2009 Questions of the Month

****
The January Question of the Month
****

What is the difference between "model verification" and "model validation"?

Answer
******

All those who answered this question sent me a nice and correct explanation for the difference. I will choose the explanation found in the AIAA Guide G-077-1998:

"Verification is the process of determining if a computational simulation accurately represents the conceptual model; but no claim is made of the relationship of the simulation to the real world. Validation is the process of determining if a computational simulation represents the real world. Verification determines whether the problem has been solved correctly, whereas validation determines whether the correct problem has been solved."

I will make one "disclaimer" comment. This is all a matter of jargon. Saying "validation" and meaning "verification" or vice versa is not a "crime", since after all "validation" and "verification" are words that are used in every-day life and their regular meaning is known by everybody. But it has become a tradition in CM to distinguish between the meanings of these two words according to the definitions above.

Answered correctly (in alphabetical order):
****

Orna Agmon Ben-Yehuda, Meir Feder, Amiel Herszage, Yoav Ophir, Alex Rashkovan, Asher Yahalom, Zvi Zaphir.

****
The February Question of the Month:
****

Imagine that you are the head of a Computation Team in an important company. You have the following conversation with one of your employees, named Rosh Katan ("Roshka"):

Roshka: Do you remember that you asked me to compare two codes, code A and code B, that are based on two different methods, so that we can decide which one of them to adopt for the big project?

You: Yes, Roshka, but I asked you this two months ago, and I was starting to

despair...

Roshka: So I did the comparison. I took the benchmark problem that we usually take and ran codes A and B for it. I took exactly the same model and mesh for both codes.

You: Ok, so what were the results?

Roshka: Code A ran 3 times faster than code B.

You: Aha.

Roshka: Yes, but on the other hand code A gave 15% maximum error and code B gave 8% maximum error.

You: I see.

Roshka: So I wanted to ask you: what is more important for us - that the code have more accuracy or more speed?

You: This is like asking if an alligator is longer or greener.

Roshka: ???

You: Ok, listen, I will tell you what you need to do before we can decide which code to take. You should...

What were the instructions that you gave Roshka?

****
Answer:
****

Before discussing the answer I will mention that a few readers "complained" that the question was too "open". I admit guilty. Indeed, one can construct a whole course discussing the answer to this question. There are many issues involved, and the answer very much depends on the specific scenario and goals and the type of analyses under consideration. For example, Roshka talks about the "maximum error" and we do not know what kind of error this is. Is this a good measure of the quality of the code or not? We do not know. There are many more issues involved, and those raised below are very far from being inclusive.

Having made this disclaimer, I will first explain the main issues that I wanted to raise here. First, there is Roshka's report that for the single mesh that he took code A was faster than code B but code B was more accurate. To this "you" have commented "This is like asking if an alligator is longer or greener." I am sure that everybody sees that this comparison is almost meaningless. (I thank RBZ who pointed out that an alligator is greener, because it is green in both length and width, whereas it is long

only in length.)

It makes much more sense to compare the speed of the two codes PER THE SAME ACCURACY, or compare their accuracy PER THE SAME SPEED. This is called "cost-effectiveness". It would be much better if Roshka could say, for example: "I managed to get solutions from codes A and B for the benchmark problem with the same accuracy - 8% maximum error for both - and code A was 2 times faster." If that was the case, there would be reason to favor code A (if we ignore all other considerations not related to cost-effectiveness).

The reason that Roshka was not in a position to do such comparison is that he took a single mesh, namely he based all his conclusions on a single run. This is not a good idea from a number of reasons. In addition to the issue of cost-effectiveness, a single mesh cannot provide any information on the rate of convergence - another important parameter in the comparison of the two codes.

One thing that "you" can tell Roshka to do is to take a sequence of meshes with various densities - from a crude mesh to a very fine mesh, and use the benchmark problem to calculate the errors generated by codes A and B for each mesh. If there are N different mesh densities, Roshka would do 2N runs (N runs with each code), and would measure the error and computing-time for each run. Then Roshka can draw a graph, whose coordinates are the error and the computing time. Each of the 2N runs would correspond to a single point on this graph. By connecting the points, Roshka can draw two accuracy-vs.-speed lines: one for code A and one for code B. From this plot you can easily extract the relative cost effectiveness of the two codes. (For example, you can look at the error generated by the two codes for a run-time of 3 hours.) You can also extract the convergence rate from this plot.

Some readers remarked that often, from a practical perspective, cost effectiveness of a code is not so important relative to other considerations. For example, if we have to choose one new code to buy out of two candidate codes, and if the run times for typical models for the kind of applications we have are not very long, then the user-friendliness and amount of person-hours to spend on learning the new code and using it routinely may be key issues to consider. However, if a typical run takes a whole day (like in some complicated CFD applications) and if Roshka is already familiar with the two codes and the matter of user-friendliness is not relevant, then cost-effectiveness would certainly be a very important issue.

All the answers that I have received were very good, but I choose to end by quoting the answer of Orna Agmon Ben-Yehuda:

"Dear Roshka, please run convergence checks on the two codes, on different mesh sizes. For each mesh size, record the time and the maximal error. Compare the codes on the level of either time or error, and not on the level of the mesh, as the algorithms might be of even different orders.

If the client has a specification of a required error, compare the time and memory complexity for the codes to reach that required error, because this is the most interesting working point.

Otherwise, if the code is intended for a long term project, then we anticipate that the rate of convergence will be more important, in the future, than the current coefficients at any nowadays working point. Therefore, compare the convergence rates of the codes, and not any specific working point.

If the client poses not current specifications and has no long term intentions, name the client Rosh Katan instead of you."

Correct answers were received from (in alphabetical order):
****

Orna Agmon Ben-Yehuda, Rami Ben-Zvi, Jonathan Tal, Pavel Trapper, Asher Yahalom, Zvi Zaphir.

****
The March Question of the Month:
****

Everybody knows what programming is (writing computer codes). But what are linear programming and nonlinear programming?

