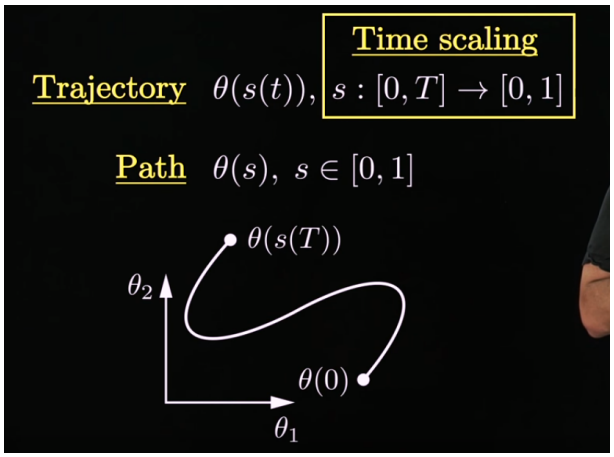
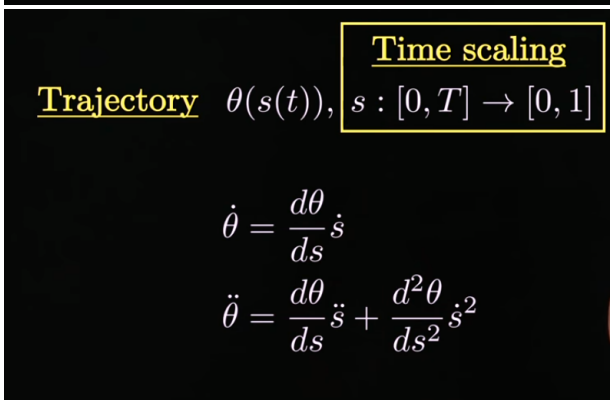
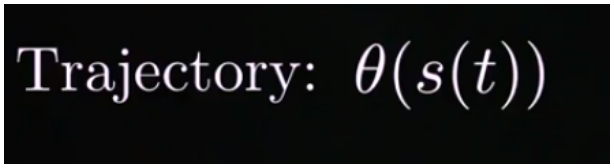
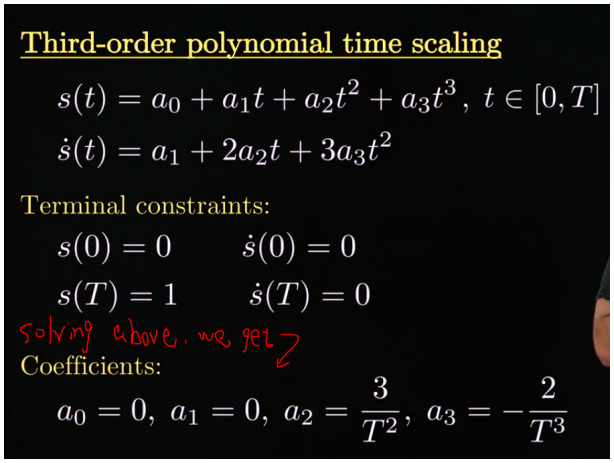


