Log of Questions of the Month for 2011

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

Suppose you peek above your friend's head and you see that she is doing an implicit Navier-Stokes simulation in time. You notice that she is taking an extremely large time-step, that cannot give any reasonable resolution in time. What can be the reason for this?

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

Your friend is interested in a steady state solution rather than in a transient one.

There are two main approaches in solving steady-state nonlinear flow problems. One approach is to attack directly the steady-state equations, and use some kind of iterations (typically Newton iterations) to approach the solution. The second approach, which in many CFD applications is quite popular, is to take the time-dependent flow equations, and step in time until steady-state is practically reached. If we take this approach, of course there is no point in getting good accuracy of the time evolution, since we are only interested in the steady-state solution, so we would typically use very large time steps, as your smart friend in the Question did.

Correct answers were given by:

Rami Ben-Zvi, Rafi Haftka, Eli Turkel.

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

Once again an episode from the Roshka series. Rosh Katan (Roshka) just returned from service in Australia, and is back working as your employee in the computational engineering company BATALA (Benchmarks And Theoretical Analysis for Low-tech Applications). One day he enters your office:

Roshka: Boss, you remember asking me to read about frictionless contact between an elastic body and a rigid body, and how to solve such problems with Finite Elements, and then to try to extend our code to handle such

problems?

You: I sure do.

Roshka: So I started to read about this stuff. I can't say I understood a lot. Something about Cohen-Yankel conditions...

You: You probably mean Kuhn-Tucker conditions.

Roshka: Whatever.

You: Well, you probably tried to read material which is too mathematical for you. I can give you some material which would be easier and more practical.

Roshka: I really don't think that I need this stuff. Tell me what is wrong with the following procedure. We have an elastic body which is statically loaded, it deforms, and then it touches a rigid obstacle, and we assume that there is no friction between them, right?

You: That's right.

Roshka: So this is what I do. I first run the code without the rigid obstacle at all. I get the deformation. Then I see that part of the deformed body "penetrates" into the obstacle, which is of course forbidden.

You: Right.

Roshka: So I find the nodes in the mesh of the body that penetrate to the obstacle. These nodes should be constrained so that upon touching the obstacle their normal displacement becomes fixed and "frozen", since they cannot penetrate into the obstacle, while they remain free to move in their tangential direction, since there is no friction. I can implement such a constraint in the code, and that would be the end of it!

You: What you suggest might actually work in some cases. But not in general, because you are forgetting something important.

What is it that Roshka forgot?

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

Roshka's suggested procedure makes the implicit assumption that "once in contact - always in contact". Namely, Roshka assumes that once a point of the elastic body touches the rigid obstacle, it remains in contact with it during the entire loading process. This may be so in many cases, but not always. Roshka's procedure is not general in that it does not allow points

to _detach_ from the rigid obstacle once contact has occurred.

For example, consider an elastic body with part of its boundary having the shape of a "double-well" (see the following picture:

http://www.google.com/imgres?imgurl=http://www.scielo.br/img/fbpe/bjp/v30n4/26fi02.jpg&imgrefurl=http://www.scielo.br/scielo.php%3Fpid%3DS0103-97332000000400027%26script%3Dsci_arttext&usg=__zsNnBN664lPgOIV8hLqUXa3Seak=&h=328&w=442&sz=10&hl=en&start=0&zoom=1&tbnid=ItRtI_iIWbLvzM:&tbnh=116&tbnw=156&ei=nx5_TYfIF8mfOuTU5cAI&prev=/images%3Fq%3Ddouble-well%2Bpotential%26um%3D1%26hl%3Den%26lr%3D%26sa%3DN%26biw%3D990%26bih%3D536%26tbs%3Disch:1&um=1&itbs=1&iact=hc&vpx=452&vpy=94&dur=567&hovh=116&hovw=156&tx=113&ty=104&oei=nx5_TYfIF8mfOuTU5cAI&page=1&ndsp=15&ved=1t:429,r:2,s:0

Suppose we apply a vertical force at the center of the double-well (just between the two wells) from above, and thus we press the elastic body to a flat rigid wall that is lying below. Then initially the two minimum points of the "double-well" will touch the obstacle. But as we increase the load, the two wells will slide and undergo a "rolling" motion, the two points that have been in contact will rise, and there will be new points of contact. Roshka's procedure would force the two original points of contact to remain in contact and would not let them rise.

Suppose Roshka goes ahead and tries his procedure on this problem. Would it be possible to look at the results and see that something is wrong? The answer is yes. If some points of contact "want" to detach from the obstacle but the scheme does not allow them to do that, then _tensile_ normal stresses would develop between the two bodies! Surely this is impossible - there must be compression between two bodies in contact and there cannot be tension between them.

There are actually three conditions that must hold at any point on the boundary of elastic body that is potentially in contact with another body:

1. The gap 'g' between this point and the nearest point in the other body must be non-negative. (This is the non-penetration condition.)

