2014 Questions of the Month

****

The January Question of the Month

****

It has been a while since we heard from our friend Roshka. To remind you, Roshka is your employee in the computational engineering company BATALA (Benchmarks And Theoretical Analysis for Low-tech Applications). Here is a dialog between the two of you.

You: Roshka, I am glad you are finally back from your vacation in Timbuktu. How was it?

Roshka: Great! One day...

You: No, please don't tell me now - we don't have the time. I'd like you to finish the project of the aircraft wing I gave you. We need to provide answers to our client in two weeks. How much progress did you make in this project before you left?

Roshka: Boss, I assure you that when I was in Timbuktu I thought about this project all the time. There is a difficulty with it. You see, I have been using our in-house Finite Element code KASLAN. I modeled the structure as a three-dimensional elastic body, as you have told me.

You: That's good. We could have used shell and beam elements too, but from various reasons I decided to do very accurate modeling here.

Roshka: Now, one of the loads that we have is the thrust from the engine. I decided to apply it as a concentrated force acting at a point on the structure. I generated a mesh, and solved the problem. Then I solved it again using a refined mesh, to see the "convergence".

You: And how does it look?

Roshka: A very strange thing happens. The stresses near the engine location become much larger when I refine the mesh. I tried to refine it even more, and one time more, and each time that I refine it I get larger and larger stresses near the engine location. So the solution doesn't seem to "converge", and I don't understand what is happening.

Explain to Roshka what is going on and what he should do.

****

Answer

****

As all the readers who sent me their answers have noticed, the exact solution of a problem in 3D linear elasticity involving a concentrated (point) load is singular. This means that the stresses at the point of application of the load are infinite. So no wonder that when the mesh is refined the stresses become larger and larger. Roughly speaking, there _is_ convergence, since the approximate solution does approach the exact solution as the mesh density becomes infinite. More precisely, the convergence can be defined and shown under a certain appropriate norm, as OL and ET point out. However, in practice, with a point load, we would get very inaccurate results (the rate of convergence is quite low), unless we use special means to capture the singularity, as ET writes (for example, "singular finite elements"), and/or special non-uniform meshes. If the load is far away from any boundary, then one can subtract out the "singular behavior" from the outset, and solve a regular problem using the numerical method, as ET suggests.

So what is the remedy for this difficulty in the general case, when we do not have a code that takes the singular behavior into account, and/or we are not content (if we are engineers) with the information that the stresses are infinite at the load application point? As RH write, " Concentrated load does mean infinite stresses, but when you use finite element analysis the load is smeared over the elements bordering the application point. The smaller the elements the smaller the area it is smeared over. So in order to get accurate stresses near the point of application, we need to do the smearing ourselves, by estimating how the load would be actually distributed." ZZ writes similarly, "The load has to be applied as a distributed force on the area which on which the force acts in reality. Thus the finer the mesh, the more elements participate in carrying the load." RBZ also replied in the same manner.

Correct answers were received from:

Rami Ben-Zvi, Rafi Haftka, Oren Livne, Eli Turkel, Zvi Zaphir.

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

The following is a real situation that occurred recently. Researcher A shows researcher B some numerical results obtained for a time-dependent problem. Researcher A also shows to B some error graphs. Researcher B says: "Look, the error here _grows_ with time! This means that your solution (or method) is unstable. You must do something to stabilize it."

Is Researcher B necessarily correct? If she is right, what can researcher A do to improve the results? In she is not right, why not?

****
Answer
****

It is quite normal that the error obtained in numerical methods for time-dependent problems grows in time. This by no means indicates instability! For example, all the classical time-stepping (finite difference) methods, like Forward Euler, Backward Euler, Trapezoidal, Midpoint (Crank-Nicolson), etc., generate an error that grows linearly in time. In wave problems, such error growth is (almost) unavoidable, for the following reason. There is always some dispersion error, even if it is very tiny. This means that the numerical wave speed is not exactly equal to the true (physical) wave speed. (And in addition, waves with difference frequencies propagate with different numerical speeds.) As the waves propagate and move away from their source, the difference between the true speed and the numerical speed generates an error that grows in time. Note that the solution itself does not grow at all! In fact, the energy may be conserved, or it can even be dissipated. However, the error grows because the _phase_ of the waves becomes less and less accurate as times progresses and the waves propagate.

One may wonder: if the error grows, how can we claim that the method converges at all? Well, this depends on the way we define convergence. The usual definition of convergence in time-dependent problems is that the numerical solution approaches the exact solution _in the finite interval_ $0<t<T$, when the discretization (the mesh density and time-step size) becomes finer and finer. The emphasis here is on the time interval we look at being finite, namely we are interested only in the response up to time T, the simulation time. Indeed, if we want to reduce the error growth in that time interval, all we need to do is to refine the discretization. The error will still grow in time, but at a smaller rate. If the discretization is fine enough, we shall not notice with our eyes any error growth during the simulation time.

ZZ gave an example for the situation described above, with a beam problem. He also commented that the error can be reduced by applying an iterative procedure.

Convergence of a superior kind is convergence for an infinite time interval. If we have such convergence, we say that it is uniform in time. However, standard methods do not provide uniform-in-time convergence, and it is very hard to achieve it for wave problems. There are sophisticated methods that give "almost uniform in time" convergence, by guaranteeing an error growth proportional to (log t), which is slower than polynomial growth.

In contrast, instability in time-dependent problems usually (although not always) manifests itself in the growth of the solution itself. Typically this growth is exponential in time, and then we say that the solution "blows up". If this happens, of course we must look for the source of the instability, and we must fix it, or replace the method with another one which is stable.

A few readers commented that an instability in the numerical solution may be the result of the original problem not being stable in itself. Namely, the instability is not numerical but physical, or may originate from the original mathematical model. This is indeed true, and so when a researcher/practitioner solves a new type of problem (and this is especially true for nonlinear problems), she must first study the physical and mathematical behavior of the system. If the original model has some sort of singular or unstable behavior, this must be taken into account when attacking the problem numerically.

RH commented that sometimes the absolute error may grow while the relative error decreases (if the solution itself grows in time). If, in the Question, researcher A shows researcher B a graph of the absolute error and not of the relative error, this may lead to a wrong conclusion. Relative errors should always be preferred over absolute ones, in assessing the solution accuracy.

Correct answers were received from:

Rafi Haftka, Moti Karpel, Gabi Luttwak, Zvi Zaphir.

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

What is more challenging - numerical integration or numerical differentiation? More specifically, what is more difficult - to compute numerically the integral of a function in a certain interval, or to compute its derivative at a certain point? Let us assume that a very high accuracy (say 0.001% error) is required in both cases.