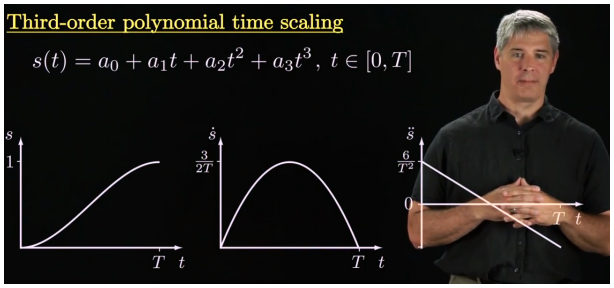


I. THE COMMON TIME SCALING TO MAKE THE PATH INTO TRAJECTORY

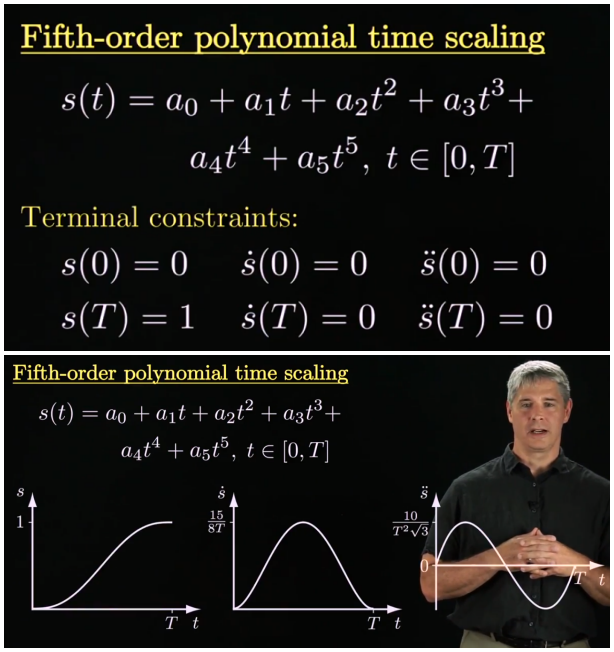


A. *Third order polynomial time scaling*

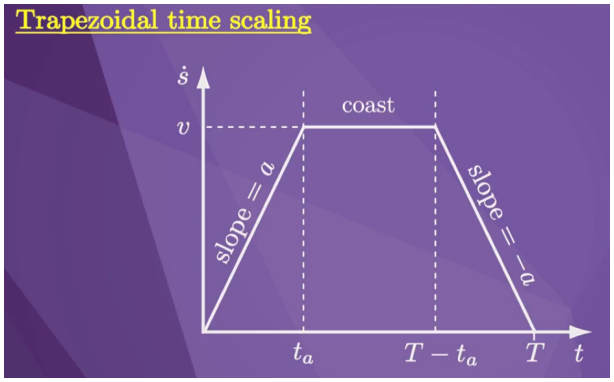




B. Figth order ploynomial time scaling

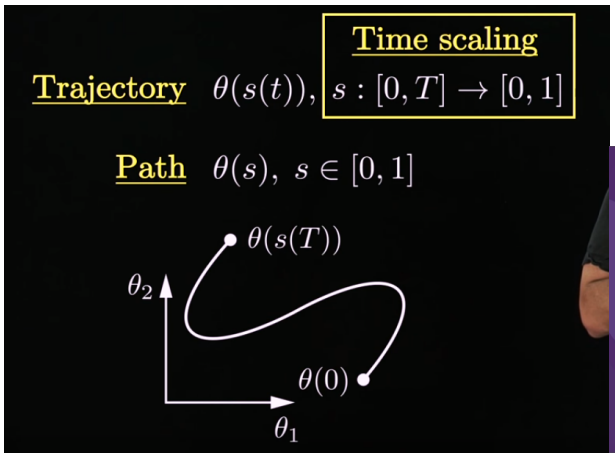


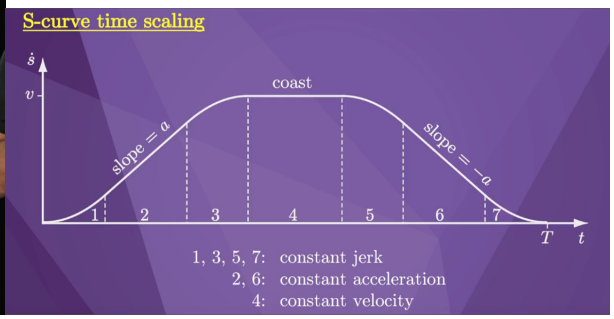
C. Trapezoidal time scaling



D. S-curve time scaling

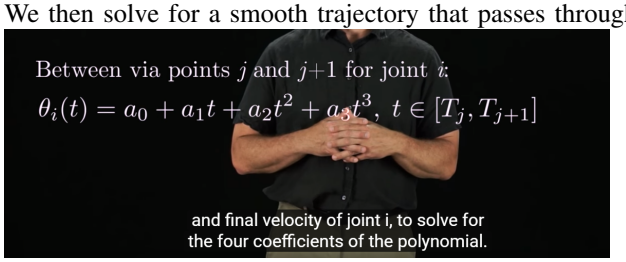
jerk is the time derivative of acceleration



