JAMES SWAN: OK. Let's go ahead and get started.

We saw a lot of good conversation on Piazza this weekend. So that's good. Seems like you guys are making your way through these two problems on the latest assignment. I would try to focus less on the chemical engineering science and problems that involve those. Usually the topic of interest, the thing that's useful to you educationally is going to be the numerics, right.

So if you get hung up on the definition of a particular quantity, yield was one that came up. Rather than let that prevent you from solving the problem, pick a definition and see what happens. You can always ask yourself if the results seem physically reasonable to you not based on your definition. And as long as you explain what you did in your write up, you're going to get full points.

We want to solve the problems numerically. If there's some hang up in the science, don't sweat it. Don't let that stop you from moving ahead with it. Don't let it make it seem like the problem can't be solved or there isn't a path to a solution. Pick a definition and go with it and see what happens, right.

The root of the second problem is trying to nest together two different numerical methods. One of those is optimization, and the other one is solutions of nonlinear equation, putting those two techniques together, using them in combination. The engineering science problem gives us a solvable problem to work with in that context, but it's not the key element of it. OK, good.

So we're continuing optimization, right. Move this just a little bit. We're continuing with optimization. Last time we posed lots of optimization problems. We talked about constrained optimization, unconstrained optimization. We heard a little bit about linear programs.

We started approaching unconstrained optimization problems from the perspective of steepest descent. OK, so that's where I want to pick up as we get started. So you'll recall the idea behind steepest descent was all the unconstrained optimization problems we're interested in are based around trying to find minima, OK.

And so we should think about these problems as though we're standing on top of a mountain. And we're looking for directions that allow us to descend. And as long as we're heading in descending directions, right, there's a good chance we're going to bottom out someplace and stop. And when we've bottomed out, we've found one of those local minima. That bottom is going to be a place where the gradient of the function we're trying to find the minimum of is zero, OK.

And the idea behind steepest descent was well, don't just pick any direction that's down hill. Pick the steepest direction, right. Go in the direction of the gradient. That's the steepest descent idea.

And then we did something a little sophisticated last time. We said well OK, I know the direction. I'm standing on top the mountain. I point myself in the steepest descent direction.

How big a step do I take? I can take any size step. And some steps may be good and some steps may be bad. It turns out there are some good estimates for step size that we can get by taking a Taylor expansion.

So we take our function, right, and we write it at the next iterate is a Taylor expansion. About the current iterate, that expansion looks like this. And it will be quadratic with respect to the step size alpha.

If we want to minimize the value of the function here, we want the next iterate to be a minimum of this quadratic function. Then there's an obvious choice of alpha, right. We find the vertex of this quadratic functional. That gives us the optimal step size.

It's optimal if our function actually is quadratic. It's an approximate, right. It's an estimation of the right sort of step size if it's not quadratic.

And so I showed you here was a function where the contours are very closely spaced. So it's a very steep function. And the minima is in the middle.

If we try to solve this with the steepest descent and we pick different steps sizes, uniform step sizes, so we try 0.1 and 1 and 10 step sizes, we'll never find an appropriate choice to converge to the solution, OK. We're going to have to pick impossibly small step sizes, which will require tons of steps in order to get there.

But with this quadratic estimate, you can get a reasonably smooth convergence to the root. So that's nice. And here's a task for you to test whether you understand steepest descent or not. In your notes, I've drawn some contours. For function, we'd like to minimize using the method of steepest descent. And I want you to try to draw steepest descent paths on top of these contours starting from initial conditions where these stars are located.

So if I'm following steepest descent, the rules of steepest descent here, and I start from these stars, what sort of paths do I follow? You're going to need to pick a step size. I would suggest thinking about the small step size limit.

What is the steepest descent path in the small step size limit? Can you work that out, you and your neighbors? You don't have to do all of them by yourself. You can do one, your neighbor could do another. And we'll take a look at them together.

OK, the roar has turned into a rumble and then a murmur, so I think you guys are making some progress. What do you think? How about let's do an easy one. How about this one here. What sort of path does it take?

Yeah, it sort of curls right down into the center here, right. Remember, steepest descent paths run perpendicular to the contours. So jumps perpendicular to the contour, almost a straight line to the center.

How about this one over here? Same thing, right? It runs the other way. It's going downhill 1, 0, minus 1, minus 2. So it runs downhill and curls into the center.

