problem. One of them is the powerful discrete element method (DEM). It can accurately model rock-slope collisions and can even simulate breakup. The main drawbacks of the method though are its slow computational speed, which rules out probabilistic simulations, and the number of input parameters required. Some of these parameters, such as spring stiffnesses, are not observable, and therefore not easily measured.

The slowness of the DEM arises mainly from the need to detect contact and the smallness of the time steps required to adequately model impact interactions. If care is not exercised with time step magnitude, the resulting behaviour might be incorrect. Unfortunately very few practitioners have enough experience with this intricacy to be able to specify correct time steps.

An alternative approach that is less sophisticated, but sufficiently captures the essential behaviour of rock falls, is rigid body impact mechanics (RBIM). RBIM models the impulses (but not the contact forces) that develop during collisions and the dynamic response of the colliding bodies [15]. It uses the equations of kinematics and motion. Unlike the DEM, it assumes the period of contact during which the velocities of bodies change to be instantaneous.

The term RBIM is an oxymoron. Ideal rigid bodies do not deform. However, for the method to adequately model impact behaviour (namely the impulses arising out of collision that change the velocities of colliding bodies), it assumes the region of contact between colliding bodies to be a very small (localized) region.

In RBIM, collision is characterized by two phases – compression and restitution. The compression phase occurs when a falling body first impacts a surface. During this phase the kinetic energy of the body is converted into internal deformational energy due to the contact force that develops. The compression phase is followed by restitution, a phase during which the elastic component of the internal deformational energy is released and converted into an exit kinetic energy.

The input parameters required for RBIM analysis are all observable and measurable. In addition to shape, mass, initial velocities, etc., it requires the friction coefficient of the contacting surfaces, and an energetic coefficient of restitution. The energetic coefficient of restitution is simply the square root of the ratio of the elastic energy recovered during restitution to the internal energy of deformation absorbed during compression. It can be readily measured or estimated. A value of 1 indicates a perfectly elastic collision in which no energy is lost, while a value of 0 implies a perfectly plastic collision in which the impacting body does not separate and fly off.

From its description, it can be seen that the parameters of RBIM, in addition to being measurable, are very intuitive to engineers. Tests have also shown that it models the real behaviour of falling objects very well. Upon impact objects can bounce, slide, rotate or stick, or exhibit combinations of these behaviours.

To illustrate the advantages of RBIM over stereomechanical modeling of rock falls, we will consider the example of a falling rock impacting a horizontal surface. The normal component of the incoming velocity ($V_{ini,\,N}$), the tangential component of the incoming velocity ($V_{ini,\,T}$) and angular velocity (ω_{ini}) of the rock are -10m/s, 10m/s and 0rad/s, respectively (Fig. 7). (Rotation is assumed positive in the clockwise direction.)

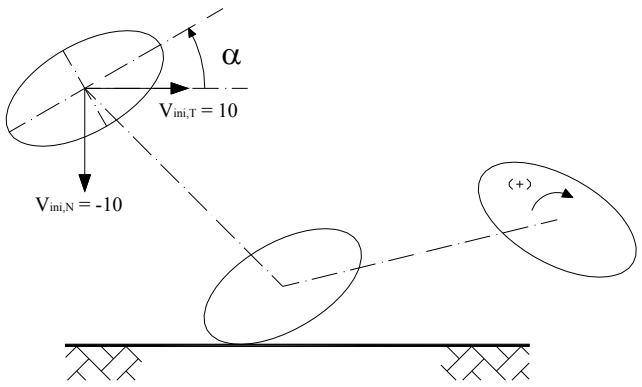

Fig. 7: The behaviour of a falling rock upon impact with a horizontal surface.

We will consider two shapes of the falling rock: a sphere and an ellipsoid (with major axis of 2m and minor axes of 1m). The rock/surface contact is

assumed to have a friction angle of 25° (coefficient of friction μ = 0.47) and the energy coefficient of restitution assumed, e^*, assumed equal to 0.8.