2. The normal stress 's' at the point (normal to the surface) must be non-positive. (This is the non-tension condition.)

3. At least one of 'g' or 's' must be zero.

The third condition means that either the gap is zero (which means there is contact) or the normal stress is zero (which means there is no contact).

The contact problem can be posed as a constrained minimization problem. In this context, the three conditions above are called the Kuhn–Tucker

conditions in the mathematical literature.

Correct answer was obtained from:

Zvi Zaphir (Kol Hakavod!)

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

This time, a question that will challenge your algorithmic thinking.

Consider a computational domain in two dimensions, and consider a finite element mesh in this domain made of triangles. Each triangle has exactly 3 nodes, which are located at its vertices (Kodkodim). Suppose the mesh is defined by the following lists: a list of nodes and their coordinates, and a list of elements and the nodes that belong to them (the "connectivity").

Suggest an algorithm that finds all the nodes that are on the boundary of the computational domain. (This information is important from various reasons; for example if one wants to plot the boundary of the domain without plotting the whole mesh, or if one wants to apply a certain procedure only to the boundary nodes.)

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

Let's look at the edges (Tzla'ot) of the triangular elements. Each edge can be identified by a pair of neighboring nodes. For example, assuming that nodes 22 and 71 in the mesh are "neighbors", namely belong to a common element, then (22,71) denotes the edge that connects nodes 22 and 71. The edge (m,n) and the edge (n,m) are identical.

Now, what is the property that distinguishes between an internal edge and an edge that lies on the boundary of the domain? The answer is simple: an internal edge belongs to exactly two elements, while an edge that lies on the boundary of the domain belongs to only one element. This is the key for the algorithm that seeks the nodes on the boundary.

So the algorithm is as follows. For all the elements in the mesh, list all the edges of the elements, in one long list. If an edge appears for the second time, delete this double-edge from the list. After ending this process for all the elements, you will be left with a list of all edges that appear only one time in the mesh, namely belong to only one element. These are the edges that lie on the boundary of the domain. The nodes belonging to these edges are the nodes that lie on the boundary of the domain, which is th desired output.

Now I will quote from DR's answer:

"Such an implementation requires O(Nel^2) operations, where Nel is the number of elements in the mesh. If Nel is very large, perhaps it makes sense to create also an ordered 'inverse connectivity array' which contains for each node the element numbers it is connected to - this would take just one scan of the regular connectivity array. Now, to find the boundary nodes we scan the regular connectivity array; for each pair of nodes encountered in an element we use the inverse connectivity array and check only the elements which are attached to the nodes in the considered element. If no similar pair is found in those elements the pair of nodes is on the boundary. This algorithm needs O(Nel+(3P+1)Nnp) operations, where P is the maximal number of elements attached to any node, and Nnp is the number of nodes."

MW also suggested a different algorithm, that is based on geometrical information (coordinates of the nodes). This algorithm would be useful, for example, in finding the grid points on the external boundary for a finite difference grid.

Correct answers were received from:

Orna Agmon Ben-Yehuda, Rami Ben-Zvi, Daniel Rabinovich, Abigail Wacher, Micha Wolfshtein.

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

Here is a Roshka story again. Please be reminded that you are Roshka's boss in the computational engineering company BATALA (Benchmarks And Theoretical Analysis for Low-tech Applications).

Roshka: Boss, I worked today on the non-viscous flow problem you gave me. I know you told me to wait till May, when we will have the new code for solving Euler's equations, but I decided not to wait but use the code that we already have.

You: But the code that we already have is for viscous problems, namely for solving Navier-Stokes equations!

Roshka: Yes, but I found a way to do the analysis anyway.

You: Really?! Tell me about it.

Roshka: Well, first I just tried to run the Navier-Stokes code with an input of zero viscosity. It didn't want to run; I received an error

message. I assume that's why you think we can't use the Navier-Stokes code for our problem.

You: It's not just...

Roshka: But do not worry, I managed to fool the code! I entered a very tiny viscosity, nu=0.0000001. And the code ran!

You: And let me guess - the results don't make sense.

Roshka (insulted): Why do you think that? I am sure that such a small viscosity is not significant, and should give the same results as a zero viscosity. But the truth is I didn't have time to look at the results because I wanted to tell you about my great discovery: see how simple it is to use the Navier-Stokes code for non-viscous flows?

You (sigh): Oy, Roshka, Roshka, what will be the end of you?

Explain your reaction.

