Hastings Greer- Chaos and Trebuchets

[00:00:00] **Elizabeth:** This is a conversation between myself and Hastings Greer, an engineer and competitive trebuchet enthusiast. we were originally supposed to talk about how he uses chaos theory in his job, but I got nerd sniped by the existence of a competitive trebuchet scene and its untimely demise.

[00:00:18] **Hastings:** Hi, I'm Hastings Greer. I'm a PhD student at the university of North Carolina, and I'm studying image registration, which is the task of taking two MRI or CT scans and finding corresponding points between them, um, using deep learning, but I've got a hobby in, uh, trebuchet development.

[00:00:38] **Hastings:** Fractals, chaos theory, you know, math.

[00:00:42] **Elizabeth:** We met because I was talking about the applications of chaos theory on LessWrong and had come to the conclusion that there weren't that many and they were mostly overblown and that this was maybe intrinsic because chaos doesn't really tell you what you can do, it just tells you what's impossible.

[00:01:00] **Elizabeth:** And then you came to tell me that telling people what they can't do is incredibly valuable. Okay, can you talk a little more about that?

[00:01:09] **Hastings:** Yeah, um,

[00:01:13] **Hastings:** most of research is working out what you can't do, right? I mean, it's a thousand ways to make a light bulb. 999 or a thousand ways not to make a light bulb. Um,

[00:01:26] **Elizabeth:** Once you said it, it made a lot of sense and there's an ongoing problem I've had where ... so my background is in biology, I am a real Darwin fan.

[00:01:37] **Elizabeth:** But if you ask me what did evolution cash out to practically? I struggle a little bit. And I think that is partially because it's so embedded that it's hard to see what changed. Partially because a lot of what it did was tell us some other stuff was weird and that we needed to figure that out.

[00:02:00] **Hastings:** That makes sense.

[00:02:00] **Elizabeth:** Especially with plate tectonics, there was a bunch of fossil evidence and distribution of species that looked totally normal until you believed in evolution, at which point they looked insane.

[00:02:12] **Hastings:** Interesting. And then they started to look normal again once you worked out the continents moved?

[00:02:17] **Elizabeth:** Yes.

[00:02:17] **Elizabeth:** So you gave a couple of examples on Less Wrong, I think the most exciting of which was trebuchet development?

[00:02:24] **Hastings:** Oh, yeah, so I've been a big, um, medieval siege weaponry enthusiast since probably middle school. I was participating every year, and this, uh. Event called the World Championship Pumpkin Chunkin.

[00:02:37] **Hastings:** I don't know if you've heard of it, it's a quintessential East Coast rural hobby of who can launch a pumpkin the farthest. You are free to use whatever resources, financial or, or computational you have available. And so there's these two really beautiful branches, which is that. Some people just are able to put 30, 000 pound, 50 foot high monstrosities out there, the people who do that are usually the people who have a lot of experience with steelworking, , backseat of the pants engineering.

[00:03:09] **Hastings:** And then there's all the, the, the computer nerds with their complicated mathematical models.

[00:03:16] **Elizabeth:** Who wins?

[00:03:16] **Hastings:** It's a real toss up. It goes back and forth. It's like they turn out to be super balanced. If this is like Protoss vs. Humans vs. Zerg.

[00:03:28] **Elizabeth:** Do they ever team up? Does the team up just win?

[00:03:31] **Hastings:** The team up eventually wins.

[00:03:34] **Hastings:** And then splits off

[00:03:36] **Hastings:** Eventually it just couldn't afford liability insurance anymore and sort of petered out and the mid 2010s, but that was a, a really beautiful community going on for a while.

[00:03:46] **Hastings:** So the result is that I've written a lot of trebuchet simulators and subjected, Trebuchet simulators to. extreme optimization pressure and seeing whether they stay connected to the real world or not.

[00:04:00] **Elizabeth:** How useful are the algorithms? Like you mentioned using genetic algorithms to design new trebuchets. How useful is that if they can't be forced to obey physical laws like chaos? That's not the right phrasing, but you know what I mean.

[00:04:14] **Hastings:** You just tell it. So chaos is sensitivity to initial conditions, right? And so you can just add that to your loss function.

[00:04:28] **Hastings:** You can say, don't be sensitive to initial conditions.