We would like to draw attention to the intuitiveness of the parameters friction angle and energy coefficient of restitution. Given the conditions prevailing at a field site, it is much easier to estimate these values than to guess values for the conventional normal and tangential coefficients of restitution (R_N and R_T) used in stereomechanical analysis.

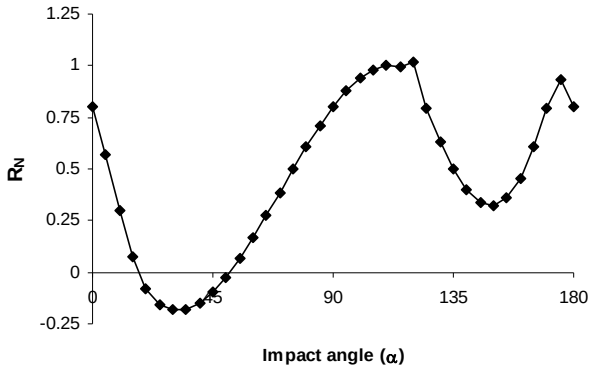

Fig. 8: Plot of normal coefficient of restitution, R_n, with impact angle for an ellipsoid.

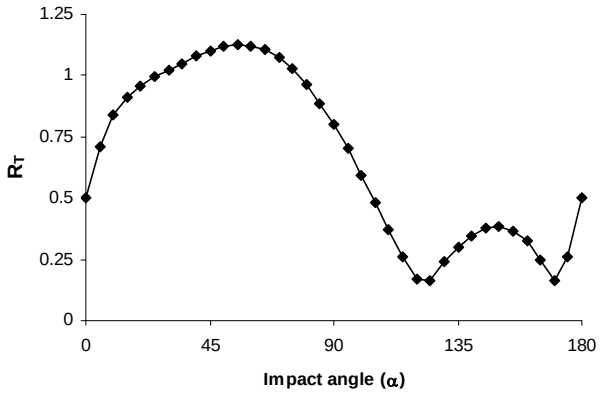

Fig. 9: Plot of tangential coefficient of restitution, R_t, with impact angle for an ellipsoid.

The equations of RBIM enable prediction of the outgoing normal, tangential and angular velocities after impact. Using these values, the conventional R_N and R_T can be easily calculated with the equations described earlier. For the case of the sphere, the RBIM calculates R_N = 0.8, R_T = 0.71, and recovery of 68% of the total kinetic energy prior to collision.

In the case of the ellipsoid we discover that the outgoing velocities, and thus the conventional coefficients of restitution, as well as the energy retained are found to strongly depend on the orientation of the ellipsoid (α on Fig. 7) at impact. Figs. 8, 9, 10 and 11 show plots of R_N, R_T, the rebound rotational velocity, and the percentage of energy retained, respectively, as functions of impact angle α.

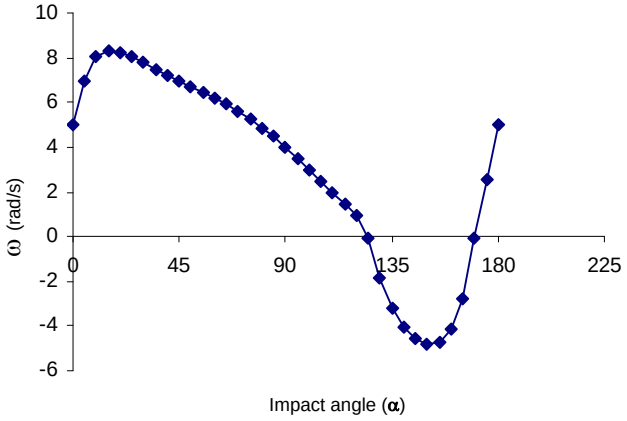

Fig. 10: Plot of angular rotation with impact angle for an ellipsoid.