What about this one up here? What's it do? Yeah, it just runs to the left, right. The contour lines had normals that just keep it running all the way to the left. So this actually doesn't run into this minimum, right. It finds a cliff and steps right off of it, keeps on going. Steepest descent, that's what it does.

How about this one here? Same thing, right, just to the left. So these are what these paths look like. You can draw them yourself.

If I showed you paths and asked you what sort of method made them, you should be able to identify that actually, right? You should be able to detect what sort of methodology generated those kinds of paths.

We're not always so fortunate to have this graphical view of the landscape that our method is navigating. But it's good to have these 2D depictions. Because they really help us understand when a method doesn't converge what might be going wrong, right.

So steepest descent, it always heads downhill. But if there is no bottom, it's just going to keep going down, right. It's never going to find it.

Oh, OK. Here's a-- this is a story now that you understand optimization. So let's see, so mechanical systems, conservation of momentum, that's also, in a certain sense, an optimization problem, right. So conservation of momentum says that the acceleration on a body is equal to the sum of the forces on it.

And some of those forces are what we call conservative forces. They're proportional to gradients of some energy landscape. Some of those forces are non-conservative, like this one here. It's a little damping force, a little bit of friction proportional to the velocity with which the object moves instead.

And if we start some system like this, we give it some initial inertia and let it go, right, eventually it's going to want to come to rest at a place where the gradient in the potential is 0 and the velocity is 0 on the acceleration is 0. We call that mechanical equilibrium. We get to mechanical equilibrium and we stop, right.

So physical systems many times are seeking out minimum of an objective function. The objective function is the potential energy.

I saw last year at my house we had a pipe underground that leaked in the front yard. And they needed to find the pipe, right. It was like under the asphalt. So they got to dig up asphalt, and they need to know where is the pipe. They know it's leaking, but where does the pipe sit?

So the city came out and the guy from the city brought this. Do you know what this is? What is it? Do you know?

Yeah, yeah, yeah. It's a dowsing rod. OK, this kind of crazy story right, a dowsing rod. OK, a dowsing rod.

How does it work? The way it's supposed to work is I hold it out and it should turn and rotate in point in a direction that's parallel to the flow of the water in the pipe. That's the theory that this is supposed to work on.

. .

I'm a scientist. So I expect that somehow the water is exerting a force on the tip of the dowsing rod, OK. So the dowsing rod is moving around as this guy walks around. And it's going to stop when it finds a point of mechanical equilibrium.

So the dowsing rod is seeking out a minimum of some potential energy, let's say. That's what the physics says has to be true. I don't know that flowing water exerts a force on the tip of the dowsing rod. The guy who had this believed that was true, OK.

It turns out, this is not such a good idea, though, OK. Like in terms of a method for seeking out the minimum of a potential, it's not such a great way to do it. Because he's way up here, and the water's way underground.

So there's a huge distance between these things. It's not exerting a strong force, OK. The gradient isn't very big here. It's a relatively weak force.

So this instrument is incredibly sensitive to all sorts of external fluctuations. The gradient is small. The potential energy landscape is very, very flat.

And we know already from applying things like steepest descent methods or Newton-Raphson that those circumstances are disastrous for any method seeking out minima of potential energies, right. Those landscapes are the hardest ones to detect it in. Because every point looks like it's close to being a minima, right. It's really difficult to see the differences between these.

Nonetheless, he figured out where the pipe was. I don't think it was because of this though. How did he know where the pipe was? What's that?

STUDENT: Where the ground was squishy?

JAMES SWAN: Where the ground was squishy. Well yeah, had some good guesses because it was leaking up a little bit. No, I looked carefully afterwards. And I think it turned out the city had come by and actually painted some white lines on either side of the street to indicate where it was. But he was out there with his dowsing rod making sure the city had gotten it right.

It turns out, there's something called the ideomotor effect where your hand has very little, you know, very sensitive little tremors in it. And can guide something like this, a little weight at the end of a rod to go wherever you want it to go when you want it to. It's like a Ouija board, right.

It works exactly the same way.

Anyway, it's not a good way to find the minimum of potential energy surfaces, OK. We have the same problem with numerical methods. It's really difficult when these potential energy landscapes are flat to find where the minimum is, OK.

So fun and games are over. Now we got to do some math. So we talked about steepest descent. And steepest descent is an interesting way to approach these kinds of optimization problems. It turns out, it turns out that linear equations like Ax equals b can also be cast as optimization problems, right.