----> Remark: Try also answering this question if the accuracy required is much higher - machine precision.

****
Answer
****

There is no single definite answer to this question. The answer actually depends on how one defines "more difficult", and on the level of accuracy that one seeks. In addition, if the function has a certain type of singularity then one must go into the details to know which is more difficult. I have received different answers and arguments from different readers, and they were all correct. (It's like the story on the Rabbi who, when asked to judge between quarreling husband and wife, told each one of them that s/he was right, and when the Rabbi's wife complained to him that as a judge he should never do that, he told her she was right too.)

Here is the short answer. If accuracy is of utmost important, and one wishes to obtain a result with machine precision, whereas computing time is of no concern, then numerical differentiation is more difficult. This is because differentiation is based on dividing a very small number by a very small number, and when the numbers are extremely small it is very difficult to avoid round-off errors. On the other hand, numerical integration is based on summation, which is an innocent operation as far as maintaining accuracy goes. If only "reasonable" accuracy is expected, and the computational effort is of concern, then usually numerical integration is more "difficult", in the sense that it requires many more operations.

I choose to quote here the full answer of OL which more or less gives us "all we wanted to know" about this question.

"I would say the answer is "depends". I give 2 reasons for each side:

D1. If the function is smooth, then it is more challenging to differentiate it, because its integral is smoother than its derivative, so floating-point errors hamper numerical differentiation more (and it is harder to pick a mesh size that minimizes the error even if a closed-form formula for the derivative or its growth is known).

D2. A good argument for why differentiation is harder is that by observation, there exist mathematical groups whose research title is "automatic differentiation" (as at Argonne National Lab) but none in "automatic integration" that I've heard of. They also do compiler optimization of the finite-difference operations mentioned in D1, another important consideration in picking the meshsize.

I1. If the function is singular, one needs to refine the integration points near the singularity, and it is not always clear what the optimal pattern should be (adaptive schemes and different bases such as Pade could be used but require care). The same applies to very large integration intervals.

I2. If the function is very expensive to evaluate, differentiation may be easier from a cost-effectiveness perspective: one may obtain accuracy in O(log 1/e) evaluations by using a high-order rule, as opposed to the complexity of integration, which depends on the number of points. That's
because differentiation is a local operation and integration is long-range."

RH has also provided an interesting insight:

"Without restrictions on the function, I believe it is obvious that numerical integration is more difficult. Take a function that is equal to a very narrow Gaussian kernel. Whatever scheme of integration I use, I have very high probability of missing the very narrow region where the function has high values. In differentiation, on the other hand, I can limit myself to an epsilonic

region near the function and augment my precision to get an accurate derivative. The dimension is also important. In high dimensions the challenge of differentiation increases linearly with the dimension. However, if the function has an isolated high peak, the challenge for integration increases exponentially."

I will also make a comment on this issue in the context of the Finite Element (FE) method. If one obtains a FE solution for the primary function u (temperature, displacement, velocity...), and wishes to calculate some integral of this function, this is usually straight forward, and the overall accuracy of the result will be at least of the same order as that for u. On the other hand, if one wishes to calculate the derivatives of this function (fluxes, stresses, ...) then a straight forward calculation will reduce the accuracy to one order less than the accuracy of u. One then needs to resort to flux/stress recovery techniques, to obtain fluxes and stresses with higher accuracy. A lot of research has been done in the last two decades on recovery techniques for derivatives of FE results.

Correct answers were obtained from:

Orna Agmon Ben-Yehuda, Rafi Haftka, Oren Livne, Zvi Zaphir.

****

The April Question (Kushiya) of the Month

****

This question requires a light mathematical ability.

A class of problems in optimization consists of minimizing a function (or a functional, if the problem is variational) with equality constraints. A well known method to solve such problems is the use of Lagrange multipliers. In CM, the typical problem involves many unknowns, but in order to demonstrate the difficulty that we wish to demonstrate, we will consider a trivial algebraic problem with two unknowns. An analogous situation may occur in variational CM problems, and therefore a good understanding of the miniature algebraic problem is beneficial.

Consider a vector v of two components, v1 and v2. We are given that the sum of the two components is 1, namely v1+v2=1. This is the constraint. We wish to find, among all the vectors satisfying this constraint, the vector v of the smallest size (norm). Namely, we wish to minimize the function F = v1^2 + v2^2 (where ^2 means "square").

To solve this problem using the Lagrange multiplier method, we write the constraint as G=(v1+v2-1)^2=0 . Then we define the new function H=F+lambda G , where lambda is an unknown Lagrange multiplier. By the way, we squared the constraint so that both F and G have the same units, and so that the constraint function G be always non-negative, which seem like good ideas.

Thus,
H = v1^2 + v2^2 + lambda (v1+v2-1)^2 .
If we differentiate H with respect to v1, v2 and lambda, we obtain 3 equations with 3 unknowns:

2 v1 + 2 lambda (v1+v2-1) = 0 , (1)
2 v2 + 2 lambda (v1+v2-1) = 0 , (2)
v1 + v2 = 1 . (3)

If we use (3) in (1), (2), we see that the second terms in (1), (2) vanish, and we are left with the 3 equations:

2 v1 = 0 , (4)
2 v2 = 0 , (5)
v1 + v2 = 1. (6)

Obviously something is wrong here, since (4) and (5) imply that v1=v2=0, and this contradicts the constraint (6) that v1+v2=1.

What went wrong here and why, and how should we fix the calculation?

****
Answer
****

The Lagrange multiplier method has a "small print" warning, that people are not always aware of. The warning is that the gradient of the constraint function G must not vanish at the minimum point that we are seeking. If the gradient of G vanishes at the minimum point then the method would not work. In the example above, the condition is that the derivatives dG/d(v1) and dG/d(v2) should not be both zero for the minimum solution (v1,v2). But in our case we defined G=(v1+v2-1)^2, and hence dG/d(v1)=dG/d(v2)=2(v1+v2-1) which _is_ zero at _any_ point, because of the constraint. This means that, with the constraint function G that we have defined in the Question, the method of Lagrange multipliers would fail. This is the reason that we got "nonsense", namely a set of equations that is singular, and has no solution.

Clearly this Question of the month was very successful, since it has raised a lot of interest among our readers. I have received very nice answers, many of them long and detailed, and three of them typed up as short mathematical notes, with a lot of insight. I regret to have to omit all these answers for the sake of brevity. I will just summarize some of the main ideas.