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

Let us start with some theoretical observations. (By the way, we are talking about subsonic flow, although I forgot to mention this in the Question.) Roshka says: "I am sure that such a small viscosity is not significant, and should give the same results as a zero viscosity." This is not true. The problem of viscous flow with small viscosity nu, that approaches zero (or with a Reynolds number that approaches infinity), is a _singular_ perturbation problem and not a regular perturbation problem. This means that taking the limit nu-->0 results in a singularity. What actually happens is that with a small nu there is a boundary layer across which some of the flow field variables change very rapidly. The smaller nu is, the thinner this boundary layer is and the larger the gradients in it become. When nu becomes zero, the layer becomes a layer of thickness zero, and the high gradients become a discontinuity.

Let us see why this happens mathematically. The term including the viscosity nu in the flow equations involves a second-order derivative. The equations thus have to satisfy two boundary conditions on the boundary: a no-penetration condition and a no-slip condition. If we take nu=0 the second-order term drops out and the equations (Euler) become first-order. Then they can satisfy only one boundary condition - the no penetration condition. So it is clear that taking the limit nu-->0 is singular in nature.

Now one may say: "Ok, so there is a thin boundary layer in the Navier-Stokes problem that Roshka solved. So what? The results will still be physical, right?" Probably not. If nu is extremely small, then the

boundary layer would be extremely small and the gradients across it would be extremely large. No "reasonable" grid/mesh that Roshka would take would be able to resolve the solution in this case, and typically the large errors would radiate away from the boundary and would pollute the entire numerical solution in the whole computational domain.

We should remark that there are special, very non-standard, schemes that know how to deal with thin boundary layers directly. For example, one may combine an asymptotic method and a numerical method that together do the job, with a reasonable mesh. But assuming that Roshka is using a standard CFD package, his results would not make sense.

Correct answers were received from:

Rafi Haftka, Amiel Herszage, Eli Turkel, Zvi Zaphir.

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

Suppose we have an equation to solve that looks like

$Au = f,$

with or without some additional conditions. Here u is the unknown. This equation may represent many different things. It can be a system of linear algebraic equations (where A is a matrix, and u and f are vectors), or a nonlinear scalar algebraic equation (where Au=A(u) is a nonlinear function of u, and u and f are constants), or a differential equation (where A is a differential operator, and u and f are functions), etc.

Suppose we solve the problem using a computational method, and obtain the approximate solution u_c.

Now if we know the exact solution, u_ex, we can calculate the error:

$e = u_{ex} - u_c .$

Of course, in practice we rarely have the exact solution, so we cannot calculate the error e. But we can calculate the residual r:

$r = Au_c - f .$

The residual tells us the extent to which the computational solution satisfies the equation. (If there are additional conditions, like boundary conditions, we need to calculate their residuals too; but let us consider for simplicity only the equation Au=f.) Many algorithms include the calculation of the residual in order to determine how well the

computational solution satisfies the equations of the given problem.

Now the question is: can the residual r always be trusted? If we solve a problem numerically and we calculate r and see that it is very small, does this always mean that the error is also very small? If yes - why can we be sure, and if not - in what cases the residual may be small and the error large?

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

Let us start with the bottom line. The residual r cannot always be trusted as a measure for the error e. There are certainly situations where the residual is small but the error is large. Let us give a few simple examples that demonstrate this fact.

Example 1. Consider the 2x2 matrix problem Au=f where the matrix A is diagonal with A_11 = 1 , A_22 = 10^-6 , and f (transposed) is (1 10^-6). The exact solution is u_ex = (1 1) . Suppose that our algorithm is stupid enough to obtain the (wrong) solution (1 2) to this problem. Then it is easy to see that |e|=1 , and hence the relative error is |e|/|u_ex| = 71 % . Namely the error is huge. However, if we calculate the residual we get r=(0 10^-6), and therefore the relative residual is |r|/|f|=0.0001 % .

Example 2 (OL). "Take for instance Poisson's equation on a 2D square, discretized with a 5-point stencil. Gauss Seidel relaxation slows down because it computes corrections based on residuals, but when the error is smooth, its residuals are << its size. Generally such errors are near null-space components of A. Notwithstanding, all such errors can often be collectively approximated by a coarser representation of your original equations (e.g. by interpolation from a coarse grid)."

Example 3 (AD). "Consider finding the root of a function, i.e. finding a s.t. f(a)=0. In this case for every guess x we can compute f(x), which is the residual. Let us now assume that near a f(x) is approximately c(x-a)^b. Then for c small or large enough b f(x) is 'small', while (x-a) is 'large'. For example if f(x) = (x-a)^9 then if |x-a|=0.01 then |R|=|f(x)-0|=10^(-18)."

The many answers I received included some more examples, which I will not write down in order for this answer to stay short.

To understand when this happens, let us assume that the problem is linear. The we have the following equations. On one hand the exact solution satisfies

A u_ex - f = 0 .

On the other hand, the approximate solution satisfies

$$A u_c - f - r = 0 .$$

We subtract the two equations, and use

$$e = u_{ex} - u_c$$

to obtain:

$$A e = - r .$$

This is sometimes called "the error equation". It means that the error satisfies the same equation as the original one, where the "load" f is replaced by minus the residual (-r). We say that the error is "driven by" the residual.