****
Answer:
****

Linear Programming and Nonlinear Programming are terms which relate to the solution of constrained optimization problems. Historically, even before the times of computers, constrained optimization problems were called "Programs". A Program is a problem of finding a maximum/minimum of a function (called objective function or cost function) of a few variables (which are the unknowns, called design variables) subject to some constraints given on these variables. A "Linear Program" is a Program in which the cost function and all the constraint functions are linear. A "Nonlinear Program" is a Program in which at least one function (the cost function and/or one or more of the constraints) are nonlinear. Solving a Linear Program is called Linear Programming, and solving a Nonlinear Program is called Nonlinear Programming.

There is a very famous method for Linear Programming. It is called Simplex, and was invented by George Danzig. The first paper on Simplex was published in 1947, when computers were only in their infancy. The historical development of optimization is very interesting; those of you who are interested are encouraged to read Danzig's delightful historical account in

http://www2.informs.org/History/dantzig/LinearProgramming_article.pdf .

The word "programming" received the meaning that we use today ("writing computer codes") only later, mid 1950's probably, when computers became more widely available. This word in the context of Non/Linear Programming came originally from the language of logistics planning. Interestingly, the Open University is using the course title Tikhnun Lineari and not Tikhnut Lineari in their Hebrew publications.

Correct answers were received from Eli Boichis and Asher Yahalom; interesting historical notes were gratefully received from Amiel Herszage and Moshe Fuchs.

****
The April Question of the Month:
****

Many computational methods lead to a linear algebraic problem, where a system of linear algebraic equations have to be solved. The matrix appearing in this system must not be singular, otherwise the system cannot be solved. However, sometimes the matrix is not singular but "almost singular", and in such cases the solution of the system may become problematic, and special care must be taken.

The question arises: how do we measure the closeness of a matrix to being singular? Roshka has an idea: he calculates and looks at the determinant of the matrix. Since the determinant of a singular matrix is zero, the determinant might be regarded as a measure of the closeness to singularity; the smaller the determinant is (in absolute value) the closer the matrix is to being singular.

The question is: Why is Roshka's idea a bad idea? And what better idea would you suggest to measure how close a matrix is to being singular?

****
Answer:
****

Using the determinant to measure the closeness of a matrix to being singular is a very bad idea from several reasons. First, the determinant depends on the size of the entries of the matrix in a misleading way. Here is a demonstration of this. Consider a NxN diagonal matrix with entries equal to 0.1 on the diagonal. This matrix is as "perfect" as the identity matrix, and is not close to being singular at all. However, its determinant is D=0.1^N (^ meaning bekhezka). If N is a large number, D will be a small number. For example, if N is a million (which is not so rare for CM applications), D will be 10^(-6), and from looking at this small number one might wrongly conclude that the matrix is close to singular. Moreover, one would conclude that a matrix like that with N=million is closer to being singular that such

a matrix with N=1 which is of course nonsense.

Related to this, the determinant D is a bad measure also because it depends on the engineering units that we use to write down the entries of the matrix. Suppose, for example, that the matrix entries have units of length. Then we will get two very different results for D if we use meters of millimeters as our units!

The standard way to measure "closeness to singularity" is through the Condition Number of the matrix. The condition number can be computed as the ratio of the maximum singular value of the matrix to the minimum singular value. (The "singular values" of a matrix are real positive values related to the Singular Value Decomposition (SVD) of the matrix; see books on numerical linear algebra.) For matrices with simple structure (called "normal matrices"), such as real symmetric matrices, the notions of singular values and eigenvalues coincide (up to a sign), and then the condition number is the ratio of the largest eigenvalue and smallest eigenvalue (in absolute values).

The condition number really measures how the solution of the system Ax=b changes when we slightly change the matrix A (or the vector b). Thus it is a measure of the sensitivity of the matrix. A singular matrix has infinite sensitivity, and an infinite condition number. A perfect matrix has condition number 1.

All readers that sent me their answers provided nice explanations and examples. Thanks a lot!

Answered correctly (in alphabetical order):
****
Rami Ben-Zvi, Hillel Tal-Ezer, Amiel Herszage, Pavel Trapper, Eli Turkel, Kosta Volokh, Asher Yahalom, Zvi Zaphir.

****
Comments to the Answer to the April Question of the Month
****

1. Achi Brandt comments that although my answer is indeed the standard one, it is not the best answer. The condition number may grow arbitrarily upon rescaling of equations and unknowns. Besides, it is not a good measure of how easy it is to numerically invert a matrix accurately; for example a nonsingular diagonal matrix can have any condition number, but is trivial to invert or to solve with. The best answer is that the "closeness to singularity" is measured not by the condition number of the matrix itself, but by the the condition number of another matrix, called the "bi-normalized matrix". See details in the 2004 paper of Oren Livne and Gene Golub (http://ruready.utah.edu/archive/papers/bin.pdf ; and see the interesting dedication!). I deeply thank Achi and Oren for their eye opening remarks.

2. Asher Yahalom spotted a mistake in the algebra of my example. I wrote

D=0.1^N and then wrote that if N was a million, D would be 10^(-6). This is of course wrong. If N=6, _then_ D would be 10^(-6). With N=10^6, D would be a much tinier number (which actually emphasizes even more the point that the determinant is a bad measure). Thanks, Asher!

****
The May Question of the Month:
****

We are doing time-dependent analysis of dynamic heat flow in a certain three-dimensional object. Our analysis code is known to be very reliable, the method it is based on is convergent, and we are smart users. Suppose we run the code with a grid/mesh and time-step size of our choice. We get a result. Now, are we guaranteed to get an improved result if we decrease the size of the time-step by two (not changing anything else)?

******
Answer
******

Before relating specifically to this question, we'll make a couple of general observations.

First, in problems that depend on space and time there are usually two main discretization parameters: the time-step size k=(delta t) and the grid/mesh parameter h (which is the size of the element in finite element methods, the grid spacing in finite difference methods, the cell size in finite volume methods). The computed solution will converge to the exact solution only if _both_ k and h approach zero. If we fix h (fix the mesh) and then reduce k more and more we will not converge to the exact solution of the problem. We will converge to another solution, which is the space-discrete solution.