DB offered a nice geometrical interpretation of the Lagrange multiplier theorem, and of the requirement that the gradient of the constraint function should not vanish at the optimum solution.

All the answers pointed to the remedy for the difficulty described. Defining G=(v1+v2-1)^2 is _not_ a good idea. If one defines G=v1+v2-1, without the square, then one would get the correct solution to this problem, which is obviously v1=v2=1/2 (and lambda=-1).

But defining G=v1+v2-1 would give us H = v1^2 + v2^2 + lambda (v1+v2-1) , which is not consistent in terms of units; that is why we squared the constraint in the first place. Three comments can be given regarding this complaint. First, unit consistency is not always very important. It is something "nice to have" but in many cases there are no consequences on the results of the calculation. Second, if one wishes to keep unit consistency, one can define G=A*(v1+v2-1), where A is a constant that has the same units of v1 and v2. Third, as LP has observed, it is always a good idea to non-dimensionalize the problem to begin with, so that all variables become non-dimensional. (I think that this is a type of a good advice that most of us are aware of but many of us cheerfully ignore.)

RG proposed an additional way to remedy the difficulty described, that yielded a singular system of equations with no solution. RG shows that it is possible to use a regularized version of the constraint function G=(v1+v2-1)^2. Namely, instead of taking G=(v1+v2-1)^2 which is no good, we can take G=(v1+v2-1)^2-epsilon^2. In other words, instead of taking the constraint (v1+v2-1)^2=0, we can take the regularized constraint (v1+v2-1)^2=epsilon^2. Thus, we are changing the original constraint only very slightly, and by doing so we are "regularizing" the system, which would give us a unique solution that is extremely close to the correct solution. Moreover, RG shows that when epsilon approaches zero, the solution of the regularized problem approaches the exact solution of the original optimization problem. Such regularization methods are very useful for CM, and are discussed, for example, in the books "Augmented Lagrangian Methods" by Fortin and Glowinski [1983] and "Augmented Lagrangian and operator-splitting methods in nonlinear mechanics" by Glowinski and Le Tallec [1989].

LP related, among other things, to the following question. In the example above the gradient of G vanished at _any_ point, so it was obvious from the outset that the Lagrange multiplier method would not work. But in more complicated situations it may happen that it would not be possible to check in advance if the gradient of G vanishes at the solution point or not, since we do not know the solution point in advance, or we simply do not want the trouble of even checking this solvability condition. In that case, we may innocently use the method without being aware of the difficulty. Would it be possible then that we would obtain a unique solution which would be wrong? This would be a very dangerous situation, because we might use this wrong solution to build an airplane...

Fortunately, LP explains, the answer is negative. If the gradient of G vanishes at the optimum solution, then the set of equations that the method would yield would always be _singular_. Namely, we will always have a clear warning sign that something is wrong in the solution process. Of course, singularity is guaranteed only if one solves the problem analytically, and we are more interested in numerical solutions. In that case, the singularity may manifest itself in the form of some numerical difficulties, typically ill-conditioning of the matrix, which would hopefully signal to us that something is terribly wrong.

Correct answers were obtained from:

Dmitry Batenkov, Meir Feder, Roland Glowinski, Rafi Haftka, Lydia Peres, Eli Turkel, Asher Yahalom.

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

Many current problems in CM are multi-disciplinary, namely involve coupled equations from two different disciplines. In the jargon of some CM conferences they are called "multi-physics" problems, although this is obviously an abuse of the language. (There is only one physics; well, two if you count Newton's physics and Einstein's physics...) A better name would be multi-field problems. One example is magnetohydrodynamics (MHD), where a fluid flow field is coupled with a magnetic field, and each one affects the other. Another example is aeroelasticity, where a flow around a structure and the structure's elastic deformation are coupled, and each field affects the other. An important difference between these two examples is that in the first example the two fields (flow field and magnetic field) both occur in the same domain, whereas in the second example the two fields (flow field and elastic deformation field) occur in two different domains. However, in both cases there is coupling between two different types of fields which affect each other.

There are two main approaches to attack such multi-field problems. One is the monolithic or augmented or non-staggered or strongly-coupled approach, in which we solve _all_ the equations of all the fields together, as one big system of equations. The other approach is called the split or staggered or iterative or weakly-coupled approach, in which we solve for each field separately, in an iterative manner. For example, in the MHD problem, we may start with the assumption that there is no magnetic field, solve the fluid flow problem, obtain the flow field, solve the magnetic problem and compute the magnetic field generated by the flow field, then solve the fluid flow problem again, and so on.

What are the relative advantages and disadvantages of these two approaches?

****
Answer
****

The main advantage of the monolithic approach is that, by definition, it solves the whole problem in "one shot". In contrast, the split method requires a number of iterations to converge to the solution. The number of iterations may be large if our first guess is bad, leading to a large computing time. In certain cases the solution may not converge at all, and the split method would fail. ET also comments: "There are indications that a fully coupled model converges faster to a steady state and gives more accurate time dependent solutions."

The main advantage of the split method is that it allows us to split the problem into two simpler problems and then to use more standard solution methods for each of the simpler problems. For example, suppose we have access to an aerodynamics code and to an elasticity code, but we do not have an aeroelastic code. By employing an iterative solution procedure, we may use our existing codes alternately to solve the aeroelastic problem. In addition, as ET notes: "Each set of equations have different properties, and so can be solved by methods that are well known. Solving the coupled system may require different methods simultaneously together with the coupling between them." GL comments similarly: "Sometimes different approaches are used to solve the problems for the different fields. For example, Finite Volumes for fluids and Finite Elements for structures, or Lagrangian meshes for structures and solids, and Eulerian meshes for fluids. Then it is feasible and efficient to loosely couple these different processors (in an iterative way)."

As GL and AY note, splitting the problem to two separate problems is especially prudent if the two problems are associated with two completely different space/time scales. The splitting allows us to easily use a different grid/mesh/scheme-order for the two fields, to achieve the necessary resolution for each scale. (This is possible also in the monolithic approach, but is usually much more complicated.)

ET comments that in practice the MHD problem is almost always solved in a monolithic way. A different case is that of the "Navier-Stokes equations with a multi-equation turbulence model and possibly chemistry. In this case the two sets of equations have different time scales which complicates a coupled approach, ie finding a time step that is appropriate for both sets of equations. The fluid equations are essentially hyperbolic for high enough Reynolds numbers while the turbulence equations are parabolic. Hence, most codes use an uncoupled approach."

I will end with some interesting general comments received from readers.