[00:04:30] **Hastings:** I think a, this might be a time when we need a visible example.

[00:04:37] **Hastings:** The sort of first naive attempt at building a trebuchet, I just. Uh, took a simulator and threw a genetic algorithm at it and had it start trying configurations until one performed really well. And so this is its, this is its design and basically the system is just. Extremely chaotic, as you, because it's got parts spinning around wildly, and they all happen to very coincidentally, long at the time of the simulation, snap and dump all the energy into the projectile for a split second.

[00:05:18] **Hastings:** And so if you let go of the pumpkin at that time, You would have a perfectly efficient trebuchet.

[00:05:24] **Elizabeth:** Um, but there's no way to like reliably get it to do that is the issue.

[00:05:33] **Hastings:** So if you, if you in real life set up these initial conditions, it would not follow this exact trajectory because

[00:05:41] **Elizabeth:** Because you can't get the initial conditions perfectly.

[00:05:44] **Hastings:** Right. And, but, and you can sort of quantify that basically every time an object in this collection of particles makes a full orbit, you're getting one doubling of your error for more like one multiplication by a constant of your error. And so, um, the naive approach is to just forbid anything from spinning around more than once.

[00:06:15] **Hastings:** And that

[00:06:17] **Hastings:** I believe helped a little bit.

[00:06:22] **Hastings:** That produced, let's see this design, which spun around. It sort of approximately worked. It produced some design. It

[00:06:34] **Elizabeth:** looks like that last hinge is cycling repeatedly.

[00:06:37] **Hastings:** Yeah, so it's um, what happened was it was scored before the last, before it spun around more than once.

[00:06:43] **Hastings:** But, um, it turns out what's

[00:06:48] **Hastings:** better to do,

[00:06:50] **Hastings:** try and remember exactly how I solved

[00:06:51] **Hastings:** That's like ancient code. This is the most recent stuff I have.

[00:06:56] **Hastings:** Um, and when I gave

[00:07:02] **Hastings:** up on optimizing topology with software and just said, I, the human, I'm going to decide the topology of the mechanism and the software can only decide the arrangement.

[00:07:16] **Hastings:** Then just picking a topology that discouraged that sort of random motion along with a better program constraint, and it can only, parts can only spin around once turned out to work well. And so you can take something like this, which is bad trebuchet, which does not launch the pumpkin and just click optimize.

[00:07:39] **Hastings:** And it will

[00:07:44] **Hastings:** score zero always. So it can't learn

[00:07:46] **Hastings:** anything. It will quickly come up with an efficient configuration and avoid, sensitivity to initial conditions. But the best thing that I would like to do is actually just go in and hard code for every simulation, do 30,

do 30 simulations, all perturbed from each other and throw away anything where they diverge badly.

[00:08:07] **Elizabeth:** And when you say sensitivity to initial conditions, you mean that if you set the trebuchet up approximately the same, it will do approximately the same thing.

[00:08:18] **Hastings:** So you can see here, if I make small motions to this particle, The overall motion doesn't change wildly. Well, as if I

[00:08:29] **Hastings:** could see,

[00:08:36] **Hastings:** I did something like this. That's going to be chaotic. Uh,

[00:08:46] **Hastings:** sure.

[00:08:47] **Hastings:** Now, if I make small changes, we get wildly divergent. Um, qualitatively, qualitatively diversion behavior.

[00:08:56] **Elizabeth:** Oh, this is amazing. Okay. So I did not properly understand your example when you gave it written. I only understood watching it now, which is that you need to design a trebuchet that is not sensitive to initial conditions because a trebuchet that is that sensitive to initial conditions is not useful when launching pumpkins.

[00:09:17] **Elizabeth:** How much do you need chaos theory for this versus just to have the concept of sensitivity to initial conditions and enough compute that you can see when something is sensitive?

[00:09:30] **Hastings:** Um, so

[00:09:41] **Hastings:** you could just

[00:09:43] **Hastings:** have enough compute to see if something is sensitive.

[00:09:45] **Hastings:** Um,

[00:09:47] **Hastings:** and that would work fine, but I think you would be confused all of the time and you would just devote resources to working out why it was sensitive. Until you develop chaos theory.

[00:10:01] **Elizabeth:** This is another one of my hypotheses that I'm curious to your take on is like, it's not that chaos is doing anything we couldn't do without it, but it like simplifies your mental model and frees up RAM or internal human compute to do, to spend on something else.