Second, it makes sense that some sort of "proportion" should be maintained between the two discretization parameters k and h when refining them. (Think of FEM in space, where you know that you need to have well-proportioned elements, namely elements with aspect ratio close to 1; the situation in space-time is somewhat similar.) Indeed, it can be shown that the appropriate "proportion" that one should try to maintain is the following. For wave problems one should keep h/(c k) approximately constant when refining, where c is the medium's wave speed. So if you decrease k by 2 you should better also decrease h by 2. For diffusion problem it is the ratio h^2/(alpha k) that should remain constant, where alpha is the diffusivity. So if you decrease k by 2 you should decrease h by 4!

But all this does not really answer the question. The above only means that it is not a good idea to fix h and reduce k more and more. But it does not necessarily mean that if you fix h and reduce k one time by a factor of 2 you might create any damage. The question is: might the overall error increase when we fix h and reduce k by a factor of 2?

The answer is yes. There is a specific explanation for this (see below), but even without it, we can realize that this is true based on the following argument. The total error is roughly the sum of the space-discretization error and the time-discretization error. There is no reason that these two errors will have the same sign at all points in space and time. Thus it is quite possible that they will partly cancel each other. Therefore it might happen that for given h and k we get good error cancellation, whereas if we reduce k by a factor of 2, we get less error cancellation. (This argument is similar to the one we gave in a previous Question, about numerical integration.)

The more specific explanation is this. The computed solution includes components of all the eigen-modes of the system. It is well-known that the low-frequency modes are represented well by the discretization whereas the high-frequency modes are not well represented, and might introduce "noise" into the solution. Therefore, it is one of the goals of a good time-stepping method to damp out the high modes (and only them!), so as to reduce this "noise". However, typically the ability of the time-stepping method to damp out the high-frequency modes goes down when k is decreased! Thus, it may happen that for given h and k we have good damping of the high modes, whereas if we decrease k by a factor of 2 we get less damping of high modes and thus larger overall error. This effect, albeit only for wave problems, was discussed and demonstrated in an article by Eran Grosu and Isaac Harari in the IACMM Alon No. 15 (see http://www.iacmm.org.il/NEWSLETTER/newsletter15.pdf).

For heat flow and diffusion there is a similar effect, and it manifests itself in the appearance of oscillations in the solution when, for a given h, the time-step size k is set below a certain value. See, for example, the paper by H.R. Thomas and Z. Zhou, "Minimum Time-step Size for Diffusion Problems in FEM Analysis", Int. J. Numer. Meth. in Engrg., Vol. 40, pp. 3865-3880, 1997. (Thanks a lot to Orna Agmon Ben-Yehuda for brining this reference into my attention!)

Correct Answers were received from (in alphabetical order):
****
Orna Agmon Ben-Yehuda, Amiel Herszage, Evgeny Shavelson, Asher Yahalom, Zvi Zaphir.

Comment on the May Question of the Month
****

Regarding this question and answer (about possible negative effects of reducing the time-step size in dynamic heat flow analysis), Isaac Harari remarks that in the CM literature there exist early references for small time step pathologies related to diffusion. The earliest reference that Isaac is aware of is:

Fujii H., "Some remarks on finite element analysis of time-dependent field

problems. In Theory and Practice in Finite Element Structural Analysis," Yamada Y, Gallagher RH (eds). University of Tokyo Press: Tokyo, 1973; 91–106.

****
The June Question of the Month
****

If we assume that the rate in which computers have developed will continue forever, in what year shall we be able to solve a "full" linear algebraic system of a million equations with a million unknowns in one second? Will we live to see this?

******
Answer
******

In order to answer this question we should first consider how many operations are needed to solve a "full" linear algebraic system of a million equations with a million unknowns. This depends on the algorithm used, but if we use the most straight-forward method, namely Gauss elimination without any "tricks", then the answer is that for a system of N equations and N unknowns the number of operations will be of the order of N^3. Thus, for a system with 10^6 equations we will need to do an order of 10^18 operations, or 10^12 Mflop (Mega floating-point operations).

So the question is now: in what year will a computer be able to do 10^12 Mflop per second, namely 10^12 Mflops? Computer speed has evolved (at least until a few years ago) according to Moore's law, which says that the speed of computers is approximately doubled every 18 months (some say every 2 years). See, for example, the graph in

http://books.google.com/books?id=waOBNNNXU28C&pg=PA5&lpg=PA5&dq=fish+belytsc
hko+moore%27s+law&source=bl&ots=eIrZUyoZI0&sig=YvgLtK0bCgtNGsnLjpfSHjLUPpM&h
l=en&ei=7DxfSuH-Ls-K-QawueHmAQ&sa=X&oi=book_result&ct=result&resnum=1

which is taken from the book "A First Course in Finite Elements" by J. Fish and T. Belytschko. From this graph, in 1998 computer speed was about 10^6 Mflops. A simple calculation shows that, according to Moore's law, the year when computer speed will be 10^12 Mflops is 2027.

So the answer seems to be 2027. And yes, all of us will surely live to see it! (Ashrey Hama'amin.)

However, Orna Agmon Ben Yehuda made the following important comment: "Speed has stopped evolving. It is only parallelization level which increases...

So, an answerable question would be something like - what level of parallelization is needed to get an algorithm for the problem (and what algorithm) to run under 1 sec? Matrix multiplication is a topic well researched in parallelization... It (also) becomes a question of money - if you have a supercomputer, you can probably do it today (minus data transfer)... Anyway, I would not use Moore's law here."

In short, it seems that actually one should not use Moore's law anymore.

Correct answer was received from Orna only. **********

****
Comments on the June Question of the Month
****

I have received two additional interesting and important comments on the June question, related to the use of Moore's law to predict the year when it will become possible to solve a "full" linear algebraic system with one million unknowns in one second.

A. Zvi Zaphir rightly comments that it makes much more sense to solve such large full systems with an iterative method rather than with a direct method like Gauss elimination. Now, when one uses an iterative method the computing time strongly depends on the accuracy required (because this will determine the number of iterations). If the accuracy required is not very high and one uses a parallel computer, the goal in the question may be achieved much earlier than 2027 (which was my answer); maybe it can be achieved even today.

B. Micha Wolfshtein has made some observations which I find illuminating on Moore's law, its origin and the question of whether it will continue to be valid. Here are the main points (with my slight editing):

1. Moore's law was based on the assumption that miniaturization will allow more transistors to be compacted into a chip of a given area. This was supposed to increase speed and density and reduce power consumption. Moore hypothesized that the number of transistors per unit area will double every 18 months. This was the basis for the Moore's law. It worked nicely for some 20 years and more, as the width of the chips reduced and reduced.