ET notes: "There have been several studies comparing loosely coupled and fully coupled turbuelent calculations. The results are not conclusive but the general feeling is that fully

coupling does not justify the extra work both in programming and in execution time, though I assume that some might disagree with that statement."

RG notes: "If one uses a split method a clever coupling can save the day and lead to methods as stable and accurate as monolithic ones. These split methods are also faster and easier to implement. Various examples of such situations can be found in recent work of Canic and her collaborators on fluid-structure interactions in hemodynamics. I will add that monolithic methods work mostly because GEMRES can essentially solve everything, at high cost sometimes."

Correct answers were received from:

Roland Glowinski, Gabi Luttwak, Eli Turkel, Asher Yahalom.

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

The May question dealt with the pros and cons of "monolithic" vs. "split" or iterative solution of multi-field problems. The following are comments that have not appeared in the Answer to this question in our last message.

An obvious advantage of the split approach is that the splitting lowers the number of unknowns in each iteration. For example, in the MHD problem, a monolithic approach requires the simultaneous solution for the magnetic field and flow field (N_m+N_f unknowns in the discrete scheme), whereas a split approach requires the repetitive separate solution for the magnetic field (separately, N_m unknowns) and the flow field (separately, N_f unknowns). This fact has been mentioned by Eli Turkel.

However, Asher Yahalom points out that sometimes, taking the monolithic approach "allows one to take advantage (at the analytic level) of the symmetries of the problem and reduce the number of variables involved. In fact barotropic MHD which naively can be thought of as a seven function problem (velocity, magnetic field and density) is really a four scalar field problem." More details can be found in "Barotropic magnetohydrodynamics as a four-function field theory," by AY, EPL journal, 89 (2010) 34005.

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

First, we shall briefly explain what _implicit_ and _explicit_ time-stepping methods are. Those of you who are familiar with the concepts may skip to the Question itself at the end.

Suppose we have to solve a time-dependent problem - for example, unsteady heat-flow or unsteady fluid-flow or wave propagation - by a computational method. We shall think of a heat-flow problem as a prototype, where we wish to find the temperature filed u(x,y,z,t) in a certain body and a certain time interval. The computational method discretizes both space and time. Suppose space discretization is done using a grid-based method, like a finite difference or a finite element scheme. After space discretization the method has to apply time discretization; we will consider time-stepping (or time-marching) methods, where the code "steps in time", and in each time-step it finds the temperatures at the grid points based on the temperature values at previous time steps.

Typically, the time-stepping scheme leads to the following "formula" for the temperature u_{jn} at a certain grid point j and a certain time-step n:

u_{jn} = F0[u_{...n}] + F1[u_{...n-1}] + F2[u_{...n-2}] + ... (Eq. 1)

Namely, the temperature at grid point j at the current time-step n is determined by the temperatures at other grid points at the current time-step n (the function F0), and by the temperatures at previous time-steps n-1, n-2, etc. (the functions F1, F2, etc.). Please forgive my crude notation here.

Now, if in this formula it so happens that F0=0 then this is an explicit formula for u_{jn} based on previous values which we already know:

u_{jn} = F1[u_{...n-1}] + F2[u_{...n-2}] + ... (Eq. 2)

In this case the time-stepping scheme is called _explicit_. If, on the other hand, F0 is nonzero, then we see that there are unknowns in both the left side and the right side of (Eq. 1). Then we have to solve a system of algebraic equations in order to find all the unknowns. In this case the time-stepping scheme is called _implicit_.

Thus, in implicit time-stepping methods, one has to solve a system of algebraic equations in order to find the time-varying temperatures. In explicit time-stepping methods, there is no need to solve a system of equations, but just calculate all the temperatures by using a scalar

"formula". The time-stepping scheme that we choose determines whether the scheme is implicit or explicit.

Another way to write (Eq. 1) is in the matrix form
A u_new = R[past solutions] .
Here A is a matrix, u_new is the current solution vector (current temperatures at all the grid points) which we wish to find, and R is a known vector that depends on the past solutions, namely on the temperatures in previous time-steps that we have already found. Now, if A is a _diagonal_ matrix, the scheme is explicit. Otherwise, the scheme is implicit. The time-stepping scheme that we choose determines the matrix A (and the vector R), and thus determines whether the scheme is implicit or explicit.

The question is: under what circumstances would we prefer to use an implicit solver, and under what circumstances would we prefer to use an explicit solver? And why?

****
Answer
****

The relative advantage of an explicit solver is clear: it does not require the solution of a system of coupled algebraic equations, just a sequential solution of scalar equations, one by one. But the disadvantage of explicit solvers is that (almost) always they are only _conditionally_ stable. This means that in order to be stable (namely not to be extremely sensitive to tiny errors, leading to a "blow up" of the solution), the time-step size used with explicit methods must be small enough. (This is called the CFL condition.) This restriction may cause a lot of trouble. For example, suppose you are interested in solving a problem whose physical time scale is of the order of 1 second. In order to get good resolution of the solution, you wish to take a time-step of 0.1 second, maybe even 0.01 second. But if you are using an explicit scheme, it may happen that you will need to take a time-step of 0.00001 second in order for the method to be stable. If you take a larger time-step the solution will blow up, and the method will be useless. Taking such a tiny time-step, only for the sake of maintaining stability, is very undesirable. You will need to make a huge number of time-steps to reach the final simulation time, and you may end up having a much more costly calculation than if you used an implicit scheme which is unconditionally-stable. With the latter scheme you have to solve a system of coupled equations, but the big effort is invested only one time at the beginning (in decomposing the matrix), and then you may use large time-steps, so you will have to make a much smaller number of time-steps than in the explicit method.

So when does it make sense to use explicit methods? When the physical time scale is as small as the time scale dictated by the stability requirement. For example, if we wish to analyze a very rapid physical phenomenon whose time scale is of the order of 0.0001 second, then the fact that we need a time-step size of 0.00001 second to maintain stability is fine, since this is the

resolution that we need anyway to get an accurate solution. Therefore, fast processes - like impact, car crashes, a hot body thrown into cold water, etc., are typically analyzed by an explicit method. On the other hand, slower processes, like the thermal behavior of an orbiting satellite, are typically analyzed by an implicit method.

Numerical problems where there is a big gap between the physical and numerical time/space scales are called "stiff problems". (There is a more precise definition but we shall not elaborate on this.) RH writes: "Explicit methods are better for unstiff problems, while implicit methods are better for stiff problems because for stiff problems stability requirements will require minuscule time steps with explicit methods." Other readers gave the same answer, in similar or different words.

The grid/mesh size and the material properties also have a strong effect on the critical time-step required for stability in explicit methods. Therefore, the decision what kind of method to prefer, depends on these factors as well.