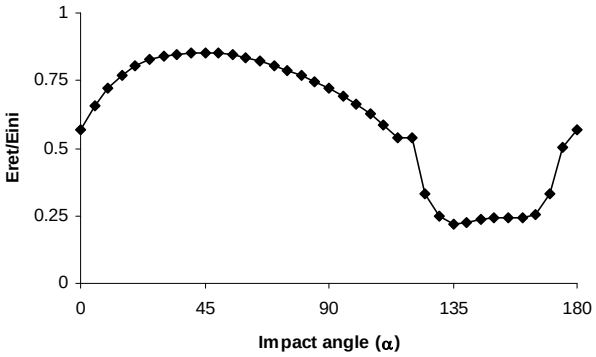

Fig. 11: Plot of the ratio of total energy retrieved (E_{ret}) after collision to total energy before impact (E_{ini}) against impact angle for an ellipsoid.

Figs. 8 through 11 show that, for the ellipsoid, R_N, R_T, the rebound rotational velocity, and the percentage of energy retained are all highly nonlinear functions of impact angle. The plot of R_T on Fig. 9 reveals that the tangential coefficient of restitution initially increases with increasing impact angle, attaining a maximum at α = 55°. Thereafter R_T begins to decline and reaches a minimum at α = 125°. The behaviour of the curve after that point is also highly variable. Similarly, the behaviour of the

normal coefficient of restitution, R_N, shown on Fig. 8 varies nonlinearly with impact angle.

Fig. 10 shows that the magnitude and direction of rotation is also highly dependent on the angle of impact. On Fig. 11 it can be seen that the energy retained in eccentric collisions can be as high as 85% and as low as 22%.

These analyses of impact behaviour of a relatively simple ellipsoidal shape clearly demonstrate that R_N and R_T vary in complex manners with impact angle for non-circular shapes. They are definitely not the constants input into stereomechanical models. An attraction of RBIM for practical rock fall analysis is it predicts all this irregular behaviour without asking users for unusual inputs or a greater number of parameters.

Another attractive feature of RBIM is that it has the potential to be sufficiently fast so as to be used for practical probabilistic analysis. Currently, research is being done at the University of Toronto to develop an active computational algorithm that performs RBIM analysis of rock falls. The results so far are quite promising.

Some issues however remain to be resolved. Two of them concern how to deal with sharp corners in order to eliminate spurious bounces, and how to speed up computations. The University of Toronto research team is experimenting with different schemes, simplifications and assumptions that will address these questions.

Several of the lessons learned from geomechanics software development are epitomized in this discussion of rock fall analysis. The example illustrates how the gap between research and the state of geomechanics practice can be narrowed through software development. It shows how the RBIM, from the user perspective, is no more complicated than existing stereomechanical models, but offers significantly better replication of true behaviour. At the same time, through the example we obtain glimpses of the creative and hard work required to make theory active.

4.3. Shear Strength Reduction Method of Slope Stability Analysis

The assessment of the stability of slopes is a very common problem in geotechnical engineering. The most popular measure of the stability of a slope is the factor of safety. Traditionally it is computed using limit-equilibrium method-of-slices analysis.

Limit-equilibrium analysis makes a number of simplifying assumptions including:

- *A priori* judgments on the shapes or locations of failure surfaces
- Assumption that the sliding mass moves as a rigid block, with the movement occurring only along the failure surface
- Assumption that the shear stresses are uniformly mobilized along the entire length of the failure surface, and
- Various assumptions on interslice forces.

The power of limit equilibrium methods lies in the fact that they are very simple, produce very reasonable answers in short computational times and require relatively small numbers of input parameters. In addition, engineers have acquired great experience with these methods over decades of use. As a result the conditions under which the answers of different limit-equilibrium techniques can be trusted are well established.

The primary disadvantages of limit-equilibrium analysis stems from its *a priori* assumption of a failure surface, and omission of stress-strain behaviour. As a result of the omission limit-equilibrium analysis cannot reveal the development of the critical failure mechanism and cannot predict deformations at failure.

A more complete solution of slope stability can be determined if the boundary conditions of the problem and constitutive laws of materials are known, and the conditions of equilibrium and strain compatibility enforced. The finite element method (FEM) is the most common numerical technique for performing such analyses.