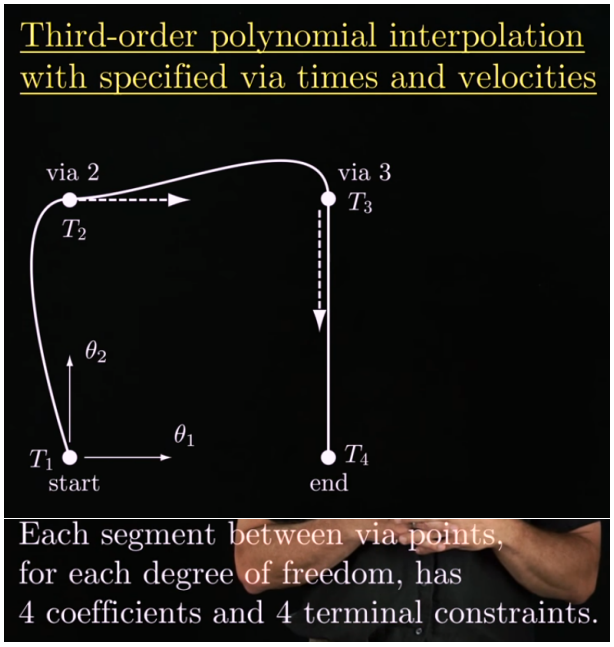


II. Via Point Trajectories.

If we want more flexibility to design the shape of the path, as well as the speed with, which it is executed, we could specify a set of configurations through which we would like, the robot to transit.
These configurations are called via points.
We also specify the times at which the robot should achieve each of these via points.
We then solve for a smooth trajectory that passes through the via points at the specified.

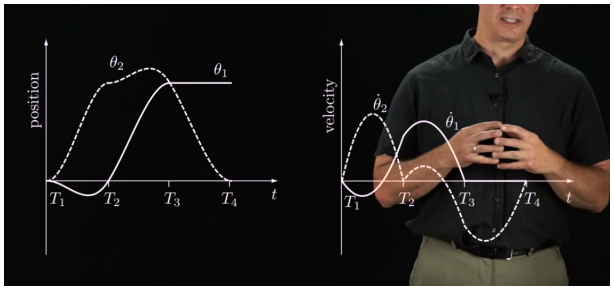


each via point has the time that robot passes through that via point the the velocity of that time, the velocity is indicated by the dashed arrows



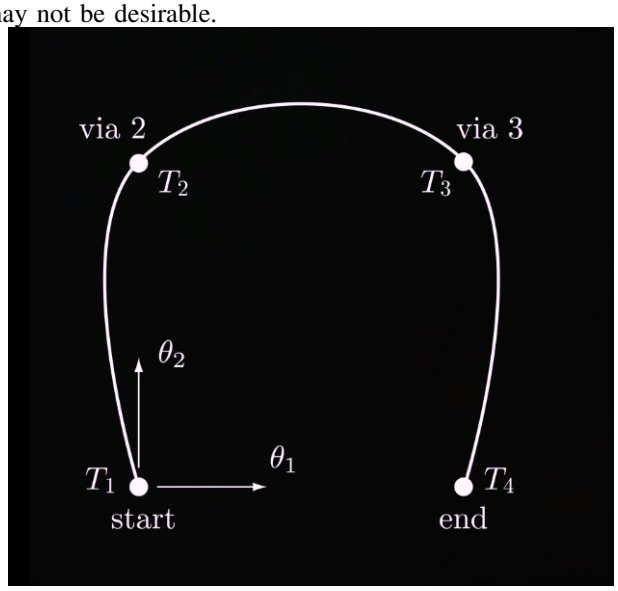
those coefficients and constraints allow us to solve the trajectory of the robot

The tangent of the path has to be aligned with the specified velocity at each via point, so we can use the velocities at the via points to change the shape of the path.