2. It is clear that Moore's law cannot be valid forever. At one point in time it must cease to hold. From physical consideration the process of reduction will stop when the width of the chip will approach molecular thickness.

3. New technologies appear today, at least on the conceptual level (e.g. quantum physics and DNA technologies) which have the potential of increasing the speed of processors and reducing their sizes. The question of how close

such developments are to practical implementation is under debate, but Some physicists claim that they will become possible in the not too far future. Will they obey Moore's law? This is difficult to say. But the point remains that processor technology appears to still have a potential for significant improvements, even in our life time.

4. One might claim that even with a given transistor density, using a parallel computer with an arbitrarily large number of nodes makes it possible to solve any problem arbitrarily fast. This claim is not correct in general. For loosely coupled problems this is possible, since the computing power is nearly proportional to the number of nodes. Unfortunately inversion of dense matrices is not a loosely coupled problem, and the parallellization efficiency deteriorates with the number of nodes, as least with the standard algorithms. The solution time for this problem depends on the speed of internal communication in the computer. The situation becomes even more confusing with the new developments in GRID and CLOUD computing, where the number of processors is staggeringly large, but the communication speed is relatively slow.

5. From the points above, it is clear that predicting the future of computer development is very difficult. Most probably improvement in computer speed will continue, one way or another. Whether it will be Moore's law or another law that will describe this future development remains to be seen.

****
The July Question of the Month
****

This time it is a history question. What was the first paper on the use of - what we call today - the finite element method (FEM) in two dimensions, based on potential-energy minimization, using a mesh with triangular finite elements and linear shape functions? What is so amazing about this paper, in addition to the fact that FEM was actually invented in it?

******
Answer
******

The first paper that answers this description is "Variational methods for the solution of problems of equilibrium and vibrations", by R. Courant, Bulletin of the American Mathematics Society, Volume 49, Number 1, pp. 1-23, 1943.

Richard Courant was a famous applied mathematician. You can see nice pictures of him in http://www-history.mcs.st-and.ac.uk/PictDisplay/Courant.html and read a short biography of him in http://www-history.mcs.st-and.ac.uk/Biographies/Courant.html . The well known Courant Institute of Mathematical Sciences in New York is called after

him.

I received two excellent answers that provided some interesting comments. MB mentions the fact that the FE method was (re)invented by engineers and became very popular in the 1950's; the name Finite Element was coined by Clough from Berkeley in 1956. Everybody back then was totally unaware of the paper by Courant. Only later it was realized (the first to note that was the mathematician G. Strang) that Courant actually invented FEM in 1943 with the paper above, and even applied it to two-dimensional problems of torsion, using a mesh with triangular finite elements and linear shape functions. MB further gives us a real treat: "The paper can be found at : http://projecteuclid.org/DPubS?verb=Display&version=1.0&service=UI&handle=euclid.bams/1183504922&page=record (press the pdf link). I urge everyone to read it, it is a pure intellectual feast and any one with a BSc in Engineering will appreciate it."

AH is also full of awe from this paper and has a similar opinion to MB. He makes the side remark, that the specific torsion example chosen by Courant to demonstrate the method (see the Appendix in Courant's paper) does not reveal the true power of the method, because, as any structural engineer knows, the shear flow in the the closed thin-walled cross section of the bar that he took is almost constant. It is not surprising, therefore, that no extra accuracy is obtained by using more than two elements in the quad (1/8 of the cross section). Had he chosen an open cross sections, things would get much more "interesting" numerically.

I asked: "What is so amazing about this paper, in addition to the fact that FEM was actually invented in it?" The answer is that back in 1943 when Courant wrote this paper, there were no computers available. It is true that there were some "calculators" (developed during the 2nd world war), but there was essentially nothing that could be programmed. I think that this is the main reason that Courant's paper was forgotten ("ha, yet another totally useless and impractical idea"), and that FEM had to be re-invented 15 years later by the engineers.

Correct answers were received from: Michel Bercovier, Amiel Herszage.
****

****
Comments on the July Question of the Month
****

The answer to the July Question included the comment that one of the amazing aspects of the Courant paper inventing FEM was that back in 1943 when Courant wrote this paper, there were no computers available. Eli Turkel comments that the same holds for another discovery connected to Courant. This is the famous CFL condition of numerical stability of finite difference schemes, named after Courant, Friedrichs and Lewy. It was published in 1928, even before the Courant FEM paper, and much before the digital computer era. E.T. also recommends reading the delightful account on Friedrichs' life

(with touches on Courant as well) at
http://www.friedrichs.us/history-KOF-life-by-C-Reid.html . I join this recommendation; the account includes some real "pearls".

****
The August Question of the Month
****

This time the Question required a little knowledge of mathematics (but only a little!).

We all know how important it is to check (verify) our codes against some known solutions, and how difficult it is to get reference solutions for some problems, like those governed by the nonlinear Navier-Stokes equations. Imagine that you are the head of a CFD group in an industrial company, and you have the following discussion with your employee Rosh Katan (Roshka).

Roshka: You remember that you asked me to find in the literature a numerical or experimental 2D reference solution to the incompressible Navier-Stokes equations, so that we can check our code with it and compare results?

You: Yes, of course. I started to give up on this because you took this long vacation...

Roshka: So I think I don't need to look in the literature, because I know how to find an exact analytic solution!

You: A 2D exact analytic solution to the nonlinear Navier Stokes equations?? This doesn't sound so likely...

Roshka: Look at my method. The code solves the two-dimensional incompressible steady-state Navier-Stokes equations. So we have three unknowns: the velocities u and v and the pressure p. And we have three differential equations. The first two are the momentum equations that look like this:

$$N[u,v] + \text{Grad } p = 0 \qquad (1,2)$$

where N[u,v] is a complicated nonlinear expression depending on the velocities u and v, and (Grad p) is the pressure gradient. The third equation is the incompressibility constraint:

$$du/dx + dv/dy = 0. \qquad (3)$$

You: So far so good, Roshka.