A method known as the Shear Strength Reduction (SSR) method originally devised in the mid-1970s [16, 17], enabled FEM analysis to be used in calculating slope factors of safety. It overcomes the liabilities of limit-equilibrium methods described above, and is more readily extended to three-dimensional analysis.

Conceptually, the SSR method is very simple: to determine the factor of safety of a stable slope, systematically reduce (divide) the shear strength of its material by factors until the slope is brought to complete failure [18]. The reduction factor that brings the slope to the verge of failure is then the

factor of safety. For the linear Mohr-Coulomb strength envelope the method is very straightforward.

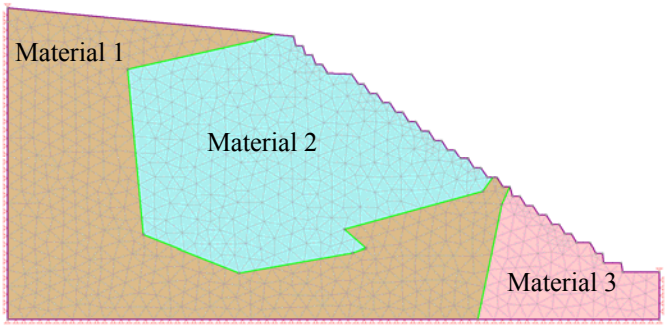

Fig. 12: Finite element model of an open pit slope consisting of three zones of material.

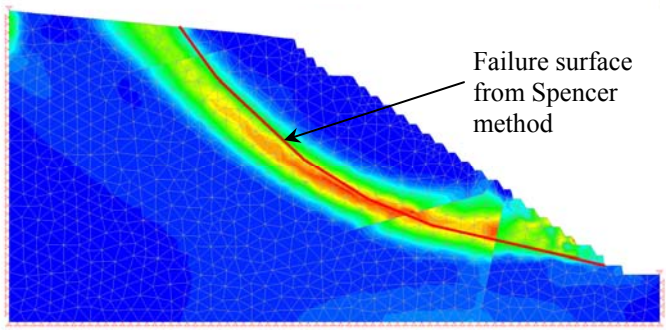

Fig. 13: Plot of the contours of maximum shear strain at failure. This model assumes all three materials to have the same stiffness. The failure surface obtained from conventional limit-equilibrium (Spencer) analysis is superimposed.

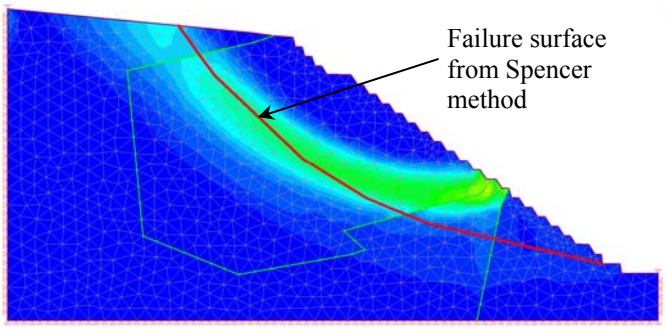

Fig. 14: Plot of the contours of maximum shear strain at failure for the model that assumes the toe material to be stiffer than the other two. The failure surface obtained from conventional limit-equilibrium (Spencer) analysis is superimposed.

The solution of an FEM model is stable when all equilibrium conditions are satisfied, and unstable or non-convergent otherwise. (The transition from stable to unstable behaviour is often characterized by a sharp increase in displacements.) As a result, in the SSR method solution convergence is used as the criterion for determining the onset of slope failure.

As an example, SSR analysis reveals that when there is great contrast in the stiffnesses of the materials in a slope, although the factor of safety predicted by the SSR method does not differ much from that obtained through conventional limit-equilibrium analysis, the location and shape of the failure mechanism can be quite different [19].

Three images are provided below that show the SSR analysis of a slope with multiple material zones. The finite element model of the slope is shown on Fig. 12.

If all three materials are assumed to have the same Young's modulus, the factor of safety and failure mechanism (band of highest maximum shear strains) predicted by SSR analysis and conventional limit-equilibrium analysis with the Spencer method are very similar. These results (the contours of maximum shear displacements and the slip surface predicted by limit-equilibrium analysis) are shown on Fig. 13.