So the solution to this equation Ax equals b is also a minima of this quadratic function up here. How do you know? You take the gradient of this function, which is Ax minus b, and the gradient to 0 to minima. So Ax minus b is 0, or Ax equals b.

So we can do optimization on these sorts of quadratic functionals, and we would find the solution of systems of linear equations. This is an alternative approach. Sometimes this is called the variational approach to solving these systems of linear equations.

There are a couple of things that have to be true. The linear operator, right, the matrix here, it has to be symmetric. OK, it has to be symmetric, because it's multiplied by x from both sides.

It doesn't know that it's transpose is different from itself in the form of this functional. If A wasn't symmetric, the functional would symmetrize it automatically, OK. So a functional like this only corresponds to this linear equation when A is symmetric.

And this sort of thing only has a minimum, right, when the matrix A is positive and definite. It has to have all positive eigenvalues, right. The Hessian right, of this functional, is just the matrix A. And we already said that Hessian needs all positive eigenvalues to confirm we have a minima. OK?

If one of the eigenvalues is zero, then the problem is indeterminate. The linear problem is indeterminate. And there isn't a single local minimum, right. There's going to be a line of minima or a plane of minima instead. OK?

OK, so you can solve systems of linear equations as optimization problems. And people have tried to apply things like steepest descent to these problems. And it turns out steepest descent is kind of challenging to apply.

So what winds up happening is let's suppose we don't take our quadratic approximation for the descent direction first. Let's just say we take some fixed step size, right. When you take that fixed step size, it'll always be, let's say good for one particular direction.

OK, so I'll step in a particular direction. It'll be good. It'll be a nice step into a local minimum.

But when I try to step in the next gradient direction, it may be too big or too small. And that will depend on the eigenvalues associated with the direction that I am trying to step in, OK. How steep is this convex function? Right? How strongly curved is that convex function? That's what the eigenvalues are describing.

And so fixed value of alpha will lead to cases where we wind up stepping too far or not far enough. And there'll be a lot of oscillating around on this path that converges to a solution.

I showed you how to pick an optimal step size. It said look in a particular direction and treat your function as though it were quadratic along that direction. That's going to be true for all directions associated with this functional, right. It's always quadratic no matter which direction I point in. Right? So I pick a direction and I step and I'll be stepping to the minimal point along that direction. It'll be exact, OK.

And then I've got to turn and I've got to go in another gradient direction and take a step there. And I'll turn and go in another gradient direction and take a step there. And in each direction I go, I'll be minimizing every time. Because this step size is the ideal step size.

But it turns out you can do even better than that. So we can step in some direction, which is a descent direction, but not necessarily the steepest descent. And it's going to give us some extra control over how we're minimizing this function. I'll explain on the next slide, OK.

The first thing you got to do though is given some descent direction, what is the optimal step size? Well, we'll work that out the same way, right. We can write f at the next iterate in terms of f at the current iterate plus all the perturbations, right.

So our step method is Xi plus 1 is Xi plus alpha pi, right. So we do a Taylor expansion, and we'll get a quadratic function again. And we'll minimize this quadratic function with respect to alpha i when alpha takes on this value. So this is the value of the vertex of this function. So we'll minimize this quadratic function in one direction, the direction p.

But is there an optimal choice of direction? Is it really best to step in the descent direction? Or are there better directions that I could go in?

We thought going downhill fastest might be best, but maybe that's not true. Because if I point to a direction and I apply my quadratic approximation, I minimize the function in this direction.

Now I'm going to turn, and I'm going to go in a different direction. And I'll minimize it here, but I'll lose some of the minimization that I got previously, right?

I minimized in this direction. Then I turned, I went some other way, right. And I minimized in this direction. So this will still be a process that will sort of weave back and forth potentially. And so the idea instead is to try to preserve minimization along one particular direction.

So how do we choose an optimal direction? So f, right, at the current iterate, it's already minimized along p, right. Moving in p forward and backwards, this is going to make f and e smaller. That's as small as it can be.

So why not choose p so that it's normal to the gradient at the next iterate? OK, so choose this direction p so it's normal to the gradient at the next iterate. And then see if that holds for one iterate more after that.

So I move in a direction. I step up to a contour. And I want my p to be orthogonal to the gradient at that next contour.

So I've minimized this way, right. I've minimized everything that I could in directions that aren't in the gradient direction associated with the next iterate. And then let's see if I can even do that for the next iteration too. So can it make it true that the gradient at the next iterate is also orthogonal to p?