Roshka: In addition we have boundary conditions for u and v. Now, I played a little and finally found some simple functions u and v that satisfy the boundary conditions and also the incompressibility constraint (3)!

You: That's very nice, but I don't think it helps us. You also have to satisfy the two momentum equations, (1,2), and I don't believe you can satisfy them with the simple functions you picked up.

Roshka: Yes I can! Let's define the vector G as G=Grad p . Then (1,2) give us simply

$$G = -N[u,v]. \qquad (4)$$

So all I have to do is to plug in the u and v that I picked into (4), calculate N[u,v], and this will give me G. In this way I satisfy all the equations! I can then run the code, and compare the results for (u,v) that I get from the code with the functions (u,v) that I took. It's very simple!

You: Yes, it is very simple, but unfortunately it is also wrong.

Why do you say that it is wrong?

******
Answer
******

Roshka chooses some functions u and v, and then calculates the vector G, which is supposed to be the gradient of p. However, it is important to realize that not every vector is a gradient of something! So, it is very possible that the G that Roshka calculates is not a gradient of any scalar function, in which case the pressure field p that corresponds to Roshka's solution does not exist at all!

Thus, there is one more equation that must be satisfied and that Roshka did not check at all. This is the "compatibility equation" that says that the vector G is a gradient of some function (the pressure). This can either be written directly as G = Grad(p), in which case p is an additional unknown that must be found, or in the form of the constraint Curl(G)=0. The vector G is a gradient of something if and only if Curl(G)=0. In two dimensions, the constraint Curl(G)=0 means $dG_x/dy=dG_y/dx$. [We can see what this means. If we plug G=Grad(p) into this we get the equation $d^2p/(dx\,dy) = d^2p/(dy\,dx)$ which simply means that the order of differentiation of p with respect to x and y is not important.]

To summarize, if Roshka chooses some simple-minded functions u and v, most chances are that the vector G that he obtains does not satisfy Curl(G)=0 and hence it is not a gradient of any scalar function, and therefore the pressure field p does not exist, which means that Roshka's solution is wrong.

I have received a number of interesting remarks on this question and solution. Both MW and RBZ comment that reference solutions for the Navier-Stokes (NS) equations can be obtained by a number of ways, e.g.: (a)

Choose a 1D problem as your test case (e.g. Poiseuille or Couette flow), and set the boundaries at an angle to the coordinates. (b) Add a general source term to the NS equations (and to the code that solves them) and "design" the source term so that the equations are satisfied for your chosen (u,v,p) solution. This would give you a non-physical solution, yet a useful solution for code verification. (c) Take a potential-flow solution. Any potential solution satisfies the NS equations, and thus would lead to a vector G that _is_ the gradient of the pressure. [In fact, remarks MW, maybe this is what Roshka has done to begin with, in which case he should be promoted rather than fired!]

RBZ also points to Roache's (similar to Roshka...) Method of Manufactured Solution, which produces exact solutions for the NS equations. This method is described in the report by K. Salari and P. Knupp, "Code Verification by the Method of Manufactured Solutions", SAND2000-1444, 2000 (available at http://prod.sandia.gov/techlib/access-control.cgi/2000/001444.pdf).

MW also comments that in fact what Roshka has done is the first step in a known numerical iterative procedure to solve the problem. This procedure, called SIMPLE, attempts to find the vector G which both leads to a correct pressure field and satisfies the momentum equations, in an iterative manner.

In ZZ's message, the analogy of this problem with that of linear elasticity was raised. If we find an elastic stress field that satisfies the equilibrium equations and the boundary conditions, this does not mean that we have found a correct solution. From the stresses we can calculate the strains (using generalized Hooke's law), but in order for the displacement field to exist, the strains must satisfy the compatibility equations. The elastic compatibility equations are analogous to the constraint Curl(G)=0 in the case of NS, and the strain-displacement relations are analogous to the equation G=grad(p) in our problem.

Correct solutions were obtained by (in alphabetical order):
****

Rami Ben-Zvi, Simon Brandon, Amiel Herszage, Micha Wolfshtein, Asher Yahalom, Zvi Zaphir.

****
The September Question of the Month
****

In recent years, the class of methods called Discontinuous Galerkin (DG) has emerged and became popular among those developing new Finite Element (FE) techniques. One of the properties of DG is that the solution (elastic displacement, temperature, etc.) is not required to be continuous across element boundaries, and so may have "jumps" on the interfaces between neighboring elements. This is so even though the exact solution is known to be continuous. In contrast, the standard FE method produces numerical solutions which are continuous by construction. (Of course, the derivatives

of the primary solutions - stresses, heat fluxes, etc. - are discontinuous even in the standard FE method. But here we are talking about the primary solution - elastic displacement, temperature, etc.)

This property of the DG may look strange - it may seem non-beneficial to allow the approximate solution to be discontinuous when we know that the exact solution is continuous. What is the explanation and motivation for this?

******
Answer
******

To fix ideas, let us consider the problem of steady-state heat conduction. (Analogous arguments can be given to problems in elasticity.) Suppose we have a finite domain (the body) in which the steady-state heat equation holds. We assume that along a part of the boundary of the body the temperature is given (a Dirichlet boundary condition) whereas on another part of the boundary the normal heat flux is given (a Neumann boundary condition). Now, let us construct a mesh of finite elements in this domain.

What is our goal? We want to find a numerical solution that has the following 5 properties:

A. In each element, the solution satisfies (exactly or approximately) the equation of steady-state heat conduction.

B. On the part of the external boundary where the temperature is given, the solution satisfies (exactly or approximately) this boundary condition.

C. On the part of the external boundary where the normal heat flux is given, the solution satisfies (exactly or approximately) this boundary condition.

D. The temperature (namely the solution itself) should be (exactly or approximately) continuous across element borders.

E. The normal heat flux (related to the solution gradient) should be (exactly or approximately) continuous across element borders.

Obviously, no numerical method would generally satisfy all five requirements exactly, because this would mean that we have found the analytical solution of the problem, and thus there is no need for a numerical method. So in general, at least some of the five requirements are to be satisfied approximately. The standard finite element (FE) method satisfies A, C and E approximately and B and D exactly. More precisely (and we will not discuss this issue here, which has to do with variational formulations) we say that A, C and E are satisfied weakly (or "in a weak sense") while B and D are satisfied strongly (or "in a strong sense").