[00:10:21] **Hastings:** I think it's more,

[00:10:25] **Hastings:** it's so fundamental that I think it's more like you could do addition a bunch of times instead of multiplying.

[00:10:35] **Elizabeth:** So you use, this is like very embedded in your work.

[00:10:39] **Hastings:** It's very embedded in writing a computer program that takes in numbers and does something complicated and puts out numbers.

[00:10:50] **Hastings:** , in numerical analysis, which is a class that took a long time ago,

[00:10:58] **Hastings:** the very first exercise we did, Um, was we were told you've got a lattice writing

[00:11:10] **Hastings:** utensil.

[00:11:12] **Hastings:** Is this going to work?

[00:11:14] **Hastings:** You've got a lattice of , every integer point placed in the grid. And you fire off a laser from the point one half, one half, going up at like one third, one. And calculate

[00:11:30] **Hastings:** how it bounces off these circular mirrors. And the exercise was, everyone implement this,

[00:11:43] **Hastings:** and now here's the professor's implementation.

[00:11:47] **Hastings:** Yours doesn't match it. You are wrong. I'm sorry,

[00:11:51] **Hastings:** no one in the class's measures matches the professor's or each other's.

[00:11:56] **Hastings:** The students all expected it to match perfectly.

[00:11:58] **Elizabeth:** And was this the lesson of the, of the project? .

[00:12:01] **Hastings:** And then because it's a high level math class, the second thing we did was calculate the derivative of location after any number of bounces. And this is an example chosen where it really nicely gets multiplied by some function of the difference between the radius and the.

[00:12:23] **Hastings:** The amount you, the distance, the ratio between the radius and the distance you traveled before hitting the sphere. So it really nicely scales by constant every single bounce and it makes it clear why the derivative of final position with respect to initial position has this extreme sensitivity. So it's.

[00:12:44] **Hastings:** It's, um, it was like the first class in numerical analysis is this is a problem you will have and you can't solve it.

The punchline there is just that CUDA is never going to, you never, any two CUDA implementations are not going to do the same computations, even if the same code .

[00:13:05] **Hastings:** And so if you are chaotic, then your long trajectories will never match between a CPU implementation and a CUDA implementation, while as for another project I'm working on, like the actual thesis research This huge neural network when we're writing tests for it, and there the CPU implementations don't match, but it's not a chaotic system.

[00:13:35] **Hastings:** So you can write your tests with an Epsilon and say, well, as long as they're within 1, 10, 000 of each other,

[00:13:45] **Elizabeth:** Just to restate this with. When you coded up the n-body problem, n-body is sensitive to initial conditions, so you move it slightly between the CUDA implementation and the DirectCPU implementation, and you can get wildly different answers.

[00:13:58] **Elizabeth:** Whereas the MRI software is supposed to be coalescing towards something real, if I can badly phrase it. So it would be very bad if it was sensitive to initial conditions and therefore you can look at multiple implementations and they should be approximately the same.

[00:14:17] **Hastings:** Yeah.

[00:14:18] **Elizabeth:** And you can assume it's an error if they don't.

[00:14:24] **Elizabeth:** , I would like to run one other theory by you, which is, so we've got Some value from Chaos Theory from you, the engineer, knowing what can't be done, or the scientist. I hypothesize that another thing Chaos Theory does is let the engineers and scientists tell their bosses that something is impossible, and their boss will accept "no because Chaos Theory" when they wouldn't accept "no, it doesn't work", or "no, it's sensitive to initial conditions".

[00:14:55] **Hastings:** I don't have any experience with that,

[00:14:57] **Hastings:** but it sounds like a good theory.

[00:14:58] **Elizabeth:** How much worse does your life get without formalized chaos theory? Or like, how much worse do individual genres or of projects? Is it like, everything takes twice as long? Do half the things you do become impossible?

[00:15:15] **Hastings:** What is, in this counterfactual, what is preventing me from noticing chaos?

[00:15:22] **Elizabeth:** You're a grad student, your graduate advisor is an asshole, and insists that numbers come out cleanly. Which, as far as I can tell, was the major problem with original chaos theory as well.

[00:15:33] **Hastings:** I would have given up debugging some things, but it wouldn't have affected my practical research because I haven't run into chaos in my PhD studies.