So now the question is: under what conditions e can be large even if r is small? This may happen when A is "almost singular" or "ill-conditioned". In this case the sensitivity of the solution to the data is very large, and a small change in r may cause a huge change in e. All the examples above were associated with an ill-conditioned operator A.

Here are two additional comments on this problem.

First, we note that in the case of matrices, the standard way to measure ill-conditioning is by the "condition number". However, Achi Brandt commented to a Question of the Month two years ago that the best measure is not the condition number of the matrix itself, but the condition number of another matrix, called the "bi-normalized matrix". See details in the 2004 paper of Oren Livne and Gene Golub (http://www.springerlink.com/content/m52512522l633w6v/ ; and see the interesting dedication!).

Second, RH comments that the condition number is not very useful in that it does not tell us when there is a real trouble. There are many cases where the operator is ill-conditioned, yet nothing bad happens. For example, finite element matrices for typical structural models used in industry are ill-conditioned, yet accurate solutions are obtained. In Rafi's words, "the critical issue is whether the forcing function f triggers the bad properties of the operator A or is a benign forcing function." See the Note by RH in AIAA Journal, Vol. 28, pp. 1322-1324, 1989.

Correct answers were received from the following:

Orna Agmon Ben-Yehuda, Adi Ditkowski, Rafi Haftka, Amiel Herszage, Oren Livne, Jonathan Tal, Eli Turkel, Shmuel Vigdergauz, Abigail Wacher, Asher Yahalom, Zvi Zaphir.

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

Consider the problem of finding a root of a non-linear algebraic equation, f(x)=0. Here f(x) is a given nonlinear function, possibly very complicated, and we seek an approximate value of x that satisfies this equation. For simplicity, let us assume that there is a single root.

There are many known methods that start from an initial guess x=x0 and approach the solution x iteratively. Among these methods, "Newton iteration" (or Newtom-Raphson) is known to be the fastest: it approaches the exact solution with a quadratic rate (rate 2), which is the optimal rate of convergence. (This means that the error at iteration j+1 is of the order of the square of the error at iteration j.) Other methods converge with a smaller rate. For example, the secant method convergences with rate of about 1.6, and the bisection method converges linearly (rate 1).

The question is then: why don't we _always_ use Newton? Why ever use any other method? (People do use other methods sometimes.)

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

The Newton iteration method has some disadvantages relative to some methods with a lower rate of convergence. The most important ones are:

1. When our initial guess is not sufficiently close to the exact solution, convergence of the Newton method is not guaranteed. (We will not dwell on what "sufficiently close" mean.) If you look at the convergence proof (in every book on numerical analysis) you will see that some assumptions are made with respect to this issue.

2. Newton's method requires the calculation of the derivative f'(x). If the function f(x) does not have a derivative at all points (i.e., not smooth), then the method may fail, or the rate of convergence will not be quadratic. But even if f'(x) exists everywhere, if the function f(x) is very complicated, or perhaps only given discretely, then it may be very difficult or impossible to find an exact derivative.

In addition, one should note that:

3. (A comment by RH and ET) When the solution is a double root, as with the function f(x)=x^2, the Newton convergence is only linear, not quadratic.

4. (ET, OL) All this gets more complicated in the vectorial case, namely

for systems of nonlinear equations, such as those obtained from finite element methods for nonlinear PDEs.

5. (ET, OL) There are also 3rd-order accurate methods, generally known as Halley's method; see, e.g., http://folk.uib.no/ssu029/Papers/Sharma07.pdf and http://en.wikipedia.org/wiki/Halley%27s_method . But these methods require the calculation of the second derivative f"(x) which in turn may be associated with even more Tzarot.

6. An interesting comment made by OL is the following: "Faster convergence does not imply higher cost-effectiveness. Computing the derivative may be even more costly than evaluating f, so the convergence per unit work may be slower than, say, the secant method. For example, Newton requires a division operation, which is more expensive than the the bisection operations of multiplication by 0.5, addition, and f evaluation. Division, of course, is computed (by the machine - DG) via Newton's method..."

7. (OL) "Newton's method converges quadratically asymptotically, but not necessarily in the first one or two iterations. Since most applications do not require more than a few significant digits, the rapid convergence doesn't kick in yet."

8. (OL) "High accuracy introduces another problem: f' is more numerically sensitive to round-off errors, requiring a higher-precision arithmetic."

9. (RH, RBZ) All the above does not mean that you should abandon Newton altogether... "It is advisable to start with a more robust root finder, and pass to Newton iterations only when you get closer to the exact solution."

Correct answers were received from:

Orna Agmon Ben-Yehuda, Rami Ben-Zvi, Rafi Haftka, Oren Livne, Daniel Rabinovich, Jonathan Tal, Eli Turkel, Asher Yahalom, Zvi Zaphir.

