Long before robots could run or cars could drive themselves, mathematicians contemplated a simple mathematical question. They figured it out, then laid it to rest — with no way of knowing that the object of their mathematical curiosity would feature in machines of the far-off future.

The future is now here. As a result of new work by Amir Ali Ahmadi and Anirudha Majumdar of Princeton University, a classical problem from pure mathematics is poised to provide iron-clad proof that drone aircraft and autonomous cars won’t crash into trees or veer into oncoming traffic.

“You get a complete 100-percent-provable guarantee that your system” will be collision-avoidant,
said Georgina Hall, a final-year graduate student at Princeton who has collaborated with Ahmadi on the work.

Princeton, ORFE

Amir Ali Ahmadi, a professor at Princeton University, has shown how a sum-of-squares algorithm can be applied to modern optimization problems.

The guarantee comes from an unlikely place — a mathematical problem known as “sum of squares.” The problem was posed in 1900 by the great mathematician David Hilbert. He asked whether certain types of equations could always be expressed as a sum of two separate terms, each raised to the power of 2.

Mathematicians settled Hilbert’s question within a few decades. Then, almost 90 years later, computer scientists and engineers discovered that this mathematical property — whether an equation can be expressed as a sum of squares — helps answer many real-world problems they’d like
to solve.

“What I do uses a lot of classical math from the 19th century combined with very new computational math,” said Ahmadi.

Yet even as researchers realized that sum of squares could help answer many kinds of questions, they faced challenges to implementing the approach. The new work by Ahmadi and Majumdar clears away one of the biggest of those challenges — bringing an old math question squarely to bear on some of the most important technological questions of the day.

### Positivity Guaranteed

What does it mean for something to be a sum of squares? Take the number 13. It’s the sum of two squares: $2^2$ and $3^2$. The number 34 is the sum of $3^2$ plus $5^2$.

Instead of numbers, Hilbert’s question — the 17th of 23 he posed at the opening of the 20th century — has to do with polynomial expressions like $5x^2 + 16x + 13$. These kinds of polynomials can sometimes be expressed as sums of squares, too. For example, $5x^2 + 16x + 13$ can be rewritten as $(x + 2)^2 + (2x + 3)^2$.

When an expression is a sum of squares, you know that it’s always nonnegative. (Because anything squared is positive or zero, and the sum of positive numbers is a positive number.) Hilbert wanted to know if it works the other way around: if all nonnegative polynomials can be expressed as a sum of squares of rational functions. In 1927 the mathematician Emil Artin proved that Hilbert’s conjecture is true.

This relationship turns out to be quite useful. If you’re handed a complicated polynomial — one with dozens of variables raised to high powers — it’s not easy to determine straightaway whether it’s always nonnegative. “Some polynomials are obviously nonnegative, others are not. It’s hard to test whether they’re always nonnegative,” said Ahmadi.

But once you show that the same polynomial can be expressed as a sum of squares, then you know nonnegativity follows as a consequence. “Sum of squares gives you a nice certificate of positivity,” said Pablo Parrilo, a computer scientist and engineer at the Massachusetts Institute of Technology who was influential in bringing the sum of squares question into the applied realm.

Knowing whether a polynomial is always nonnegative might seem like a mathematical triviality. But a century after Hilbert asked his question, polynomial nonnegativity has turned out to answer applied problems that affect us all.

### The Best Way

Sum of squares meets the real world in the field of optimization. Optimization theory is concerned with finding the best way to do something amid constraints — like finding the best route to work given the current traffic conditions and a stop you need to make along the way. Scenarios like these can often be distilled into polynomial equations. In such cases, you solve, or “optimize” the scenario, by finding the minimum value taken by the polynomial.

Finding the minimum value of a polynomial with many variables is hard: There’s no straightforward high-school-style algorithm for computing the minimum value of complicated polynomials, and these same polynomials are not easy to graph.
Georgina Hall, a final-year graduate student at Princeton, collaborated on the new work.
Because the minimum value of a polynomial is hard to compute directly, researchers infer it by other means. And this is where nonnegativity, and the question of whether a polynomial is a sum of squares, comes in. “Certifying nonnegativity is really the heart of all optimization problems,” said Rekha Thomas, a mathematician at the University of Washington.

One way to find the minimum value is to ask yourself: What’s the most I can subtract from a nonnegative polynomial before it turns negative somewhere? In answering this question you might test different values — can I subtract 3 from the polynomial such that it’s still nonnegative? What about 4? Or 5? As you repeat this procedure, you’re interested in knowing at each step whether the polynomial is still nonnegative. And the way you check that is by checking whether the polynomial can still be expressed as a sum of squares.