These time plots show the position and velocity of each joint during the trajectory.

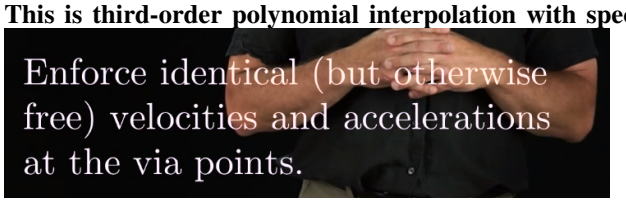
You can see that the positions and velocities are continuous at the via points, but the Discontinuity in the acceleration may not be desirable.



Therefore, another solution is to leave the velocities at the via points free, but to constrain the velocity before and after a via point,

and the acceleration before and after a via point, to be equal.

This is third-order polynomial interpolation with specified via times only.



There are other ways to shape a path using via points or control points.

In particular, with B-splines, the path does not pass exactly through the control points,

but the path is guaranteed to remain in the convex hull of the control points, unlike the path you see here.

This property ensures that the path does not violate joint limits.

III. Time-Optimal Time Scaling (Part 1 of 3)

In the next few videos, we consider the following problem:

Given a desired path theta-of-s,

find the time-optimal time scaling along this path,

considering the dynamics of the robot and torque limits at the robot joints.

Minimum-time motions can be used to maximize the productivity of a robot.

You could imagine trying to optimize other criteria, like the amount of energy consumed by the actuators, but in these next few videos we will focus on time-optimal trajectories.

A. $$M(\theta)\ddot{\theta} + \dot{\theta}^{\mathrm{T}}\Gamma(\theta)\dot{\theta} + g(\theta) = \tau$$

above the is the dynamics

$$\dot{\theta} = \frac{d\theta}{ds}\dot{s}$$

$$\ddot{\theta} = \frac{d\theta}{ds}\ddot{s} + \frac{d^2\theta}{ds^2}\dot{s}^2$$

then we have

$$\underbrace{\left(M(\theta(s))\frac{d\theta}{ds}\right)}_{m(s)\in\mathbb{R}^n}\ddot{s} + \underbrace{\left(M(\theta(s))\frac{d^2\theta}{ds^2} + \left(\frac{d\theta}{ds}\right)^{\mathrm{T}}\Gamma(\theta(s))\frac{d\theta}{ds}\right)}_{c(s)\in\mathbb{R}^n}\dot{s}^2 + \underbrace{g(\theta(s))}_{g(s)\in\mathbb{R}^n} = \tau$$

then we have

$$m(s)\ddot{s} + c(s)\dot{s}^2 + g(s) = \tau$$

dynamics on the path

Each of m-of-s, c-of-s, and g-of-s is a vector function of s,
where c-of-s times s-dot-squared is a velocity-product term,
g-of-s is the gravity term,
and m-of-s plays the role of a mass
This equation is the dynamics of the robot when it is restricted to move along the path theta-of-s
This equation says nothing about the dynamics when the robot is off the path.

B. to the joint vector theta, we have to consider the limits on the forces or torques that the robot's actuators can produce

The limits at the i'th joint can be written as

$$\tau_i^{\min}(\theta,\dot{\theta}) \le \tau_i \le \tau_i^{\max}(\theta,\dot{\theta})$$

For example, tau_i-min could be $-5N/M$ and tau_i-max could be $5N/M$.

But in general the limits are a function of theta and theta-dot.

In particular, the maximum torque that can be produced by an electric motor typically decreases as the velocity increases, until eventually it becomes zero.

and recall:

$$\dot{\theta} = \frac{d\theta}{ds}\dot{s}$$

we can rewrite the actuator limits as

$$\tau_i^{\min}(s,\dot{s}) \le \tau_i \le \tau_i^{\max}(s,\dot{s})$$

then we can get

$$\tau_i^{\min}(s,\dot{s}) \le m_i(s)\ddot{s} + c_i(s)\dot{s}^2 + g_i(s) \le \tau_i^{\max}(s,\dot{s})$$

The i'th actuator therefore places limits on the possible accelerations s-double-dot along the path when the robot is at the state (s, s-dot).