[00:15:44] **Elizabeth:** How about if you were doing a PhD in trebuchets, how screwed would you be?

[00:15:51] **Hastings:** Um, I think you could seat of your pants your way way through it. It would just slow things down. And now if I was doing a PhD in weather prediction, it would just be,

[00:16:01] **Elizabeth:** Yeah, everyone agrees. Chaos won weather prediction.

[00:16:04] **Hastings:** Yeah.

[00:16:05] **Hastings:** Um, orbital

[00:16:07] **Hastings:** mechanics. Like if I was trying to understand the structure of the solar system, or literally anything downstream of why we have caring about solar system structures, that it would just be game over.

[00:16:22] **Elizabeth:** , you said it would slow down trebuchets. And presumably, that can stand in for a lot of mechanical engineering projects.

[00:16:29] **Hastings:** Uh, God, you wouldn't be able to do

[00:16:34] **Hastings:** any sort of

[00:16:35] **Hastings:** Computational fluid dynamics without noticing it.

[00:16:37] **Elizabeth:** That's another place where I keep seeing hints of chaos being useful and I can't tell if it fell flat or became so foundational that we, no one talks about it.

[00:16:50] **Hastings:** Yeah, and I don't have, because I have been studying, it's been something I'm familiar with so long. It's a tool I reach for pretty quickly when winnowing out solutions. But I mean, there's a thousand things you're using to winnow out solutions when building a trebuchet. That's like material strengths, air, um, air friction, uh, ground deflection.

[00:17:11] **Elizabeth:** So if you take that out take twice as long, 10x as long?

[00:17:14] **Hastings:** Um, it would just. It would increase the ugh factor around simulation, dramatically. To have this monster eating things and not telling you why.

[00:17:26] **Elizabeth:** Okay, and you have no idea why. And It's not, it sounds like it's not necessarily taking less machine compute when you know that chaos is happening.

[00:17:36] **Elizabeth:** It's just that it feels more manageable and maybe gives you, like the tools you were showing earlier looked like they were built to highlight sensitivity to initial conditions.

[00:17:50] **Hastings:** The later ones that I've developed have been, yeah.

[00:17:53] **Elizabeth:** So that gave you something of a guide for what tools would be useful.

[00:17:57] **Hastings:** Yeah, and the final design that we used to get the best record our team ever got was like 1, 600 feet of pumpkin launch.

[00:18:07] **Hastings:** Actually, um,

[00:18:11] **Hastings:** really

[00:18:14] **Hastings:** skirts the edge of chaotic behavior.

[00:18:19] **Hastings:** What we ended up doing was we took a normal trebuchet and we, so if you take a normal trebuchet, so here we're getting a range of 330 feet and you make the counterweight 10 times heavier, um, you should get 10 times more range out of it. But in practice you lose coupling efficiency because the counterweight can only fall so fast.

[00:18:47] **Hastings:** And so instead here we're getting like, uh, four times farther instead of 10 times farther. And in practice it's even worse than that because. All of these components have to get heavier as your counterweight increases in scale. So what we did was, um, drop an extra component in the sling. So I don't know if you had I'm curious about

[00:19:13] **Hastings:** if you look at this system,

[00:19:16] **Hastings:** does this immediately scream is going to be chaotic,

[00:19:21] **Elizabeth:** What I want to do is play with it and then know the right answer. Um, my guess is that the system you showed that had a much smaller band of possibilities is less chaotic.

[00:19:35] **Hastings:** Exactly, and so because we had this insight from chaos theory that the problem with this sort of system is.

[00:19:47] **Hastings:** That your error is getting doubled every time something does a circle. Then this system, which has an ugh field around it because it has so much complex interacting complexity can still be predicted. So it's never going, it's, it's capable of doubling error very quickly, but if you scoot it on a trajectory that dodges all those error doubles, it stays predictable.

[00:20:11] **Hastings:** And so we were able to build it and predict its behavior to like the first try. All these tiny, like, lengths about releases and stuff, we could compute and get it to work in practice. And so that allowed And so

[00:20:28] **Elizabeth:** this is the actual software you use to design things, and the predictions mostly came true, although it sounds like there's some dampening factors.