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

The June Question asked about the disadvantages of the Newton iteration method for solving nonlinear algebraic equations. We should remark that the Newton iteration can be described by the linear equation

f'(x_n) [x_{n+1} - x_n] = - f(x_n) .

The secant method approximates f'(x_n) by a finite difference, and is thus described by the equation

f(x_n) - f(x_{n-1})

---------- [x_{n+1} - x_n] = - f(x_n) .
x_n - x_{n-1}

In this context I received the following two interesting comments, comparing Newton's method to the secant method.

1. Hillel Tal-Ezer writes: "Basically, as opposed to the common belief, the secant method is more efficient then Newton's method since in the latter one has to compute 2 functions per iteration while in the secant method one has to compute only one. Hence, the rate of convergence per function computation is sqrt of 2 and it is less then 1.6 which is the rate of the secant method." This is especially true with a system of equations. Even in the scalar case, where one can divide by f'(x_n) and then evaluate f(x_n)/f'(x_n), one may have to evaluate f(x_n) and f'(x_n) separately if f(x) is a complicated function (say, given by an infinite series).

2. Oren Livne writes that although the above is true, "depending on the initial guess, Newton might require even less iterations relative to secant (than what we would expect from the rate of convergence). Especially if only a moderate accuracy is required, when all this is done within a time-dependent iteration. In this case one usually needs only 1 Newton iteration vs. 2 secants. Secant requires 3 evaluations for 2 iterations (not 2), so the gain is smaller."

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

What is "super convergence"? Give at least one example to this phenomenon.

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

Super convergence is the general name for the phenomenon in which a numerical method, which is usually associated with certain convergence properties, exhibits much better convergence properties under some special circumstances.

Here are a few examples, contributed by our readers:

1. (ES) Consider the equation

(ku')'=f

with some given boundary conditions, where u(x) is the unknown function, f(x) is a given function ("loading"), and k(x) is also given (material

property). This equation describes, for example, the linear static longitudinal deformation of a rod, or the steady-state heat conduction in a rod. Suppose we solve the problem by the finite element method, with standard linear elements (piecewise-linear shape functions), and with exact integration. Then it can be shown that the convergence rate in u to the exact solution will be 2. Nothing more dramatic than this can be said if k(x) is some general function. However, if k is constant, namely the rod is uniform, then an amazing thing happens: the solution can be shown to be exact at the nodes for any loading function f(x). This is called "nodal exactness" and is a nice example of super convergence.

Mathematically, the reason for this happening, is that the Green's function associated with this problem can be represented exactly with the piecewise-linear shape functions. With standard finite elements, nodal exactness happens only in very simple cases, like the one above.

2. (ME) A similar phenomenon happens with beam bending, which is governed by the 4th-order equation

(EI u")" = f .

If the standard Hermite-cubic shape functions are used, the integration is exact and EI is constant, there is nodal exactness for any loading f(x).

The following examples were all suggested by OL. In all of them, super-convergence occurs because the leading term in the expression for the error vanishes in special circumstances.

3. The finite element approximation to the solution of the 2-D Poisson equation normally provides first-order accurate gradient, but under certain conditions one can obtain second-order accuracy.

4. The 5-point 2nd order finite-difference discretization of the 2-D Poisson's equation recovers the solution to 4th order accuracy, if all the solution's fourth-order derivatives happen to vanish. In this case, the leading error term is annihilated.

5. The p-th order numerical integration of a function whose p-th derivative vanishes will be more accurate than p-order.

6. Newton's method for root finding of f(x)=0 normally converges quadratically; however, near a root where f'(x) is nonzero but f''(x)=0, one obtains cubic convergence. Similarly for higher-order methods.

7. Consider the function f(x) = exp(-1/x^2) and define the power series g(x) = \sum {n=1}^{\infty} [f^{(n)}(x) (1-x)^n] / n!, where f^{(n)} is the n-th derivative of f. The partial sums will converge much more rapidly to the value of g at the point x=0 (and near it) than for all other x values. This is because all derivatives of f vanish at x=0, and therefore decay to 0 much rapidly with n for small x, as opposed to large x.

Correct answers were received from:

Moshe Eisenberger, Oren Livne, Evgeny Shavelzon.

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

On one of the tents in Rothschild, I saw the sign: "Give us more Roshka riddles!" So here is another Roshka scene. To remind you, Roshka is your employee in the computational engineering company BATALA (Benchmarks And Theoretical Analysis for Low-tech Applications). By the way, you would be surprised to know that there actually exists a company called Batala: see http://www.indiamart.com/batala-engineering/ .

You: Roshka, come here, I'd like to talk with you about the report you wrote on the quasi-static thermal stress analysis of the BATALA satellite, the Satlan.