There are special FE methods in which the type of satisfaction of A-E is

different. For example, FE methods in which A (the differential equation in each element) is satisfied exactly are called Trefftz methods.

The _balance_ of the satisfaction of all 5 requirements - some of them weakly and some of them strongly - has a major effect on the quality and behavior of the numerical method. If one is not careful and tries to enforce too many requirements strongly this may result in a bad numerical method that yields poor numerical solutions! One way in which this is manifested is the phenomenon known as "locking" that some of you may be familiar with. It is maybe surprising at first, but can be understood after some reflection, that a strong satisfaction of a certain requirement (from A-E) is not necessarily better than a weak satisfaction of this requirement.

Now we come to Discontinuous Galerkin (DG) methods. In these methods both D and E are satisfied weakly. Thus, not only the normal heat flux but also the temperature is not continuous across element borders. The continuity of both temperature and normal heat flux is enforced in a weak sense. Does this look strange? Yes - if you are very accustomed to the standard FE method. But if you think about this a little, there is nothing particularly Kadosh about property D that makes it more important than all other properties. Enforcing it weakly is not more "strange" than enforcing any other property weakly.

It turns out that DG methods are associated with particularly good balance among the five requirements, and as a result they tend to be more well behaved than the standard FE method in some situations.

All this is just "hand waving" and general talk. Here are some more concrete details on the advantages of DG methods. RBZ quotes from Bernardo Cockburn's lecture notes on DG for convection-dominated problems (http://www.math.umn.edu/~cockburn/LectureNotes.html):

/ The main features that make the methods attractive are: /

* /their formal high-order accuracy, /
* /their nonlinear stability, /
* /their high parallelizability, /
* /their ability to handle complicated geometries /(e.g., hanging nodes in general and within hp-adaptivity)/, and /
* /their ability to capture the discontinuities or strong gradients of the exact solution without producing spurious oscillations. /

There is a lot to explain about each of these properties, but we shall stop here. PT points out another important advantage of taking the primary field (temperature) to be discontinuous. In some cases this allows one to invert some matrices on the element level rather than on the global level, which saves a lot of computational effort.

Correct answers were received from:
****
Rami Ben-Zvi, Pavel Trapper.

****
The October Question of the Month
****

Here is another episode in the beloved sequence "Roshka's Follies".

Imagine the following conversation between you and your employee Rosh Katan ("Roshka").

Roshka: About this wave problem that you gave me to solve, related to the new Project...

You: Yes, how about it?

Roshka: It goes very well. Remember, you told me to solve it for many various wave-lengths, the smallest being 0.001 meter and the largest being 1 meter.

You: Right.

Roshka: So I started with the 1 meter wave-length. I created a mesh with 20 elements per wave-length, because I read in an old report that there is a rule of thumb saying that 10 elements per wave-length should be good enough, and I wanted to be safe.

You: But you should check that...

Roshka: Not to worry! You remember that for the 1 meter wave-length (and for this wave-length only) we have some experimental results? Well, I compared the numerical results to the experimental results and the agreement is excellent!

You: Very good.

Roshka: So in principle this is the end of the story. I will go on and solve the problem for all the wave-lengths from 1 m down to 0.001 m, and in each case I will take 20 elements per wave-length. The mesh for the smallest wave-lengths will be quite dense, but it's not too terrible. I'll finish on Monday and then leave for my ski vacation.

You: Yet another vacation?! You just returned from one! Anyway, I hate to disappoint you, Roshka, but what you suggest to do will most probably _not_ give us good results.

Explain why you said this to Roshka.

******
Answer
******

What I had in mind is the so-called "pollution effect" that is caused by dispersion error. It turns out that keeping a fixed number of elements (or grid points) per wave length is not a safe procedure in general, even for simple linear wave problems. The rule of thumb saying that one needs about 10 elements per wave length is good for relatively long waves, but as the waves get shorter (or their frequency increases) one needs more and more elements per wave length to maintain the same level of accuracy!

Eli Turkel, together with his co-workers, was one of the pioneers to discover and research this phenomenon. In his 1985 paper with Bayliss and Goldstein (J. Computational Physics, Vol. 59, pp. 396-404, 1985) they show that with a 2nd-order-accurate numerical method the number of required grid points per wave length lambda increases like 1/sqrt(lambda). Moreover, they show in this paper that the pollution becomes smaller for higher-order methods, and that with a really high-order method it becomes negligible. Thus, ET comments, "for a spectral scheme Roshka is in fact correct!"

Later, Babuska and his coworkers analyzed this phenomenon in more detail and coined the name "pollution effect" to it. ET comments: "In fact (this effect) goes back to a much earlier paper of Kreiss and Oliger who showed that (in the time domain) one needs more points per wave length - again as a function of the accuracy of the scheme - for longer periods of time, because of phase errors. In frequency space this is equivalent to the pollution effect."

Finally, ET notes: "Despite that these facts are known for over 20 years, most papers when doing a convergence study assume that the number of points per wavelength is fixed for various wave lengths. So Roshka is in good company."

RG in his answer relates in detail to exactly the same effect, and explains what Roshka needs to do: "In order to achieve the same global error (for all wave lengths) as for the 1m wave length, Roshka needs to set an appropriate resolution for each wave length that he solves so that the total accumulated error will be limited to that of the 1m problem. The resolution for the 0.001m wave length should be much higher in order to get the same global error."

AY and ZZ proposed two correct answers that were different from the above. AY points out that if the problem Roshka is dealing with is nonlinear, then complex wave phenomena are expected - for example, interaction among waves of different wave lengths - and there is no guarantee that the simple rule that Roshka adopted would be good enough. ZZ relates to the effect of damping: "With FEM, it is very difficult to determine the full damping matrix. Therefore it is usual to assume that the damping matrix is

proportional to the mass and stiffness (Rayleigh damping). As long as the frequencies are low and the damping is relatively small, this is fine. As the frequencies increase proportional damping is not valid anymore, and a more appropriate damping simplification is usually not available. Therefore high-frequency acoustic problems are usually solved by energy methods and not by FEM."

Correct answers were obtained by (in alphabetical order):
****

Ran Ganel, Eli Turkel, Asher Yahalom, Zvi Zaphir.