[00:20:36] **Hastings:** Yeah, so this is an open source replication I've made of the soft, the software we use at the time working model 2d, um, is where I grabbed this feature of having these like little ghost images of the future from, but it was professional, like 30, 000 software that we just sent emails begging for it and we were a bunch of high school students.

[00:20:55] **Hastings:** So they said, sure.

[00:20:57] **Elizabeth:** All right, so that was going to be my next question is, I know anyone who watches this is going to be dying to know what software this is.

[00:21:02] **Hastings:** Uh, this is jstreb.hgreer.com/

[00:21:05] **Hastings:** It's this problem of scaling the counterweight, not buying So the range of the trebuchet is extremely efficient. If you, if your efficiency is fixed, it is extremely simple.

[00:21:18] **Hastings:** It's two times your mass ratio. So the difference between your counterweight and your pumpkin, um, times the distance the counterweight falls.

[00:21:28] **Hastings:** You can either get a better range by making your counterweight fall farther, which is basically free, but expensive, because you have to build a tall thing that's hard to fit under bridges, or by making your counterweight heavier, but that makes it really hard to design efficient mechanisms.

[00:21:46] **Elizabeth:** Because it can only fall so fast.

[00:21:48] **Hastings:** Right. Exactly. And so your energy gets locked up in this slow moving lumbering thing. And the result is that, um, there's almost. No two teams come up with the same topology. This is all I've got a bunch of different candidates over the years. So this fixed counterweight, um, which is just your most basic possible trebuchet.

[00:22:12] **Hastings:** So you can put a hinge on the counterweight. Um, you can use sliders instead of pivots and that boosts efficiency for complicated reasons, all the way up to like. This design was famous, famously over complicated.

[00:22:35] **Elizabeth:** Pretty though.

[00:22:36] **Elizabeth:** How much is progress in trebuchets driven by better design versus better physical technology?

[00:22:46] **Hastings:** Um,

[00:22:49] **Hastings:** better design. So there's a limit, which

[00:22:53] **Hastings:** is this 100 percent efficiency. Arbitrarily heavy counterweights, um, arbitrarily high mass ratios, so you never have to unfold it or anything, and I was sort of in the field as it went from this design being the best available, where, oh, right, the last thing is this counterweight is only following a small fraction of the height of the frame,

[00:23:23] **Hastings:** so

[00:23:26] **Hastings:** that ratio of how far the counterweight is following to your frame height, Makes your design more, some sort of meta efficient, like, better bang for your buck.

[00:23:38] **Elizabeth:** Um, because it lets you get more distance without making the trebuchet taller.

[00:23:42] **Hastings:** And so, when I sort of joined the forums investigating this sort of thing in 2006, it was, there was really no consensus on how to do this. Very few people involved in new math. And then, it progressed through the years until eventually, the um, unfortunately, the correct design was come up with.

[00:24:05] **Hastings:** Which is a counterweight moving over a pulley and then a cam on your throwing arm.

[00:24:13] **Elizabeth:** A cam?

[00:24:14] **Hastings:** Yeah, so

[00:24:17] **Hastings:** here you've got a rope going over all these points on your arm. And it's just allowed to, um, unwrap.

[00:24:26] **Hastings:** Oh.

[00:24:29] **Hastings:** And so this, because it only has two degrees of freedom. It's not, doesn't tend towards chaos. So it's easy to predict numerically and because you have a lot of control over your cam profile, it turns out you can just tune it for whatever efficiency you want at any mass ratio you want.

[00:24:46] **Elizabeth:** Okay. And so this could have been built for decades, hundreds of years, possibly. And the design was found in 2006.

[00:24:54] **Hastings:** Uh, this I think was found. I don't know when it was found. There was a mild kerfuffle because during the Arab spring, some group in Syria found the plans for this. on YouTube and built a copy and were lobbing bombs around with it, which was confusing to a bunch of nerds on A php forum.

[00:25:17] **Elizabeth:** Yeah, I also feel really confused like there is something awesome about Designing an improvement on medieval weapon technology But it turns out it's still

[00:25:28] **Hastings:** weapon technology bad

[00:25:29] **Elizabeth:** It was discovered fairly recently.

[00:25:31] **Hastings:** Yeah. With like no later than 2010, 2011. And then it took a while to everyone to notice that, Oh, for most constraints,

[00:25:41] **Hastings:** this is the correct design.