The stiffness of the material at the toe (Material 3) of the slope was then increased a hundred times and the analysis re-run. The new slope failure mechanism is visible on Fig. 14. Although the factor of safety remained the same, the shear strain pattern changed, especially in the toe area.

The SSR results are very intuitive. They show that the shear deformations tend to concentrate in the softer materials. These results can have quite an impact, for example, on where to place instruments in order to monitor the onset of failure.

The SSR method is a classic illustration of how wide the gap between research and the state of geomechanics practice can be. Although in principle SSR analysis could be performed with any geotechnical FEM program, it was not applied to routine slope analysis until quite recently.

In our opinion, aside of computing speed in the past, the primary reason was related to the manual effort involved in setting up the several models with different factored strength, especially for models comprising multiple materials. A user also had to properly keep records in order to know which file used which reduction factor.

The SSR example also enables us to see how software designed to be simple for users can alter

the state of practice. Until the implementation of automatic meshing, the user who wanted to perform such analysis had to manually define meshes for each of the models. Upon running all models the user then had to open each model in order to determine the onset of instability and hence the factor of safety. And after all this work, it was still not easy to visualize the development of the failure mechanism.

Itasca [20] and PLAXIS [21] were the first to develop commercial software that automatically performed SSR analysis. The latest release of Phase^2 [22] includes many interface and computational engine additions that aid ready creation of SSR models and make interpretation of SSR results easier. It also allows SSR analysis for Hoek-Brown and Generalized Hoek-Brown materials, making the method readily applicable to rock slopes.

Users have been also given the ability to seamlessly import limit-equilibrium models into the FEM program – the geometry is automatically meshed and each material in the model assigned a default stiffness value. As a result users can choose to analyze conventional models with the new tool at hardly any costs in terms of effort to set up models.

Since the SSR method became accessible to practitioners, it has rapidly gained in popularity. Practitioners use it to gain insights into problems in ways previously not possible with limit-equilibrium tools.

5. A MESSAGE TO THE ACADEMIC COMMUNITY

In academia, rewards such as research grants, recognition, degrees, etc., accrue to those who seek for new knowledge as defined by traditional criteria used in science. The traditional assessment criteria, we believe, have not been very kind to software development.

Through the arguments offered in this paper, we hope we have convinced the academic community that geomechanics software development is not a trivial and simple-minded exercise in transforming known theoretical concepts into computer algorithms. It is exciting work, integral to the practice and advancement of science and engineering. It generates new knowledge of significant scientific value in diverse ways.

It is our hope therefore, that software development will receive greater attention and will be given greater support. This will go a long way in advancing the real-world practice of geomechanics. It has the potential to radically improve our solutions and widen the scope of problems we can tackle.

6. CONCLUDING REMARKS

In the 17th century, academia discovered the transforming influence of academic publishing, the formal process of subjecting new ideas to critical review and ensuring open, transparent and widespread sharing of ideas. Initially academic publishing was scorned, but today one cannot comprehend what the world would be like without this vehicle. Technological changes, especially the Internet, are rapidly changing the landscape of academic publishing as more and more publications move to the electronic format, and making the dissemination of ideas more widespread than ever.

We believe that because software makes knowledge active it presents an even more powerful medium for disseminating scientific and engineering knowledge; it is a much more ready means for testing ideas. Reminiscent of the beginnings of academic publishing, in many important and influential establishments such as universities and funding agencies, user-friendly software development is not deemed an endeavour of scientific calibre.

We hope that this paper will help the geomechanics community to better appreciate the impact software and active computational algorithm development can have on the state of our practice. They present incredible opportunity for transforming theory into practice. They offer means of gaining new insights and performing thought experiments.

Most of the lessons we talk about, we learned from experience. We share them to encourage others to take up this enterprise, seeing it is as an exciting way of doing geomechanics. It contributes to the advancement of both research and real-world practice. It is our hope also that these lessons will help people, who evaluate research proposals, to better appreciate the important position software development occupies in the science of geomechanics.