By doing this, I get to preserve all the minimization from the previous steps. So I minimize in this direction. And now I'm going to take a step in a different direction. But I'm going to make sure that as I take that step in another direction, right, I don't have to step completely in the gradient. I don't have to go in the steepest descent direction. I can project out everything that I've stepped in already, right. I can project out all the minimization I've already accomplished along this p direction.

So it turns out you can solve, right, you can calculate what this gradient is. The gradient in this function is Ax minus b. So you can substitute exactly what that gradient is.

A, this is Xi plus 2 minus b dotted with p, right. This has to be equal to 0. And you can show that means that p transpose A times p has to be equal to 0 as well.

You don't need to be able to work through these details. You just need to know that this gives a relationship between the directions on two consecutive iterates, OK.

So it says if I picked a direction p on the previous iteration, take how it's transposed by A, and make sure that my next iteration is orthogonal to that vector, OK.

Yeah?

STUDENT: So does that mean that your p's are all independent of each other, or just that adjacent p is? K, k plus 1 p's are?

JAMES SWAN: This is a great question. So the goal with this method, the ideal way to do this would be to have these directions actually be the directions of the eigenvectors of A. And those eigenvectors for symmetric matrix are all orthogonal to each other. OK? And so you'll be stepping along these orthogonal directions. And they would be all independent of each other. OK?

But that's a hard problem, finding all the eigenvectors associated with a matrix. Instead, OK, we pick an initial direction p to go in. And then we try to ensure that all of the other directions satisfy this conjugacy condition, right. That the transformation of p by A is orthogonal with the next direction that I choose.

So they're not independent of each other. But they are what we call conjugate to each other. It turns out that by doing this, these sets of directions p will belong to-- they can be expressed in terms of many products of A with the initial direction p. That'll give you all these different directions.

It starts to look something like the power iteration method for finding the largest eigenvector of a matrix. OK? So you create a certain set of vectors that span the entire subspace of A. And you step specifically along those directions. And that lets you preserve some of the minimization as you step each way.

So what's said here is that the direction p plus 1 is conjugate to the direction p. And by choosing the directions in this way, you're ensuring that p is orthogonal to the gradient at i plus 1 and the gradient i plus 2. So you're not stepping in the steepest descent directions that you'll pick up later on in the iterative process. OK?

So when you know which direction you're stepping in, then you've got to satisfy this conjugacy condition. But actually, this is a vector in n space, right. This is also a vector n space. And we have one equation to describe all and components.

So it's an under-determined problem. So then one has to pick which particular one of these conjugate vectors do I want to step along. And one particular choice is this one, which says, step along the gradient direction, OK, do steepest descent, but project out the component of the gradient along pi.

We already minimized along pi. We don't not have to go in the pi direction anymore, right. So do steepest descent, but remove the pi component.

So here is a quadratic objective function. It corresponds to a linear equation with coefficient matrix 1 00 10, a diagonal coefficient matrix. And b equals 0. So the solution of the system of linear equations is 00.

We start with an initial guess up here, OK. And we try steepest descent with some small step size, right. You'll follow this blue path here.

And you can see what happened. That step size was reasonable as we moved along the steepest ascent direction where the contours were pretty narrowly spaced. But as we got down to the flatter section, OK, as we got down to the flatter section of our objective function, those steps are really small. Right? We're headed in the right direction, we're just taking very, very small steps.

If you apply this conjugate gradient methodology, well, the first step you take, that's prescribed. You've got to step in some direction. The second step you take though minimizes completely along this direction.

So the first step was the same for both of these. But the second step was chosen to minimize completely along this direction. So it's totally minimized.

And the third step here also steps all the way to the center. So it shows a conjugate direction that stepped from here to there. And it didn't lose any of the minimization in the original direction that it proceeded along.

So that's conjugate gradient. It's used to solve linear equations with order n iterations, right. So A has at most n independent eigenvectors, independent directions that I can step along and do this minimization.

The conjugate gradient method is doing precisely that. Doesn't know what the eigendirections are, but it's something along these conjugate directions as a proxy for the eigendirections. So it can do minimization with just n steps for a system of n equations for n unknowns.

It requires only the ability to compute the product of your matrix A with some vector, right. All the calculations there are only depended on the product of A with a vector. So don't have to store A, we just have to know what A is. We have some procedure for generating A.

Maybe A is a linear operator that comes from a solution of some differential equations instead, right. And we don't have an explicit expression for A, but we have some simulator that produces, take some data, and projects A to give some answer, right. So we just need this product. We don't have to store A exactly.