Then we get:

$$\text{if } m_i(s) > 0, \quad L_i(s,\dot{s}) = \frac{\tau_i^{\min}(s,\dot{s}) - c(s)\dot{s}^2 - g(s)}{m_i(s)}, \quad U_i(s,\dot{s}) = \frac{\tau_i^{\max}(s,\dot{s}) - c(s)\dot{s}^2 - g(s)}{m_i(s)}$$

$$\text{if } m_i(s) < 0, \quad L_i(s,\dot{s}) = \frac{\tau_i^{\max}(s,\dot{s}) - c(s)\dot{s}^2 - g(s)}{m_i(s)}, \quad U_i(s,\dot{s}) = \frac{\tau_i^{\min}(s,\dot{s}) - c(s)\dot{s}^2 - g(s)}{m_i(s)}$$

L_i, the lower limit on s-double-dot, and U_i, the upper limit on s-double-dot,

These equations tell us the maximum and minimum accelerations s-double-dot along the path

If we calculate L_i and U_i for all the joints, then L of (s, s-dot), the minimum feasible acceleration s-double-dot at the state (s, s-dot), is just the maximum of the lower limits

$$L(s,\dot{s}) = \max_i L_i(s,\dot{s})$$

Similarly, U of (s, s-dot),

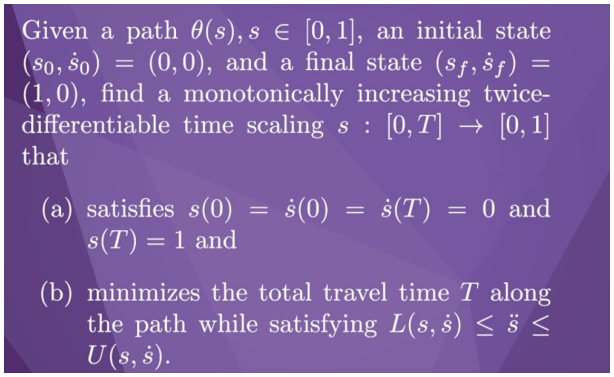
$$U(s,\dot{s}) = \min_i U_i(s,\dot{s})$$

We can now express the constraints on the robot's acceleration along the path compactly as s-double-dot is greater than L of (s, s-dot) and less than U of (s, s-dot).

$$L(s,\dot{s}) \le \ddot{s} \le U(s,\dot{s})$$

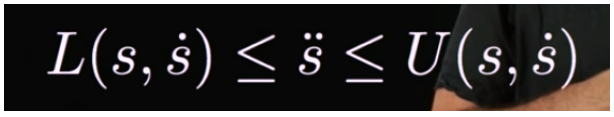
If $L(s,\dot{s}) > U(s,\dot{s})$, then there is no feasible acceleration $\ddot{s}$ that keeps the robot on the path at $(s,\dot{s})$.

we can mathematically express the time-optimal scaling problem as follows:

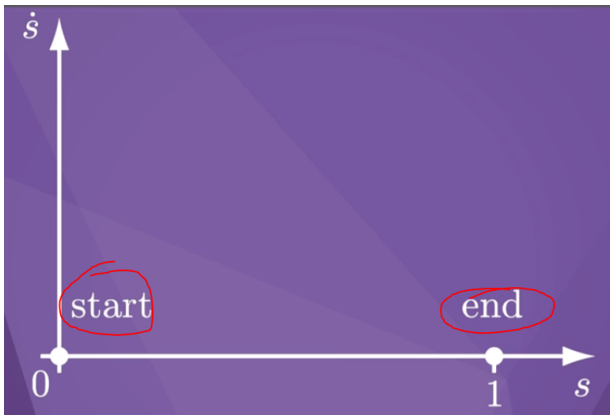


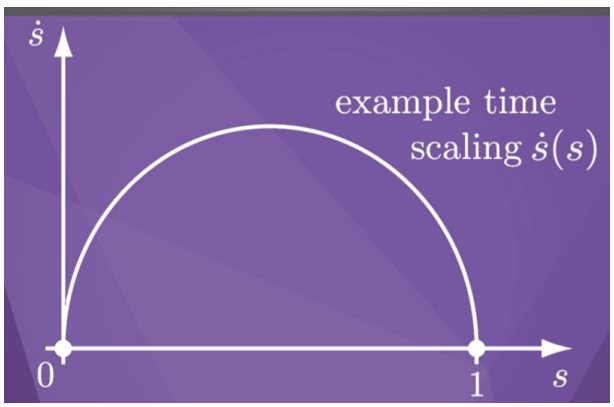
IV. Time-Optimal Time Scaling (Part 2 of 3)

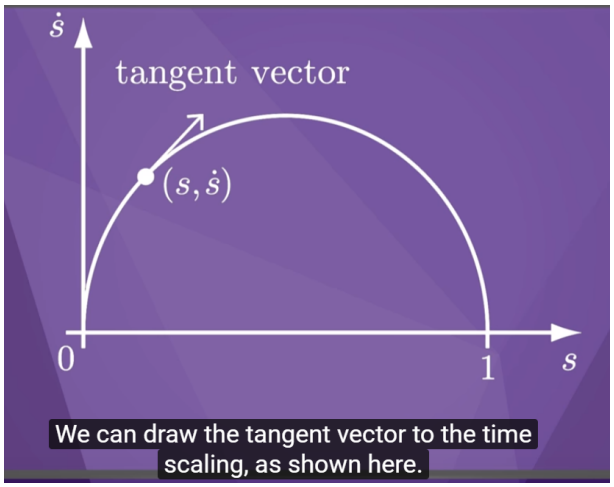
In the last video we learned to express the robot's joint force and torque limits as constraints on the feasible accelerations s-double-dot along the path theta-of-s, as a function of the state (s, s-dot):

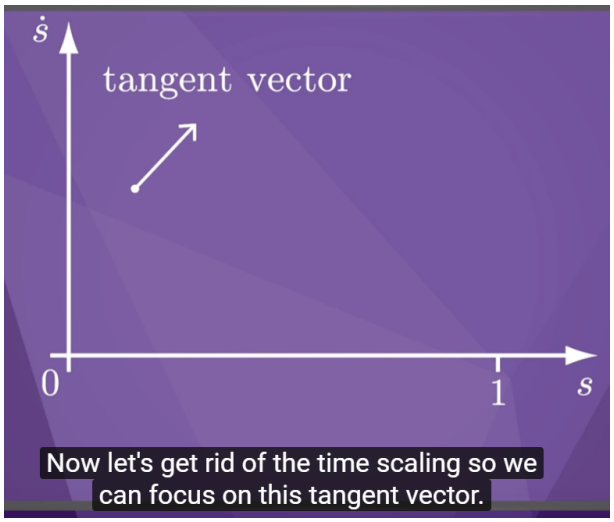


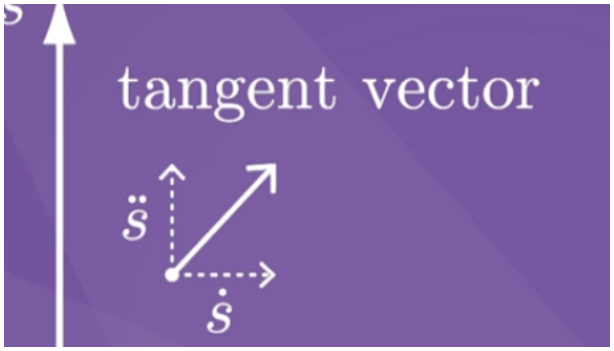
In this video, we'll express those constraints graphically and gain some insight into the time-optimal time-scaling problem