ACKNOWLEDGEMENTS

The authors would like to express their gratitude to the geomechanics research group at the Lassonde Institute of the University of Toronto and the software development team at Rocscience Inc. Without their combined effort this work would not have been possible. Special thanks are due Dr. Vijayakumar Sinnathurai and Mr. Parham Ashayer for their assistance with the examples.

REFERENCES

1. The Free Dictionary. Available online from http://www.thefreedictionary.com .
2. Armour, P. G. 2000. The case for a new business model: Is software a product or a medium? *Communications of the ACM*, Vol. 43, No. 8, pp. 19-22.
3. Armour, P. G. 2001. Software as currency: With its valuable characteristics, software is the most effective storage medium, making it the future currency of the world. *Communications of the ACM*, Vol. 44, No. 3, pp. 13-14.
4. Kuhn, T.S. 1970. *The structure of scientific revolutions*. 2nd ed. Chicago: University of Chicago Press.
5. Winsberg, E. 2003. Simulated experiments: methodology for a virtual world. *Philosophy of Science*, Vol. 70, pp. 105-125.
6. Press, W.H., et al. 2002. *Numerical recipes in C++ - the art of scientific computing*. 2nd ed. Cambridge: Cambridge University Press.
7. Vijayakumar, S., Cormack, D.E. 1987. Green's functions for the biharmonic equation: bonded elastic media. *SIAM Journal on Applied Mathematics*, Vol. 47, No. 5, pp. 982-997.
8. Vijayakumar, S. 2005. *New boundary element methods for solid mechanics: integration methodology and new Green's functions* [Ph.D. thesis]. Toronto: University of Toronto.
9. Vijayakumar, S., J.H. Curran, and T.E. Yacoub. 2000. A node-centric indirect boundary: three-dimensional displacement discontinuities. *Computers and Structures*, Vol. 74, No. 6, pp 687-703.
10. Stevens W.D. 1998. *Rock fall: A tool for probabilistic analysis, design of remedial measures and prediction of rock falls* [Masters thesis]. Toronto: University of Toronto.
11. Hoek E. 1990. *Rock fall - a program in Basic for the analysis of rock falls from the slopes*. Unpublished notes. Golder Associates/University of Toronto.
12. Rocscience Inc. 1998. RocFall – risk analysis of falling rocks on steep slopes.
13. Colorado Department of Transportation. 2000. Rock fall simulation program.
14. Guzzetti F. et al. 2002. STONE: a computer program for the three-dimensional simulation of rock-falls. *Computer & Geosciences*, Vol. 28, pp. 1079-1093
15. Stronge W.J. 2000. *Impact mechanics*. Cambridge: Cambridge University Press.
16. Smith I.M., Hobbs R. 1974. Finite element analysis of centrifuged and built-up slopes. *Geotechnique*, Vol. 24, No. 4, pp. 531 - 559.
17. Zienkiewicz, O. C., Humpheson, C., and Lewis, R. W. 1975. Associated and non-associated visco-plasticity and plasticity in soil mechanics. *Geotechnique*, Vol. 25, No. 4, pp. 671–689.
18. Griffiths, D.V. and Lane, P.A. 1999. Slope stability analysis by finite elements. *Geotechnique,* Vol. 49, No.3, pp. 387-403.
19. Hammah, R.E., Yacoub, T.E., Corkum, B., Curran, J.H. 2005. A comparison of finite element slope stability analysis with conventional limit-equilibrium investigation. In *Proceedings of the 58th Canadian Geotechnical and 6th Joint IAH-CNC and CGS Groundwater Specialty Conferences - GeoSask 2005*, Saskatoon, Canada.
20. Itasca. 2002. FLAC/Slope: user's guide.
21. PLAXIS BV. 2004. PLAXIS 2D Version 8. eds. R.B.J. Brinkgreve, W. Broere, and D. Waterman
22. Rocscience Inc. 2005. Phase^2 v6.0 – a two-dimensional finite element analysis program.