Some additional comments were received by the readers. RH comments that in Finite Element methods, explicit time-stepping is possible only if one applies "mass lumping" to the mass matrix. Mass lumping is not always safe (e.g., for some non-standard elements), and in such cases it is safer to use implicit methods. RBZ points out that in addition to the advantage of not having to solve a system of coupled equations, explicit methods are easier to implement and require less memory and storage. ET mentions that time-stepping done in order to reach steady state is always implicit. He adds that "Reality is a lot more complex than the standard answer. There are semi-implicit methods where some terms are implicit and some are explicit. There are explicit methods with preconditioners that are implicit allowing large time scales (much of my research in the last few years). There are dual time step methods for some class of problems which are theoretically implicit but in practice solved explicitly." So the separation between explicit and implicit methods is sometimes not so sharp. GL gives a nice answer and some examples, such as: "For an incompressible flow, the sound speed is infinite and only an implicit scheme is feasible." RG gave a nice example of a cloth simulation where the cloth is represented as mass-spring system.

Correct answers were received from:

Rami Ben-Zvi, Arie Gershengoren, Rony Goldenthal, Rafi Haftka, Roy Holland, Gabi Luttwak, Jonathan Tal, Eli Turkel, Zvi Zaphir.

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

A while ago, the Question of the Month asked how we would analyze the time-varying elastic stresses in a small object, like a mobile phone, that falls down on the floor (a practical question for mobile phone builders). Now let's ask a related question. Suppose a small and very rigid object (we can think of it as a cube) falls down on the floor, but now we are concerned with the floor and not with the object. We would like to calculate the time-varrying elastic stresses in the floor due to the impact (a practical question for floor builders). Note that we know the weight and the size of the object, the height from which it falls, and the elastic material properties of the floor. We can assume that the object does not bounce significantly off the floor once it touches it.

How will we do this calculation? (Note that no time-varying impact force is known, in case you thought of using it as input.)
You do not have to go into small technical details, but just to provide the main ideas of the computational model.

****
Answer
****

The most serious difficulty in this type of analysis is the lack of input on the impact force. In what follows we suggest a few ways to handle this issue.

One way to go about this is to include in the analysis both the floor and the impacting object. We do not have to simulate the fall of the object. We start (at t=0) from the minute the object touches the floor; its initial position and velocity is given. We may assume that for t>0 the object always touches the floor, so there is continuity in the displacements and tractions across the contact surface. Since the object is very stiff, we must give it very high material stiffness values, and this may be a source of numerical difficulties that we should be cautious about.

As ZZ comments, there are some engineering ways to estimate the impact force variation. For example, "the program MSC.Adams suggests that the force is defined by a stiffness depending exponentially on the penetration depth and accompanied with damping. All those parameters are selected by the user."

In lack of any other information, one may choose to assume that the impact force is an impulse (Dirac delta) in time, with an unknown magnitude P. Assuming that the whole problem is linear (linear elastic material and small deformation) and that no dissipation occurs, the magnitude of the force P may be determined in the following way. Due to conservation of energy, the total

energy (elastic + kinetic) must be constant at any time. We know the initial energy of the system, E0: it is the kinetic energy of the object when it hits the floor. All this energy is transferred to the floor. Hence, for any time t>0, the total energy of the floor must be E0. Now, we take P=1 and run the analysis. (How to implement a Dirac delta in time is the subject of another question; it is possible but we shall not go into details here.) We choose any time, say a few time-steps after t=0, and calculate the energy E1 from the velocities, stresses and strains obtained in the floor. The energy is known to be proportional to P^2. Hence, if for P=1 we obtained the energy E1, we can easily find the force P that yields the known energy E0, namely P=sqrt(E0/E1). Once P is known, we can multiply (scale) all the displacements, strains and stresses obtained from the analysis by P to get the true solution.

A more sophisticated method is possible, but it requires some programming, namely it does not allow the use of a standard commercial code. We do not assume anything on the time-variation of the impact force except that it is constant in each time step. This amounts to approximating the time-function of the impact force by a "stairs function" (see, e.g., http://dali.feld.cvut.cz/ucebna/matlab/techdoc/umg/chspec18.html). If the time steps are very small, this is a good approximation. The magnitude of the force in each step is unknown; we denote the unknown force at time-step n by Pn. We start from t=0, we take P1=1, and let the code march a single time-step. Then we calculate the total energy, and by the technique explained above find what the true P1 is. Then we correct (scale) the values of the displacements and velocities by multiplying all of them by P1. These values serve as initial values for the next time step. Then we take P2=1, and we advance by another single time-step. Again we calculate the energy, and by a similar technique determine the true value of P2. We continue the analysis in this fashion, till the final simulation time is reached.

There are other difficulties associated with this type of analysis, for example the fact that the floor is very large (unbounded). However, as the saying goes, Nistapek Beze.

Correct answer was obtained from:

Zvi Zaphir

****

A Comment on the July Question of the Month

****

The July Question dealt with the stresses generated in a floor when a small stiff object hits it. The main issue was the lack of information on the impact force time variation. In the Answer, a few computational schemes have been proposed. The last scheme, described to be "a more sophisticated method", relied on conservation of energy, and attempted to find the force value in each time step. Following this, Ilan Degani wrote to me (quoted here with certain abbreviation): "It seems that what is actually assumed is that the force transfers all the energy

to the system in the first time step. If there is conservation of energy after that, then the force acts just in the first time interval and turns off after that. If the time-stepping method itself conserves energy, then the force after the first time-step will be zero and the energy constant." I think ID is perfectly right. The numerical scheme will yield a very small force (that has to do with the discretization only, not with the physics), after the initial step. This means that the "sophisticated method" has no advantage over a much simpler scheme in which the force applies only in the first time step. Kol Hakavod, Ilan!

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

This month's Question is about the solution of algebraic eigenvalue (e.v.) problems. We consider the simplest e.v. problem, which is of the form Kv=sv, where s is the eigenvalue and v is the eigenvector. The matrix K is assumed to be symmetric and non-negative (or positive semi-definite). This means that all the eigenvalues are non-negative. Usually we are interested in the positive eigenvalues and in the eigenvectors associated with them. The zero-e.v. modes, if exist, are usually non-interesting. They correspond to constant-temperature modes (in thermal problems), or to rigid-body motion modes (in structural problems), etc.

There are many algorithms that solve the e.v. problem Kv=sv and find all the eigenvalues and eigenvectors. Some of them require the inversion of the matrix K. Alas, if there are zero eigenvalues, the matrix K is singular and cannot be inverted. How can one overcome this difficulty and still use the algorithm that requires the inversion of the matrix, even when zero eigenvalues exist?