“The thing you want to ask is, ‘Is the polynomial nonnegative?’ The problem is, answering nonnegativity is hard with more variables,” said Ahmadi. “That’s why we use sum of squares as a surrogate for nonnegativity.”

Once researchers know the minimum — which is, remember, the optimal value of the polynomial — they can use other methods to identify the inputs that lead to that value. Yet in order for nonnegativity to help solve optimization problems, you need a way of quickly computing whether a polynomial is equal to a sum of squares. And it took 100 years after Hilbert’s question for researchers to figure that out.

**Breaking Up the Problem**

Hilbert’s 17th question crossed from pure mathematics into real-world application around the year 2000. That’s when several different researchers figured out an algorithmic method for checking whether a polynomial is a sum of squares. They achieved this by translating the sum of squares question into a “semidefinite program,” which is a type of problem that computers know how to handle. This in turn made it possible for researchers in fields like computer science and engineering to use the power of nonnegativity to guide their search for optimal ways of solving problems.
Anirudha Majumdar leads the Intelligent Robot Motion Lab at Princeton University.
But semidefinite programming has a big limitation: It is slow on large problems and can’t handle many of the most complicated polynomials researchers really care about. Semidefinite programming can be used to find a sum of squares decomposition for polynomials with a handful to about a dozen variables raised to powers no higher than about 6. The polynomials that characterize complex engineering problems — like how to ensure a humanoid robot stays on its feet — can involve 50 or more variables. A semidefinite program could chew on that kind of polynomial until the end of time and still not return a sum of squares answer.

In a paper posted online last June, Ahmadi and Majumdar explain a way to get around the slowness of semidefinite programming. Instead of trying to find a sum of squares decomposition by solving a single, slow semidefinite program, they show how to do it using a sequence of simpler problems that are much faster to compute.

These types of problems are called “linear programs,” and they were developed in the 1940s to answer optimization problems related to the war effort. Linear programs are now well understood and quick to solve. In their new work, Ahmadi and Majumdar show that you can solve many linked linear programs (or, in some cases, another kind of problem known as a second-order cone program) and combine the results to get an answer that’s almost as good as the answer you could get with a semidefinite program. The upshot is that engineers have a new, practical tool that they can use to test for nonnegativity and find sum of squares decompositions quickly.

“We looked at a number of problems from robotics and control theory and demonstrated that the quality of solution we were getting was still useful in practice and much quicker to compute,” said Majumdar.

**Proof of Safety**

Speed of solution means everything when you’re in a self-driving car. And in that situation, a polynomial can serve as a kind of mathematical barrier around obstacles you don’t want to hit — if you can find it fast enough.

Imagine a simple example: a self-driving car in a giant parking lot. There’s nothing in the lot except for a guard booth at the far end. Your goal is to program the car so that it will never drive into the booth.

In this case, you’d start by putting a coordinate grid on the lot. Now make a polynomial that takes points on the grid as inputs. Make sure that the value of the polynomial at the location of your car is negative, and the value at the location of the guard booth is positive.

At some set of points between your car and the booth, the polynomial will cross over from negative to positive. Since your car is allowed to be on points only where the polynomial is negative, these points form something like a wall.

“If I start in a certain location, I’m not going to cross to the other side of the line where the obstacle is. This gives you a formal proof of safety for collision avoidance,” said Ahmadi.

Now, it’s no good if this wall is midway between the car and the booth. You want to craft your polynomial so that the wall hugs the obstacle as closely as possible. This fences off the guard booth while giving the car plenty of space to move around.

In practice, you want to minimize a value — the distance between the wall and the booth — and so you shift the graph of the polynomial around to see how far you can push it before it ceases to be
nonnegative. And you’re probing that line by testing whether the shifted polynomial remains a sum of squares.

A near-empty parking lot is one thing. But in realistic driving scenarios, a car’s sensors continually identify new and shifting obstacles — cars, bikes, kids. Every time a new obstacle appears, or an existing one moves, the car has to come up with elaborate new polynomials to fence them off. That’s a lot of sum of squares checks to do on the fly.

Seven years ago a different pair of researchers imagined that it might be possible to use such polynomial techniques to segregate autonomous cars from places they shouldn’t go. But at the time, computational speed made the idea a pipe dream.

Ahmadi and Majumdar’s new approach provides a way for carrying out such rapid-fire calculations. So, if and when self-driving cars are able to navigate the world safely, we’ll have Google and Tesla to thank — and also David Hilbert.