It's only good for symmetric positive definite matrices, right. This sort of free energy functional that we wrote or objective function we wrote only admits symmetric matrices which are positive definite. That's the only way it will have a minimum. And so the only way a steepest descent or descent type procedure is going to get to the optimum.

But there are more sophisticated methods that exist for arbitrary matrices. So if we don't want symmetry or we don't care about whether it's positive definite, there are equivalent sorts of methods that are based around the same principle.

And it turns out, this is really the state of the art. So if you want to solve complicated large systems of equations, you know Gaussian elimination, that will get you an exact solution. But that's often infeasible for the sorts of problems that we're really interested in.

So instead, you use these sorts of iterative methods. Things like Jacobi and Gauss-Seidel, they're sort of the classics in the field. And they work, and you can show that they converge on are lots of circumstances. But these sorts of iterative methods, like conjugate gradient and its brethren other Krylov subspace methods they're called, are really the state of the art, and the ones that you reach to.

You already did conjugate gradients in one homework, right. You used this PCG iterative method in Matlab to solve a system of linear equations. It was doing this, right. This is how it works. OK? OK.

OK, so that's country ingredients. You could apply it also to objective functions that aren't quadratic in nature. And the formulation changes a little bit. Everywhere where the matrix A appeared there needs to be replaced with the Hessian at a certain iterate. But the same idea persists.

It says well, we think in our best approximation for the function that we've minimized as much as we can in one direction. So let's choose a conjugate direction to go in, and try not to ruin the minimizations we did in the direction we were headed before.

Of course, these are all linearly convergent sorts of methods. And we know that there are better ways to find roots of non-linear equations like this one, grad f equals zero, namely the Newton-Raphson method, which is quadratically convergent.

So if we're really close to a critical point, and hopefully that critical point is a minima in f, right, then we should rapidly converge to the solution of this system of nonlinear equations just by applying the Newton-Raphson method.

It's locally convergent, right. So we're going to get close. And we get quadratic improvement. What is the Newton-Raphson iteration, though?

Can you write that down? What is the Newton-Raphson iteration that's the iterative math for this system of non-linear equations, grad f equals 0? Can you work that out? What's that look like?

Have we got this? What's the Newton-Raphson iterative map look like for this system of non-linear equations? Want to volunteer an answer? Nobody knows or nobody is sharing. OK, that's fine.

Right, so we're trying to solve an equation g of x equals 0. So the iterative map is Xi plus 1 is Xi minus Jacobi inverse times g.

And what's the Jacobian of g? What's the Jacobian of g? The Hessian, right.

So the Jacobian of g is the gradient of g, which is two gradients of f, which is the definition of the Hessian. So really, the Newton-Raphson iteration is Xi plus 1 is Xi minus Hessian inverse times g.

So the Hessian plays the role of the Jacobian, the sort of solution procedure. And so everything you know about Newton-Raphson is going to apply here. Everything you know about quasi-Newton-Raphson methods is going to apply here.

You're going to substitute for your nonlinear. The nonlinear function you're finding the root for, you're going to substitute the gradient. And for the Jacobian, you're going to substitute the Hessian.

Places where the Hessian is, the determent of the Hessian is 0, right it's going to be a problem. Places where the Hessian is singular is going to be a problem. Same as with the Jacobian.

But Newton-Raphson has the great property that if our function is quadratic, like this one is, it will converge in exactly one step. So here's steepest descent with a fixed value of alpha, Newton-Raphson, one step for a quadratic function.

And why is it one step?

STUDENT: [INAUDIBLE]

JAMES SWAN: Good. So when we take a Taylor expansion of our f, in order to derive the Newton-Raphson step, we're expanding it out to quadratic order, its function is quadratic. The Taylor expansion is exact. And the solution of that equation, right, gradient f equals 0 or g equals 0, that's the solution of a linear equation, right.

So it gives exactly the right step size here to move from an initial guess to the exact solution or the minima of this equation. So for quadratic equations, Newton-Raphson is exact.

It doesn't go in the steepest ascent direction, right. It goes in a different direction. It would like to go in the steepest descent direction if the Jacobian were identity. But the Jacobian is a measure of how curved f is. The Hessian, let's say, is a measure of how curved f is. Right?

And so there's a projection of the gradient through the Hessian that changes the direction we go in. That change in direction is meant to find the minimum of the quadratic function that we approximate at this point.