****
Answer
****

The difficulty is overcome by a simple operation called "eigenvalue shifting" (or "spectral shift"). Let us take the original problem Kv=sv, and add the vector pv to both sides, where p is a positive number. Then we get (K+pI)v=(s+p)v , where I is the identity matrix. Denoting K*=K+pI and s*=s+p, we get the new eigenvalue problem K*v=s*v , in which the matrix K is symmetric and positive definite, namely is invertible. Once this new eigenvalue problem is solved, the eigenvalues of the original problem can be found by s=s*-p. The eigenvectors of the two problems are identical.

OL also comments: "Usually not all eigenvalues and eigenvectors are needed. Either the smallest eigenpairs are of interest (ground energy), or the largest (web page ranking vector) are of

interest, or some global quantity that integrates over all eigenpairs is (say, the electronic density in density functional theory). Also, algorithms that depend on direct inversion are slow (at least O(n^3)). Many algorithms like Lanczos are in one way or another based on spectral shifting and inversion/solving a linear system K*v = f for some f, but they only approximately solve for the inverse action."

Correct answers were obtained from:

Rami Ben-Zvi, Roland Glowinski, Rafi Haftka, Oren Livne, Jonathan Tal, Shaul Tayeb, Zvi Zaphir.

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

Often one wants to find the physical state of a system after a lot of time has passed, and transient effects are gone. In Fluid Mechanics (FM) this is called the "steady state" solution, whereas in Solid Mechanics (SM) it is called the "static" solution. Consider a nonlinear problem in either FM (say, Navier-Stokes equations) or SM (say, large deformation of a structure), and suppose one is interested in the steady-state/static solution. Now, traditionally the computational approach taken is very different in FM and SM. In FM, the common approach is to use a numerical method for the time-dependent problem, and to step in time (in a special accelerated way) in order to reach the steady-state solution. On the other hand, in SM, the common approach is to consider the static (elliptic) nonlinear problem, start with an initial guess (say, a solution to a similar linear problem), and work iteratively (for example, using Newton-Raphson iterations) until convergence is reached.

The two approaches are so well rooted in the two communities, that sometimes I encounter researchers in FM that are not aware at all of the possibility to find the steady-state solution without stepping in time, and vice versa.

What is the reason for this difference in approaches between the two problems?

****
Answer
****

In my opinion, a major part of the answer is historical in nature. The oldest computational methods were finite difference (FD) methods and spectral methods. Finite volume methods and Finite element (FE) methods came significantly later. When the FE method started to become popular, in the 1960's, it was used mainly for SM problems. For FM problems, FDs continued to rule. Thus two different communities were formed then: FM people using mainly FDs and SM

people using mainly FEs. There was not a lot of contact between them, and the little there was had the air of a strong competition, not of sharing ideas. (Of course this is all generally speaking, and there were exceptions.) Each community developed its own "tools". The FD tools were much more established, whereas the FE tools were just started to be invented then. The idea to step in pseudo-time in order to reach steady state originally started from FD people. FE pioneers sort of ignored this tradition, and developed their own tool, namely iterative solution of the elliptic static equations (e.g. via Newton-Raphson). Each community liked to use its own favorite tools. Finite volume people adopted ideas from both communities, adding their own.

It should be remarked that this situation is far from being sharp nowadays. From obvious reasons the communities have become much closer to each other than in the 60's and 70's, and there is a lot of idea sharing, and a lot of people doing both FD and FE work. And in fact, today a lot of FM problems are solved by FEs, and quite a few commercial CFD codes are FE-based.

Actually, today the "facts" mentioned in the Question are not entirely correct, as ME comments. He and ET write that several commercial CFD codes do find the steady state by iterations of the steady-state equations. ME: "FLUENT and Star CCM+ which together command more than 50% of the CFD market are both cell-centered finite volume method codes. For the pressure-based family segregated methods, e.g SIMPLE and it's variants, they start from the steady form of Navier Stokes by default. Relaxation is used from one iteration to the next. However, there is a pseudo transient option when using a full coupled coupled solver approach where instead of adding relaxation into the center coefficient, a transient term with a false (Courant no) time step is added. FIDAP, a FE based CFD code, uses the steady form of the NS equations for both segregated pressured based methods and the coupled methods. CFX which uses is a node based control volume approach does indeed use a false transient approach in all cases."

RG writes: "In the particular case of viscous flows there is a strong belief that the time dependent approach coupled with good numerical schemes will produce only stable steady state solutions (the ones that people think take place in nature, within some approximation due to the veracity of the model). On the other hand if such stable solutions do not exist the time dependent approach will produce past, a critical Reynolds number, time periodic or close to time periodic solutions associated with a Hopf bifurcation phenomenon. That is the time dependent approach if properly monitored will tell us when Hopf bifurcation takes place, an important information indeed. Having said that there are many situations where unstable steady state solutions of a flow problem have interesting properties (minimal drag for example); it is then important to know them and then "make them stable" via active control if we are interested to take advantage of them. There are several ways to compute these unstable solutions, path following + Newton's being one of them, or to use a time dependent approach with additional dissipation so that we can compute steady state solutions beyond the Hopf bifurcation."

RG and ZZ also attribute the difference in approaches has to do with the fact that SM is based on a Lagrangian formulation whereas FM is based on an Eulerian formulation.

GL suggests an additional reason: since most FM problems are nonlinear, it may be harder to guess an initial steady-state solution that is required for the iterative approach.

AY writes: "It seems to me that in SM the equilibria (static solutions) are extrema points of some effective potential functional, the stable equilibria are minima points of the same. Of course the same is true also in some types of FM problems, however this knowledge is less wide spread."

ET writes: "The main difference between FM and SM is that the steady state equations of FM are NOT elliptic. They are a combined elliptic-hyperbolic system. In fact the Euler equations for supersonic flow are purely hyperbolic. When adding viscosity there is also a parabolic element. The early codes for the full potential equation indeed tried switching the difference formulae depending on whether the flow was subsonic or supersonic. However, this works only for a scalar equation. For a system of equations this becomes complicated and so the idea is to introduce time stepping which converts the system to a pure hyperbolic-parabolic system. One still needs upwinding but this is more for shocks."