Roshka: Boss, I cannot take another scolding from you. Every time that I do some work you tell me it is wrong. I have had enough of that. I even started to consider quitting BATALA and starting my own company.

You: Are you serious?

Roshka: Yes. I even played in my head with a possible name for my company. Maybe I will call it Results Evaluation and Simulation Technology (REST).

You: Not a bad name. But Roshka, actually I wanted to praise you for this report on the thermal stresses. I did not find any mistake in it, and it seems like a very good work.

Roshka: Ha, really? Then maybe I will stay in BATALA for a while...

You: Just one little thing before I sign this report. You did all these nice plots of the stresses, and I think it would be nice if you add some deformation plots, to show how Satlan deforms under the thermal loading.

Roshka: Hmmm, boss, there is a little problem with that. Although the stresses came out quite reasonable, the displacements look completely crazy.

You: What do you mean?

Roshka: All the displacements, at all the nodes without exception, came out astronomical, something of the order of 10^18. So I just ignored them and plotted only the stresses. The stresses have a reasonable order of

magnitude, as you saw in the report.

You: Well, I think I know what you did wro... what should be improved, Roshka. And I would not trust the stresses that you received either. Let me tell you what to do; don't worry, it's a very simple fix in the input. But you'll have to run the code again and do all the plots in the report again.

What was Roshka's mistake and what is the fix?

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

What probably happened was the following. The Satlan is not loaded by any surface loads since it "floats" freely in space. Therefore Roshka probably applied zero-traction (zero surface forces) boundary condition all over the surface of the model. But such a boundary condition allows arbitrary rigid body motion (RBM). In other words, the solution is not unique, since any RBM can be added and still all the equations and boundary conditions would be satisfied. Since there is an infinite number of solutions, the model's coefficient matrix (stiffness matrix in the Finite Element method) must be singular.

Now, if this was exactly the case, then Roshka would have received an error message saying "Error: The stiffness matrix is singular". However, due to round-off errors, it often happens that a theoretically-signular matrix becomes practically non-singular. (For a matrix to be singular, it's determinant must be zero. Any small perturbation in this matrix that would cause the determinant to become nonzero would make the matrix non-singular.) Since the matrix is non-singular, the code does not encounter any difficulties in solving the system and obtaining a solution. Because the matrix is "nearly singular", the solution of the system for the displacements gives huge numbers.

[Think of this: you want to compute 1/z, where z is actually zero but you are not aware of this. But because of round-off errors, z is not exactly zero but is 10^{-18}. Then the result you would get would be 10^18.]

The stresses are obtained from the derivatives of the displacements, and are calculated usually as a post-process after the displacements are found. Even though the displacements came out to be huge, the stresses, each of which is a "finite difference" of two huge numbers, come out in the correct order of magnitude. You can think of taking the derivative as removing the RBM from the whole motion, and leaving only the deformation. But because we are talking here about a cancellation caused by the difference of two huge numbers, the results may not be reliable, and this is why even if one is interested only in the stresses, it is advisable to do the whole calculation anew without any RBM.

All that is needed is to apply some zero-displacement (Dirichlet) boundary conditions to the discrete model in order to prevent RBM. (In the FE jargon this is called to "close" the DOFs.) Assuming that the Satlan model is a 3D model, with 3 degrees of freedom (DOFs) - ux, uy, uz - at each node, one needs zero-displacement boundary conditions at 6 wisely chosen DOFs in order to prevent the 6 rigid-body modes (3 translations and 3 rotations). Then the solution would be unique, the displacement solution would represent the deformation, and the stresses will come out correctly.

Of course, the displacement solution will be different if we "close" these 6 DOFs or those 6 DOFs. But the difference between the two solutions will only be up to RBM, which we do not care about anyway. The stresses will come out exactly the same regardless of how we prevent the RBM.

Correct answers were received from:
==========

Rami Ben-Zvi, Rafi Haftka.

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

Various time-depenednt problems in mechanics have the property that energy is preserved. Of course, if one introduces damping or another dissipative mechanism into the problem there is no preservation of energy, but in many cases people do want to look at models that have no damping in them.

Now, there are numerical methods that are energy-preserving. So, if we use such a method to solve a problem whose exact solution preserves energy, the energy-preservation property will be inherited by the approximate solution.

This sounds great. So why do people sometimes prefer to use a dissipative method (e.g., Lax-Wendroff scheme) instead of an energy-preserving method, even if they know that the exact solution _is_ energy preserving?

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

There are a number of reasons (somewhat related to each other) for sometimes preferring a dissipative scheme over an energy-preserving one:

1. Dissipative schemes typically have better stability properties than non-dissipative schemes. There are known cases where a certain discrete time-dependent model, based on a standard energy-preserving scheme, is unstable (and the solution blows up exponentially in time), but the

replacement of the scheme with a dissipative scheme cures the instability.