****
The November Question of the Month
****

Most of us agree that it is very important to be Sovlani (tolerant) to the Zulat who is different from us in religious faith, political tendency or computational preference. I dedicate the following Question to this value.

The two most popular general computational methods are the Finite Difference Method (FDM) and the Finite Element Method (FEM). List the N most important advantages of FDM over FEM, and the N most important advantages of FEM over FDM. The choice of N is up to you (N=1 is ok!), but a strict rule is that the number N of advantages that you count for FDM must be identical to the number N of advantages that you count for FEM.

******
Answer
******

Before providing an answer, here is an important comment made by Roland Glowinski: "Concerning CFD and many other applications, Finite Difference Methods have been replaced by Finite Volume Methods which contain FDM as particular cases. Actually, the situation is not clear cut since Discontinuous Galerkin Methods (which officially are FE methods) have a Finite Volume flavor." Thus, the borders between the various methods are not well-defined, and there is a whole "spectrum" of methods connecting FDM and FEM.

Having said this, here is an answer to the question with N=5, based on the collective answers of OABY, RBZ and AY, and my own contribution tossed in.

FDM advantages:

FD-1. Conceptual Simplicity. FEM does not even come close to the conceptual simplicity of FDM. It takes at most 30 minutes to explain the basics of FDM to someone who knows what a differential equation is, to the level of understanding exactly how to implement the method in a simple

setting. In contrast, a similar level of understanding of FEM takes a good number of hours and involves concepts that are much more difficult to grasp.

FD-2. Simplicity of Implementation. I will quote one of the answers: "FDM is very simple and straightforward for implementation on a regular domain (e.g., rectangular in 2D using a uniform grid), with simple boundary conditions." In contrast, FEM is not just more involved conceptually, but takes much more effort to implement. I know a number of cases where an applied mathematician or an engineer needed to solve a little "isolated" problem numerically, and simply implemented the appropriate FD scheme on the spot, from scratch, and had numerical results within two hours. I have never heard of anyone doing the same with FEM...

FD-3. Time-Dependent Problems. FDM is very appropriate for dynamic problems, and it deals with space and time discretization in a uniform and natural way. In contrast, the classical FEM is intended for space discretization only, while time discretization is done using FD. (It is true that space-time FE schemes are available too, but they are arguably less natural, and the fact is that they did not "catch" despite a research boom in the 1980s-90s. Most commercial FE codes use FD discretization in time to this day.)

FD-4. Explicit Schemes. In time-dependent problems, explicit FD schemes (namely schemes that do not require the solution of any system of algebraic equations) are available, and are based on direct application of the FD operators. On the other hand, if consistent FE discretization is used in space, no explicit schemes are possible unless "mass lumping" is performed. Lumping, which amounts to "diagonalizing" a non-diagonal matrix in (more or less) brute force, is not always a healthy and safe process (although there are certainly scenarios when it is safe), it introduces additional errors, and in general one should be careful when practicing it.

FD-5. Mathematical Analysis. The mathematical analysis of FDM (for error estimates, etc.) is classical, well-established and very powerful. It goes back to giants like Euler. On the other hand, the mathematical analysis of FEM is modern; it started with the work of G. Strang (Ad Me'a Ve'esrim) in the 1970's. This is not to say that the latter is inferior to the former (in fact see FEM advantages below), but it means that in the general tool-box of mathematical analysis for numerical schemes, the FDM tools are more well-established today. A manifestation of this is that in some cases, mathematical analysis of a FE scheme is performed by considering a regular grid, looking at the difference equations (!) emanating from the FEM scheme, and analyzing those.

FEM advantages:

FE-1. Geometrical Flexibility. FEM allows direct and straight-forward treatment of problems in domains with very complex geometries. In

contrast, FDM requires some special treatment in order to cope with irregular geometry.

FE-2. Local Mesh Refinement. FEM allows to easily have a refined discretization in a local region of the domain, where errors are expected to be higher, or where the geometrical of other features of the problem require a finer resolution. As a consequence, it also allows a straight-forward _adaptive_ mesh refinement. All this is much more difficult and cumbersome in FDM.

FE-3. Solution Defined Everywhere. In FEM, by definition of the method, the solution is defined everywhere in the domain, not just at the mesh nodes (= grid points). In contrast, in FDM the solution is only defined at the grid points. Thus, if one is interested in the solution between two grid points, in FDM this solution is calculated in an ad-hoc manner, whereas in FEM it is calculated consistently with the entire scheme.

FE-4. Concentrated Sources/Forces. FEM allows the direct and simple application of concentrated (point) sources and forces. A concentrated force always introduces a "singularity", but due to the fact that FEM is based on a variational formulation, the FE scheme can usually live in peace with such singularity without any need for special treatment. In contrast, concentrated sources/forces cannot be applied directly in a FDM scheme.

FE-5. Variational Framework. The fact that FEM is based on a variational formulation equips it with additional properties that FDM totally lacks. For example, FEM applied to self-adjoint problems leads to the beautiful "best-approximation property", and is associated with the minimization of an energy functional. Such properties provide important information on the FE solution behavior and error estimates.

Correct answers were obtained by:

Orna Agmon Ben-Yehuda, Rami Ben-Zvi, Asher Yahalom.
Comment: Roland Glowinski

Comments on the November Question of the Month

The November Question of the Month, that dealt with the relative advantages of the Finite Difference Method (FDM) and the Finite Element Method (FEM), has led to a number of interesting comments, which I briefly summarize here.

Asher Yahalom comments that my assertion that "FEM is based on a variational formulation" is not always true, since there are important cases (e.g., the Navier Stokes equations) where no extremum principle of a functional exists, but still FEM is very useful, and is then based on a

weak formulation rather than on a variational formulation. One can actually say that FEM is always based on a weak formulation. Strictly speaking this is perfectly right, but the fact is that in the jargon of FEM, people do refer to a "weak formulation" also as a "variational formulation". This jargon has become so popular and common, that now everybody in the FEM business considers the weak formulation to be a "variational formulation". I agree that strictly and historically speaking, there is a slight abuse of terminology here, since originally the term "variational" only referred to the existence of an associated functional and an extremum (or stationary-type) principle.