AH relates to a similar reason: "In the problem of a blunt nosed body at supersonic or hypersonic speeds (and some other problems in FM)... there is clearly a sonic line between two flow regions, whose location is not known a priori. Now, even if one neglects the viscous terms and works with the Euler equations, the mathematical nature of the equations in the two regions is different. Namely, elliptic in the steady subsonic one, and hyperbolic in the steady supersonic one. Non time-dependent solution methods for one type of problem, would not work for the other one. Time dependent schemes instead, do not have this problem, since shocks can be then born without having been there before. This was the breakthrough used by Moretti and Abbett in their 1966 article. In the SM realm, ... it is dificult to think of situations like the ones described above."

Correct answers were received from:

Michael Engelman, Roland Glowinski, Amiel Herszage, Oren Livne, Gabi Luttwak, Eli Turkel, Asher Yahalom, Zvi Zaphir.

****

Comment on the Sept. Question of the Month

****

The Sept. Question dealt with the tendency in nonlinear Fluid Mechanics to find the steady state by stepping in pseudo-time in contrast to the tendency in nonlinear Solid Mechanics to find the static solution by performing iterations in order to find the solution to the static problem

directly. I described the historical background that partly led to these tendencies. Micha Wolfshtein comments on this:

"... The solution of the elliptic problem is attractive mainly for steady or weakly non-linear problems. It so happens that most fluid problems of interest these days are nonlinear and require some iterative methods. This is not the case for classic elasticity. Therefore we see more methods using inversion of the matrix operators among elasticians than among fluid dynamicists... Adding the elegance of FE compared to the clumsiness FD or FV it is not surprising that FE was the system of choice for the first numerical elasticians, in particular Grigull. Later French scientists favoured the method probably due to its elegance. Another reason for increasing popularity of FE is ease of meshing which is a part of basic system."

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

Suppose we have run a computational code and obtained numerical results. We wish to roughly estimate the errors in these results, preferably even locally (e.g., in this region the error is of the order of 2%, and in that region it is of the order of 10%). Alas, we have no analytical solution, no numerical reference solution and no lab experiments to compare our results to. All we have is our code and the results it obtains. How can we (roughly) estimate the errors of the results we have got?

****
Answer
****

In all the discussion below, it is assumed that the code has been debugged, that it yields correct results, and that it convergences to the true solution when the discretization is refined. In other words, we do not question the validity of the code or the numerical method used. All we ask is to estimate the errors for a specific set of results obtained by the code.

There are several possibilities. Perhaps the most obvious one is to solve the same problem with a much finer grid/mesh. For example, if the original problem has a grid/mesh with cells/grid-spacing/elements of size h, we may consider a grid with cell size h/2. We may regard the results of the much finer grid as the "reference solution", and estimate the errors by looking at the difference between the results of the two meshes. Even more accurate estimates can be done by comparing h, h/2 and h/4 grids. This is related to the Richardson extrapolation technique (see, e.g., Wikipedia, http://en.wikipedia.org/wiki/Richardson_extrapolation), which also gives accelerated convergence. By the way, the two additional grids do not have to be finer than the original one; they may be coarser, or one may be coarser and one may be finer, since what matters is to have three grids of significantly different densities. This avenue was suggested by

MW and OL. The latter also pointed to the connection with the so-called tau-extrapolation in the multigrid method.

Another option, which does not require running the code additional times, is to use one of the modern techniques to obtain an a posteriori error estimate. One such technique, suggested by AY, is based on calculating the residual of the original equations (the PDE, boundary conditions, etc.). Roughly speaking, one has to substitute the numerical solution obtained into the original equations, and see to what extent they are satisfied. If the original PDE is Lu+f=0, then plugging the approximate solution u* yields Lu*+f=r, where r is the residual. If L is a well-behaved operator, there is a good correlation between the residual and the actual error e=|u-u*|. This procedure cannot always be applied in a straight forward manner, since the approximate solution u* is not always smooth enough to allow us to substitute it into the PDE. In such cases, u* has to be smoothed first, say by using some kind of smooth interpolation.

Another a posteriori error estimation technique is possible in methods where the solution and/or its derivatives are defined anywhere in the domain but are not continuous. For example, in the standard finite element method, the value of the derivative of the solution (heat flux or stress) jumps from one element to its neighbor. In the limit where h--->0 these jumps tend to zero. The magnitude of the jumps can give us a good estimate on the error of these derivatives. A better still a posteriori error estimate is obtained from combining the information on the residual and the flux jumps.

ZZ and RH propose another way to estimate the error. Their idea is to find a "similar problem" that has an analytic solution, which would enable one to calculate the local errors. RH provides more details; his interesting proposed technique can be described by the following steps:
(1) Find an analytic expression that fits well the approximate solution. This can be done, for example by a "curve fitting" least-squares type technique.
(2) In the spirit of the "method of manufactured solutions", plug this analytic expression into the equations and boundary conditions, and find the forcing terms in these equations, so that the equations are exactly satisfied. Thus, you have found a new problem, similar to the original one, for which you have an analytical exact solution.
(3) Solve the new problem by the code, and from the results calculate the errors.

Correct answers were obtained from:

Rafi Haftka, Oren Livne, Micha Wolfshtein, Asher Yahalom, Zvi Zaphir.

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

Many industrial problems involve complicated geometry. When using finite elements or finite volumes, it is standard to use a mesh that fits the geometry (boundary-fitted mesh). With finite differences this is more tricky, but problems can be solved in complicated domains by transforming them first to simpler domains. However, using geometry-fitted meshes or grid transformations is not always convenient, and may require a large computational effort. In particular, if the geometry changes over time (e.g., a rotating propeller, or the domain change in some fluid-structure interaction problems) it may be too cumbersome and time-consuming to generate a new mesh for each configuration. Are there other options?

****
Answer
****

There is a class of computational methods that allow using a regular and fixed mesh/grid even when the geometry is complicated or changing in time. In this case the mesh does not fit the boundaries and interfaces, but the boundaries and interfaces are embedded inside the mesh. Thus, the boundaries and interfaces cross the mesh lines. Of course, this requires modification of the standard method (FE - finite elements, FV - finite volumes, FD - finite difference) to account for this fact. Here are a few types of methods in this class.

Immersed Boundary Method (IBM): This method has become quite popular in FE, FV and FD. RS writes: "The basic idea is that the geometry is immersed in a fixed Cartesian mesh... eventually only a change in the cells overlapping the boundary is needed, and some source terms are also added to the governing equations." OL writes: "Moving and immersed boundary methods keep a regular mesh, but rather approximate the moving boundary by a local approximation (piecewise linear or splines), and update the discrete equations around the interface at every time step to. The areas on both side of the interface in the cells straddling the lines can be integrated analytically given the approximation, and dependent quantities such as fluxes are re-discretized. Stated differently, the discretization scheme depends on the local parameters of the curve representation. Since the interface is a lower-dimensional manifold than the entire domain (say, a 1-D curve in a 2-D domain or 2-D surface in a 3-D domain), the extra computational cost of such a discretization is negligible as the grid becomes large." MB mentions that IBM was invented by Peskin (1972) for flows in the heart. SF mentions that his group has developed a version of IBB called "mirroring IBM" with distinct advantages.