2. Using a dissipative scheme instead of an energy-preserving scheme is sometimes equivalent to introducing artificial damping/viscosity/dissipation into the problem. This often makes the problem more "well behaved" for numerical treatment. For example, suppose the exact solution of the original problem involves a sharp shock. With standard numerical methods it is very difficult to capture the shock, and in many cases the appearance of the shock may lead to a serious deterioration of the entire numerical solution. By introducing artificial viscosity (via the use of a dissipative scheme), the shock is slightly smeared, and this typically results in a much better behavior of the numerical method. The Khokhma of a good dissipative scheme is that it achieves this without (significantly) damaging the accuracy of the solution.

3. Some dissipative schemes are designed so that they selectively damp away "components" of the solution that are undesired. For example, numerical solutions of wave problems include "modes" of various frequencies, and whereas the low-frequency modes are usually well resolved, the high-frequency modes predicted by the method are totally spurious. These high spurious modes may cause a lot of trouble, and smart dissipative schemes suppress them while leaving the low modes unharmed.

4. AY: If one is looking for a steady state solution by advancing the solution in time, one may wish to use a dissipative scheme in order to converge to the steady state smoothly and avoid ever-lasting oscillations.

5. ET and AH also gave more specific explanations on the reason that the Lax-Wendroff scheme is successful. I will not include them here for the sake of brevity.

ET has made a very interesting "cultural" observation. He writes that in some fields like meteorology (numerical weather prediction), practitioners often refuse to use dissipative schemes, since these schemes "reduce the mass of the atmosphere", which the practitioners consider inappropriate. In other words, some or most computational meteorologists insist on looking at the problem in a very physical way, and so they view dissipative schemes as something that contradicts the laws of nature. I will leave it to you to decide if this approach is justified or helpful.

Correct answers were received from:

Orna Agmon Ben-Yehuda, Michael Bogomolny, Roland Glowinski, Amiel Herszage, Oren Livne, Eli Turkel, Asher Yahalom.

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

Take two problems that are similar in some general way but differ in their "details" - which of them is more difficult? The answer may depend on whether we are talking about an analytic treatment or a numerical treatment. There are some properties that make problems easier or more difficult for analytic/numerical treatment. For example, nonlinear problems are almost always more difficult than "similar" linear problems, for numerical treatment and certainly for analytic treatment. Problems associated with non-simple geometry are typically much more difficult for analytic treatment than simple-geometry problems, and may be also be more difficult for numerical treatment although not necessarily (depending on the method used).

As strange as it may sound at first, there is a "property" that if a problem possesses it becomes easier for analytical treatment (compared to a problem that lacks this "property") but more difficult for numerical treatment. In fact I can think of several such "properties". Give at least one example for such a "property".

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

Here are three situations involving a "property" that if a problem possesses it becomes easier for analytical treatment but more difficult for numerical treatment.

1. The problem involves a small parameter that makes it a "singular perturbation problem". Such a small parameter is a blessing for analytical treatment, because then one can use the powerful tools of asymptotic analysis. On the other hand, a small parameter may create a lot of difficulties numerically. One example is a problem whose solution involves very fast oscillations. Another example is a problem involving a thin boundary layer. Analytically one can use methods like WKB or multiscale or matched asymptotic expansions (see, e.g., the great book by Bender and Orszag). Numerically, high oscillations or thin boundary layers require non-standard methods, since with standard method one would need an extremely fine grid/mesh that might make the whole solution process impractical.

2. The problem involves a "constraint" on the field being solved of the divergence-free type, rotor-free type, etc. One example is an incompressibility constraint in fluid flow or in elasticity. Analytical treatment of "incompressible medium" problems is usually easier than that of "compressible medium" problems. But numerically, the incompressibility

constraint usually introduces a new level of difficulty into the method. In particular this is the situation with variational (Finite Element) methods.

3. Infinite domain problems. Analytically, problems associated with an infinite domain are almost always easier than problems associated with a finite domain. Example: it is much easier to solve (analytically) problems of infinite elastic plates than to solve problems of finite plates. On the other hand, the unboundedness of the domain often introduces significant difficulty to numerical methods, since most numerical methods are based on solving the equations in a finite computational domain.

Additional correct answers were proposed by the readers, related to energy dissipation, heat radiation, calculation of integrals, and other subjects. For the sake of brevity we shall not elaborate on them.

Correct answers were received from:

Rami Ben-Zvi, Oren Livne, Asher Yahalom.

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

The mechanical engineer in a mobile-phone company wants to check if the new mobile phone design is endurable enough to impact. Specifically, she wants to check if dropping the phone to the floor from a height of 1 meter will break it. She decides to do a numerical simulation using software that solves numerically the elastodynamics equations (e.g., a finite element code) and finds the stresses as a function of location and time. (Then she will insert the stresses to a little code that she wrote herself, that will tell her if the phone is endurable to impact or not.) She creates a discrete model of the phone (e.g., a finite element model). She assumes that the phone falls down such that its flat part hits the floor, and then the phone remains on the floor (there is no bouncing). What forcing / boundary conditions / initial conditions should this engineer prescribe to the model? (Thanks Yaakov Mosseri from Motorola for the inspiration.)