Micha Wolfshtein comments that there are important cases in which the main computational question is not about discretization using FDM or FEM, but about the algorithmic way to approach the whole problem on the continuous level. For instance, in analyzing time-dependent flows governed by the Navier Stokes equations, due to the mixed nature of the equations (e.g., parabolic-hyperbolic), one has to decide if and how to split the equations. (Most common solvers are based on time splitting methods, which often originate from the SIMPLE algorithm.) Once the algorithm for handling the equations is decided upon, one may use FEM, FDM, finite volumes, or other methods of discretization, but this is less crucial than the splitting algorithm itself.

I gave "Geometrical Flexibility" as the first advantage of FEM over FDM. While this is true, Eli Turkel comments that FDM/FVM are associated with another type of flexibility that FEM does not share. In FEM, everything is more or less "automatic" once a basis is chosen, while FDM/FVM gives much more freedom to design the method so that it fits the problem.

ET gives two specific examples related to this. First, to solve the Helmholtz equation in 2D with high order accuracy on a 3x3=9 point Cartesian grid, one can let the coefficients be general and get a 6th order method for constant coefficients and a 4th order accurate method for general variable coefficients. For a FEM this is probably not possible; higher-order FE schemes usually involve more points either interior or exterior to the mesh. ET adds, in the spirit of the question that required full balance between the FEM and FDM advantages: "Of course on the other hand Discontinuous Galerkin (DG) and spectral schemes allow much higher accuracies than normally achieved by FDM/FVM."

The second example is the solution of some PDEs in conservation form (e.g., advection-dominated advection-diffusion), where one generally needs an upwind scheme or else artificial viscosity. For FEM, upwind does not arise naturally but is forced on the solution. For FDM scheme it arises naturally from the theory; upwind is as standard as central difference! Historically, upwind schemes were developed for FDM and only much later were incorporated into FEM based on the FD analog. Similarly, DG schemes now use a Riemann problem solver which originated from FD schemes.

The December Question of the Month

The word "crime" appears in two contexts (at least) in CM. There is "variational crime" and there is "inverse crime". Explain (in general terms) these concepts or one of them of your choice.

Answer

Let's first talk about "variational crime", a term coined by Strang and Fix in their classical 1973 book on the mathematics of the Finite Element method (FEM). In the variational formulation of FEM we define a number of function spaces (= sets of functions = collections of functions). One of them, let's call it S, is the set the exact solution should belong to. Another one, let's call it S*, is the set in which we seek the FE solution. The set S* is supposed to be totally contained in the set S. Of course, S* is much "smaller" than the original set S. (Any function in S* can be expanded in terms of a finite number of basis functions - the FE shape functions, whereas functions in S generally cannot be expanded by this basis alone.) The basic idea of FEM (if we ignore many other properties and details) is to say: instead of looking for the solution in the huge set S, we will restrict ourselves to finding the solution in the much smaller set S*. Now, if we use FEM such that S* is _not_ totally contained within S, we say that we commit a "variational crime".

Here is an example. Consider a linear heat conduction problem in steady state. We are given a body, and suppose that we know the temperature distribution T=T0 on the surface of the body. We want to find the temperature distribution inside the body. The exact solution (the exact temperature field) is known to be a continuous function in the body. So the set S will be the set of all continuous functions which satisfy T=T0 on the surface of the body. (Actually I am simplifying the situation here a little bit for the sake of clarity.) Now, in the standard FEM, we generally enforce the condition T=T0 only at the _nodes_ on the surface of the body, not _everywhere_ on the surface! Thus, in general we allow the FE solution to violate the condition T=T0 and thus we commit a variational crime. Note that this crime is committed very routinely, every time that we run a FE code with an "essential" boundary condition such as T=T0.

In fact, in general we commit _two_ variational crimes in the example above. The requirement in S is that T=T0 be satisfied on the _true_ surface of the body. But in FEM we many times approximate the shape of the body by a simpler one; for example when we use the simplest linear FEs we approximate a curved boundary (like a circle) by a straight-edge boundary (like a polygon).

Other examples of variational crimes include, and I quote AY, "the FE

functions do not satisfy smoothness conditions on interelement boundaries, or numerical quadrature is employed".

Committing a variational crime means actually "breaking the rules" of the FE formulation. However, Strang and Fix have proved that many variational crimes (like in the examples above) are "legitimate", in that they lead to errors that are not larger than the main FE errors that we make anyway. But of course, when you "break the rules" you must do it in a certain way that will be legitimate (in the fashion of the Strang-Fix analysis). If you are not careful, "breaking the rules" may lead to non-sense results.

Now let's move to "inverse crime". This is a totally different concept, related to verification of codes that solve inverse problems. Here is an example of an inverse problem taken from the field of ultrasound testing of structures (or of the human body). We are given a structure, but we do not have complete information on what is inside it. For example, we suspect that there is a crack somewhere in the structure but we know nothing about the size, location and shape of this crack. Now we apply a certain load (or an incident wave) on the structure, and we make some measurements of the structure response. The inverse problem is then the following: Given certain measurements on the surface of the structure, find the size, location and shape of the crack.

Suppose we have written a code that solves such inverse problems, and we want to check it (namely to "verify" it). Here is a nice trick that allows us to do this. We take the model of the structure and introduce a certain crack in it, with size, location and shape as we like. Then we run the model by a standard analysis code (not the inverse code) and we find the structure's response. Next we take some of the results of this response and we use them as "measurement input" for our inverse code. We then run the inverse code, which is supposed to detect and identify the crack. If the inverse code manages to identify the crack that we have planted ourselves, then we call it a success.

This procedure is perfectly fine if one is careful. The warning "if one is careful" means the following. It has been found by "inverse problem" researchers, that if we use exactly the same model - with the same grid (mesh) and other computational parameters - for the analysis code that "produces" our measurements as for the inverse code, then the fact that the inverse code succeeds in finding the true crack does _not_ establish a verification of the code. In more general words, and I quote AY, "the act of employing the same model to generate, as well as to invert, synthetic data" is not regarded as safe verification, and is called an "inverse crime". Thus it is always recommended to use a much finer grid (mesh) for the code that generates the measurement than the grid used for the inverse code.

Correct answer was received from: Asher Yahalom.