Related methods are also called Embedded Domain methods and Fictitious Domain methods. See, for example, the announcement in http://13.usnccm.org/MS502, and note that one of the organizers is our Isaac Harari.

Methods based on Calderon projections and difference potentials: ET and Semyon Tsynkov have developed FD methods of this type, where some of the regular grid points lie outside the boundary.

X-FEM: These methods were developed by Belytschko et al. (see Obituary in the previous News Update) for solving problems with any kind of discontinuity, so that the FE mesh geometry remains fixed and is not bothered by the discontinuity. A major example is the problem of a propagating crack in an elastic body. As the crack propagates, the FE mesh remains fixed, and no remeshing has to be done for each new crack configuration.

MB also mentions the Nitsche method and other penalty methods for interfaces, and the Chimera methods, with a fixed grid and a moving one. "A good review is in Fixed Mesh Methods in Computational Mechanics by Codina and Baiges (2010)."

GL discusses three additional methods, used in the context of CFD: Volume of Fluid method, Level Set method and Shock Fitting method. He also shows the relation to Arbitrary Lagrangian-Eulerian (ALE) methods, where the motion of the mesh is fully controlled by the algorithm. Finally, GL writes: "Another possible approach, is to have separate and possible overlapping meshes for say the solid structure and the fluid. The classical example is the Coupled Eulerian Lagrangian (CEL) technique of W. Noh. The solid structure has a body-fitted Lagrangian mesh which is moving over an Eulerian mesh. The fluid fills up the part of the Eulerian mesh which is not covered up by the Lagrange mesh."

A totally different option is simply not to use a mesh at all. As RS writes, there are a few methods that are mesh-less: they involve a large number of "solution points" that are distributed in the computational domain, but these points are freely distributed (up to some mild limitations), do not necessarily form any pattern and are not connected to form a mesh. Among these methods are Element-Free Galerkin (of Belytschko et al.), Smoothed Particle Hydrodynamics and Moving Least Squares. AY mentions the method of Moving Coordinates as an additional option for problem with geometry that changes in time.

Correct answers were received from:

Michel Bercovier, Steven (Chaim) Frankel, Rafi Haftka, Oren Livne, Gabi Luttwak, R. Sreekanth, Eli Turkel, Asher Yahalom.

****

Comment on the November Question of the Month

****

The November Question dealt with schemes that solve problems with complicated geometry

without using body-fitted meshes. One class of methods of this type is called Immersed Boundary Method (IBM). In this context we wrote: "Steven Frankel (SF) mentions that his group has developed a version of IBM called "mirroring IBM" with distinct advantages." SF corrects me that mirroring IBM has been known before, and that the scheme that his group has developed is a novel multiblock IBM.

****

The December Question of the Month

****

Suppose we have a standard linear time-dependent problem to solve, say a transient linear heat conduction problem, or a problem of linear elastodynamics. There are two main general approaches to linear time-dependent problems: time-stepping and modal analysis.

In the time-stepping approach, we attack the problem directly by marching in time (e.g., by using finite difference approximations), and finding the approximate solution at each time-step.

In the modal analysis approach, first of all we solve the associated eigenvalue problem and find an approximation to the eigenvalues and eigenfunctions (up to a certain order). Then we express the unknown solution of the original time-dependent problem as an eigenfunction expansion with unknown time-dependent coefficients (like "Fourier coefficients"), and then we solve for these coefficients.

The question is: What are the relative advantages of each approach, and in what cases would we prefer the one or the other?

****

Answer

****

Modal analysis requires modal decomposition of the problem, which in many cases (especially if many modes are required to represent the time-dependent solution) is significantly more expensive than solving the problem one time by time-stepping. Thus, relatively low computational cost is the main advantage of time-stepping over modal analysis. (Other advantages will be mentioned below.) Therefore, typically when a single time-dependent simulation is desired, time-stepping is preferred.

However, there are often situations where the same system or sub-system has to be simulated again and again in various configurations - for different loads, different initial conditions, different sub-systems connected with it. In such cases, it pays to do a modal analysis - once and for all - for the system of interest, and then use the modes to repeatedly solve the problem very quickly in various configurations. This is a common approach in aeronautical engineering, for

example. After devising a model of the wing (say), one finds the first ~20 modes of the wing via modal analysis, and then uses these modes to solve various problems involving this wing, either isolated or as a sub-system of the entire aircraft.

A few readers pointed out that modal analysis would be cheaper than time-stepping in cases where the solution is known to consists of just a few modes, especially if one is interested in the long-time behavior.

Another advantage of modal analysis is that it provides excellent way to control what modes should be included in the solution. When using a time-stepping method, the solution actually contains all the modes of the discrete system. (If there are N degrees of freedom (DOFs) in the discrete model, there are N modes.) The high modes (modes associated with high frequencies) are known to have very poor accuracy; they are just high-frequency "noise" superposed on the solution. Alas, one cannot easily get rid of this "noise", and it may pollute the numerical solution. On the other hand, when doing modal analysis, one may choose what modes to use. For example, one may choose to take the lowest 20% modes, and ignore all the other modes (which is the right thing to do if those 20% low modes represent well the relevant physics of the system). This way the time-dependent solution will be clean of all the high-frequency "noise".

On the other hand, time-stepping allows all kinds of features that are not possible with modal analysis. For example, time-stepping allows time-step adaptive control, where the accuracy of the solution may be adjusted in various times and even at various locations; time-stepping may be used as a means of reaching the steady state solution; time-stepping allows a very high-resolution for very short processes (e.g., impact of cars), and more.

ZZ remarks that historically modal analysis was dominant, but with the increase of computer power, there is a tendency to prefer more and more direct time-stepping, even when the number of DOFs is large. RP adds that modal analysis is possible only for linear problems, while time-stepping is more general and is free from this limitation. OL adds that when long-term dynamics is desired, modal analysis allows a solution in late times without the need to step through the solution in earlier times. Also, multi-scale analysis fits more naturally with time-stepping than with modal analysis.

Correct answers were obtained from:

Oren Livne, Ronen Payevsky, Asher Yahalom, Zvi Zaphir