So as long as we have a good quadratic approximation, Newton-Raphson is going to give us good convergence to a minima or whatever nearby critical point there is. If we have a bad approximation for a quadratic, then it's going to be so good, right.

So here's this very steep function. Log of f is quadratic, but f is exponential in x here. So you got all these tightly spaced contours converging towards a minima at 00. And here I've got to use the steepest descent step size, the optimal steepest descent step size, which is a quadratic approximation for the function, but in the steepest descent direction only. And here's the path that it follows.

And if I applied Newton-Raphson to this function, here is the path that it follows instead. The function isn't quadratic. So these quadratic approximations aren't-- they're not great, right.

But the function is convex, right. So Newton-Raphson is going to proceed downhill until it converges towards a solution anyways. Because the Hessian has positive eigenvalues all the time.

Questions about this? Make sense? OK?

So you get two different types of methods that you can play with. One of which, right, is always going to direct you down hill. Steepest descent will always carry you downhill, right, towards a minima.

And the other one, Newton-Raphson, converges very quickly when it's close to the root. OK, so they each have a virtue. And they're different. They're fundamentally different, right. They take steps in completely different directions.

When is Newton-Raphson not going to step down hill?

STUDENT: [INAUDIBLE]

What's that?

STUDENT: [INAUDIBLE]

JAMES SWAN: OK, that's more generic an answer than I'm looking for. So there may be circumstances where I have two local minima. That means there must be maybe a saddle point that sits between them. Newton-Raphson doesn't care which critical point it's going after. So it may try to approach the saddle point instead. That's true. That's true.

Yeah?

STUDENT: When Hessian [INAUDIBLE].

JAMES SWAN: Good, yeah. With the Hessian doesn't have all positive eigenvalues, right. So if all the eigenvalues of the Hessian are positive, then the transformation h times g or h inverse times g, it'll never switch the direction I'm going. I'll always be headed in a downhill direction. Right? In a direction that's anti-parallel to the gradient. OK?

But if the eigenvalues of the Hessian are negative, if some of them are negative and the gradient has me pointing along that eigenvector in a significant amount, then this product will switch me around and will have me go uphill instead. It'll have me chasing down a maxima or a saddle point instead.

That's what the quadratic approximation of our objective function will look like. It looks like there's a maximum or a saddle instead. And the function will run uphill. OK?

So there lots of strengths to Newton-Raphson. Convergence is one of them, right. The rate of convergence is good. It's a locally convergent, that's good.

It's got lots of weaknesses, though. Right? It's going to be a pain when the Hessian is singular at various places. You've got to solve systems of linear equations to figure out what these steps are. That's expensive computationally. It's not designed to seek out minima, but to seek out critical points of our objective function.

Steepest descent has lots of strengths, right. Always heads downhill, that's good. If we put a little quadratic approximation on it, we can even stabilize it and get good control over the descent.

Its weaknesses are it's got the property that it's linearly convergent instead of quadratically convergent when it converges. So it's slower, right. It might be harder to find a minima.

You've seen several examples where the path sort of peters out with lots of little iterations, tiny steps towards the solution. That's a weakness of steepest descent. We know that if we go over the edge of a cliff on our potential energy landscape, steepest descent it just going to run away, right. As long as there's one of these edges, it'll just keep running downhill for as long as they can.

So what's done is to try to combine these methods. Why choose one, right? We're trying to step our way towards a solution. What if we could craft a heuristic procedure that mixed these two? And when steepest descent would be best, use that. When Newton-Raphson would be best, use that.

Yes?

STUDENT: Just a quick question on Newton-Raphson.

JAMES SWAN: Yes?

STUDENT: Would it run downhill also if you started it over there? Or since it seeks critical points, could you go back up to the [INAUDIBLE].

JAMES SWAN: That's a good question. So if there's an asymptote in f, it will perceive the asymptote as a critical point and chase it. OK? And so if there's an asymptote in f, if can perceive that and chase it. It can also run away as it gets very far away. This is true. OK?

The contour example that I gave you at the start of class had sort of bowl shape functions superimposed with a linear function, sort of planar function instead. For that one, right, the Hessian is ill-defined, right. There is no curvature to the function. But you can imagine adding a small bit of curvature to that, right.

And depending on the direction of the curvature, Newton-Raphson may run downhill or it may run back up hill, right? We can't guarantee which direction it's going to go. Depends on the details of the function.

Does that answer your question? Yeah? Good.

STUDENT: Sir, can you just go back to that one slide?