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

There are a number of approaches in doing this analysis. Probably the simplest one is the one proposed by RG and ZZ. Here is a quote from RG's answer:

"I would give to the entire model the impact velocity it reaches just before hitting the floor, sqrt(2gh), except the nodes on the surface of impact. To the nodes on the flat surface I would give a BC of zero vertical displacement (leaving the horizontal directions free). If friction with the floor is important there is a difficulty with this solution. For being practical and conservative it seems fine to neglect the friction."

The complete statement of this suggested model is, therefore, as follows. We solve the elastodynamics equations in the phone domain. There is no forcing (no body forces), so the equations are homogeneous. The boundary conditions are: zero traction on all the surfaces (namely the surfaces are free from any loading) except the flat bottom, zero horizontal traction on the flat bottom, and zero vertical displacement on the flat bottom. The initial conditions (at time t=0, which is the instant of impact) are: for all points zero displacements, zero horizontal velocity, and a non-zero vertical velocity which is equal to V=sqrt(2gh), the velocity just before the impact.

Note that the initial conditions dictate the velocity _before_ the impact, whereas the boundary conditions dictate the displacement on the flat bottom _after_ the impact. This makes sense mathematically; when some kind of discontinuity occurs at time t=0, initial conditions are originally given at time t=0- (an instant before the discontinuity) whereas boundary conditions are given at time t=0+ (an instant after the discontinuity).

ZZ proposed a few other approaches as well.

RH commented that there were various ways to understand the no-bouncing assumption. For example: (a) all the nodes on the flat bottom remain in contact with the floor; or (b) at least one point remains in contact with the floor; or (c) the center of gravity of the phone does not have any vertical motion after the impact; or (d) the average of the vertical motion of all the points on the flat bottom is zero at all times after the impact (this one is my own addition); etc. It is not clear which option is closer to the physical reality, but I think that option (a) is probably what most engineers would choose, since it is included in any commercial code that solves these types of problems. The other options are not so common, and one would have to develop special codes to implement them.

Correct answers were received from:

Ran Ganel, Rafi Haftka, Zvi Zaphir.

****

The December Question of the Month

****

Sometimes, when people try to solve a certain problem with a method that uses a mesh/grid, they say (usually in a worried voice) that the solution is "mesh dependent" or "grid dependent". This statement may sound strange, since of course _any_ numerical solution of _any_ problem depends on the density and geometry of the mesh/grid! So what do people mean when they say this?

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

When people say that the solution is "mesh dependent" or "grid dependent" they usually mean that the solution depends on the discretization in some pathological way. In practical terms, if some part of the solution keeps changing significantly when we refine the mesh/grid, we may say - using the CM jargon - that the solution is mesh/grid dependent.

Here is one example (suggested by ZZ). Suppose we are solving a problem involving a very thin boundary layer, but suppose that we are not aware of this layer and are using a standard numerical scheme with a cell/element size which is larger by a few orders of magnitude than the thickness of the layer. Obviously, as we repeatedly refine the mesh by, say, a factor of 2, the solution at the vicinity of the layer will keep changing significantly, and we will not see any sort of "convergence". In the best case, the solution will "jump" across one cell/element from its boundary value to its nominal value, and no matter how much we refine the grid, the solution will always "jump" across one cell.

Additional examples are:

* (ZZ) Shock-wave problems, when not resolved appropriately.

* (ZZ) Problems with cracks (or other kinds of singularities), when not treated appropriately.

* (RG) Elasto-plastic problems involving rupture and damage, e.g., a localization phenomenon due to strain softening.

* In some cases, a concentrated force applied at a finite element node may lead to a "mesh-dependent solution" in the vicinity of this node.

* A "mesh/grid dependent solution" may occur if we try to enforce on the numerical model more data than required for the problem to be well-posed. For example, suppose we are solving Laplace's equation (e.g., linear steady-state heat conduction) in a simply-connected domain (say, a "rectangle" with curved boundaries). We prescribe the function (temperature) along the entire boundary (Dirichlet condition). Now suppose that for some reason we want also to enforce the condition u=25 at one

internal point inside the domain (say, because we measured it in the lab). If we go ahead and enforce this value at a certain internal node, the solution will be "mesh dependent". This is because we are trying to solve a problem with too much data; the solution does not exist at all. (Even if u=25 is consistent with the analytical solution, it will usually not be consistent with the numerical solution, and thus the discrete problem will not have a solution.)

Correct answers were received from:

Rami Ben-Zvi, Ran Ganel, Asher Yahalom, Zvi Zaphir.