JAMES SWAN: Yeah. I'm just pointing out, if the eigenvalues of h is further negative, then the formula there for alpha could have trouble too.

JAMES SWAN: That's true.

STUDENT: Similar to how the Newton-Raphson had trouble.

JAMES SWAN: This is true. This is true, yeah. So we chose a quadratic approximation here, right, for our function. We sought a critical point of this quadratic approximation. We didn't mandate that it had to be a minima. So that's absolutely right.

So if h has negative eigenvalues and the gradient points enough in the direction of the eigenvectors associated with those negative eigenvalues, then we may have a case where alpha isn't positive. We required early on that alpha should be positive for steepest descent. So we can't have a case where alpha is not positive. That's true.

OK. So they're both interesting methods, and they can be mixed together. And the way you mix those is with what's called trust-region ideas, OK.

Because it could be that we've had an iteration Xi and we do a quadratic approximation to our functional, which is this blue curve. Our quadratic approximation is this red one. And we find the minima of this red curve and use that as our next best guess for the solution to the problem. And this seems to be working us closer and closer towards the actual minimum in the function. So since quadratic approximation seems good, if the quadratic approximation is good, which method should we choose?

STUDENT: Newton-Raphson.

JAMES SWAN: Newton-Raphson, right. Could also be the case though that we make this quadratic approximation from our current iteration, and we find a minimum that somehow oversteps the minimum here. In fact, if we look at the value of our objective function at this next step, it's higher than the value of the objective function where we started.

So it seems like a quadratic approximation is not so good, right. That's a clear indication that this quadratic approximation isn't right. Because it suggested that we should have had an minima here, right. But our function got bigger instead.

And so in this case, it doesn't seem like you'd want to choose Newton-Raphson to take your steps. The quadratic approximation is not so good. Maybe just simple steepest descent is a better choice. OK, so it's done.

So if you're at a point, you might draw a circle around that point with some prescribed radius. Call that Ri. This is our iterate Xi. This is our trust-region radius Ri.

And we might ask, where does our Newton-Raphson step go? And where does our steepest descent step take us? And then based on whether these steps carry us outside of our trust-region, we might decide to take one or the other.

So if I set a particular size Ri, particular trust-region size Ri and the Newton-Raphson step goes outside of that, we might say well, I don't actually trust my quadratic approximation this far away from the starred point. So let's not take a step in that direction.

Instead, let's move in a steepest descent direction. If my Newton-Raphson step is inside the trust-region, maybe I'll choose to take it, right. I trust the quadratic approximation within a distance Ri of my current iteration.

Does that strategy makes sense? So we're trying to pick between two different methods in order to give us more reliable convergence to a local minima.

So here's our Newton-Raphson step. It's minus the Hessian inverse times the gradient. Here's our steepest descent step. It's minus alpha times the gradient.

And if the Newton-Raphson step is smaller than the trust-region radius, and the value of the objective function at Xi, plus the Newton-Raphson step is smaller than the current objective function, it seems like the quadratic approximation is a good one, right. I'm within the region in which I trust this approximation, and I've reduced the value of the function. So why not go that way, right? So take the Newton-Raphson step.

Else, let's try taking a step in the steepest direction instead. So again, if the steepest ascent direction is smaller than Ri and the value of the function in the steepest descent direction, the optimal steepest descent direction or the optimal step in the steepest ascent direction is smaller than the value of the function at the current point, seems like we should take that step. Right?

The Newton-Raphson step was no good. We've already discarded it. But our optimized steepest descent step seems like an OK one. It reduces the value of the function. And its within the trust-region where we think quadratic approximations are valid.

If that's not true, if the steepest descent step takes us outside of our trust-region or we don't reduce the value of the function when we take that step, then the next best strategy is to just take a steepest ascent step to the edge of the trust-region boundary.

Yeah?

STUDENT: Is there a reason here that Newton-Raphson is the default?

JAMES SWAN: Oh, good question. So eventually we're going to get close enough to the solution, all right, that all these steps are going to live inside the trust-region ring. Its going to require very small steps to converge to the solution. And which of these two methods is going to converge faster?

STUDENT: Newton-Raphson.

JAMES SWAN: Newton-Raphson. So we prioritize Newton-Raphson over steepest descent. That's a great question.

Its the faster converging one, but its a little unwieldy, right. So let's take it when it seems valid. But when it requires steps that are too big or steps that don't minimize f, let's take some different steps instead. Lets use steepest descent as the strategy.

So this is heuristic. So you got to have some rules to go with this heuristic, right. We have a set of conditions under which we're going to choose different steps. We've got to set this trust-region size.

This Ri has to be set. How big is it going to be? I don't know. You don't know, right, from the start you can't guess how big Ri is going to be. So you got to pick some initial guess.

And then we've got to modify the size of the trust-region too, right. The size of the trust-region is not going to be appropriate. One fixed size is not going to be appropriate all the time. Instead, we want a strategy for changing its size.

So it should grow or shrink depending on which steps we choose, right. Like if we take the Newton-Raphson step and we find that our quadratic approximation is a little bit bigger than the actual function value that we predicted, we might want to grow the trust-region. We might be more likely to believe that these Newton-Raphson steps are getting us to smaller and smaller function values, right. The step was even better than we expected it to be.

Here's the quadratic approximation in the Newton-Raphson direction. And it was actually bigger than the actual value of the function. So we got more, you know, we got more than we expected out of a step in that direction. So why not loosen up, accept more Newton-Raphson steps? OK, that's a strategy we can take.

Otherwise, we might think about shrinking instead, right. So there could be the circumstance where our quadratic approximation predicted a smaller value for the function than we actually found. It's not quite as reliable for getting us to the minimum.

These two circumstances are actually these. So this one, the quadratic approximation predicted a slightly bigger value than we found. Say grow the trust-region, right. Try some more Newton-Raphson steps. Seems like the Newton-Raphson steps are pretty reliable here.

Here the value of the function in the quadratic approximation is smaller than the value of the function after we took the step. Seems like our trust-region is probably too big if we have a circumstance like that. Should shrink it a little bit, right?

We took the Newton-Raphson step, but it actually did worse than we expected it to do with the quadratic approximation. So maybe we ought to shrink the trust-regional a little bit.

And you need a good initial value for the trust-region radius. What does Matlab use? It uses 1. OK.

It doesn't know. It has no clue. It's just a heuristic. It starts with 1 and it changes it as need be.

So this is how fsolve solves systems of nonlinear equations. This is how all of the minimizers in Matlab, this is the strategy they use to try to find minima. They use these sorts of trust-region methods.

It uses a slight improvement, which is also heuristic, called a dogleg trust-region method. So you can take a Newton-Raphson step or you can take a steepest descent step. And if you found the steepest descent step didn't quite get you to the boundary of your trust-region, you could then step in the Newton-Raphson direction.

Why do you do that? I don't know, people have found that it's useful, right. There's actually no good reason to take these sorts of dogleg steps.

People found that for general, right, general objective functions that you might want to find minima of, this is a reliable strategy for getting there. There's no guarantee that this is the best strategy. These are general non-convex functions.

These are just hard problems that one encounters. So when you make a software package like Matlab, this is what you do. You come up with heuristics that work most of the time.

I'll just provide you with an example here, OK. So you've seen this function now several times. Let's see, so in red covered up back here is the Newton-Raphson path. In blue is the optimal steepest descent path. And in purple is the trust-region method that Matlab uses to find the minima.

They all start from the same place. And you can see the purple path is a little different from these two. If I zoom in right up here, what you'll see is initially Matlab chose to follow the steepest descent path. And then at a certain point it decided, because of the value of the trust-region that Newton-Raphson steps were to be preferred. And so it changed direction and it started stepping along the Newton-Raphson direction instead.

It has some built in logic that tells it when to make that choice for switching based on the size of the trust-region. And the idea is just to choose the best sorts of steps possible. Your best guess at what the right steps are.

And this is all based around how trustworthy we think this quadratic approximation for objective function is.

Yeah, Dan?

STUDENT: So for the trust-region on the graph what Matlab is doing is at each R trust-region length it's reevaluating which way it should go?

JAMES SWAN: Yes. Yes. It's computing both sets of steps, and it's deciding which one it should take, right. It doesn't know. It's trying to choose between them.

STUDENT: Why don't you do the Newton-Raphson step through the [? negative R? ?]

JAMES SWAN: You can do that as well, actually, right. But if you're doing that, now you have to choose between that strategy and taking a steepest descent step up to R as well, right. And I think one has to decide which would you prefer.

It's possible the Newton-Raphson step also doesn't actually reduce f. In which case, you should discard it entirely, right. But you could craft a strategy that does that, right. It's still going to converge, likely. OK? OK.

I've going to let you guys go, there's another class coming in. Thanks.