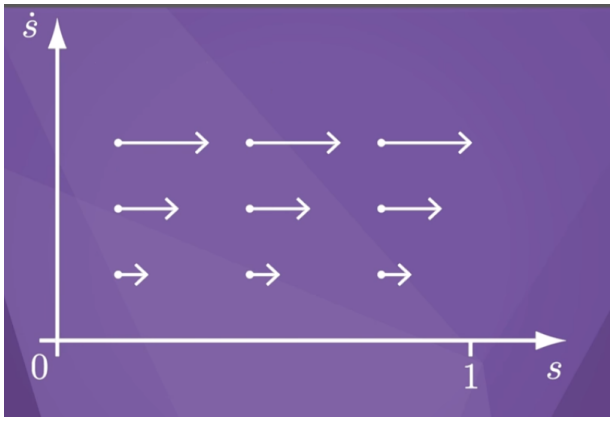


The tangent vector consists of a horizontal component and a vertical component.

The horizontal component expresses the rate of change of s, so it is just s-dot, which can be drawn as proportional to the height of the point along the s-dot axis

The vertical component expresses the rate of change of s-dot, in other words, the acceleration s-double-dot

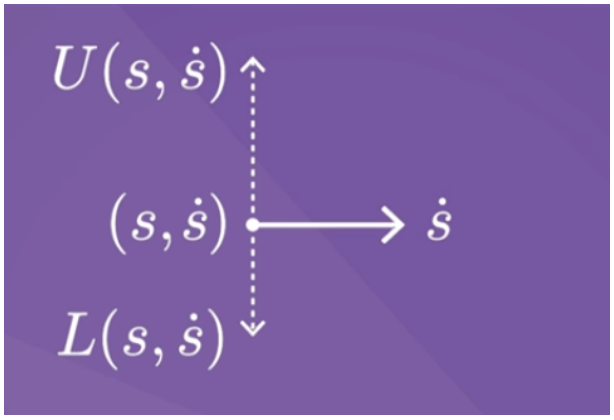
If we assumed that s-double-dot is always zero, then the tangent vectors at states would look like this:



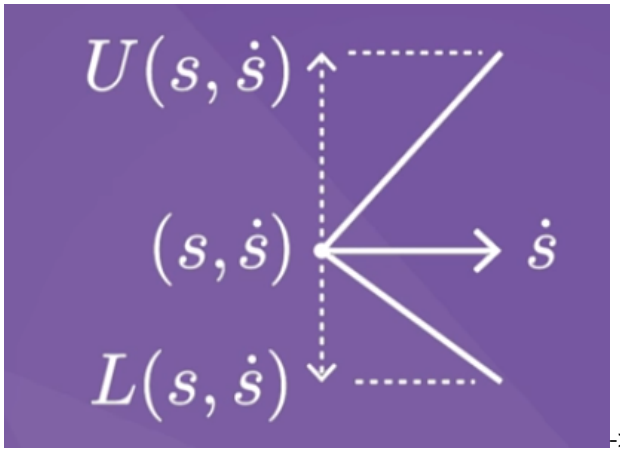
The horizontal component of a vector is determined by the s-dot value of the point.

Now let's focus on one particular tangent vector at the state (s, s-dot):

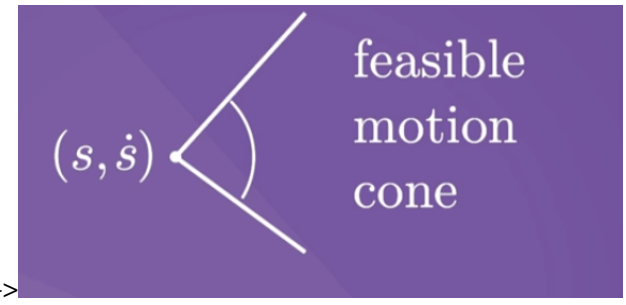
But now let's assume that the vertical component, the acceleration s-double-dot, can be any value in the range from L of (s, s-dot) to U of (s, s-dot), the range of feasible accelerations



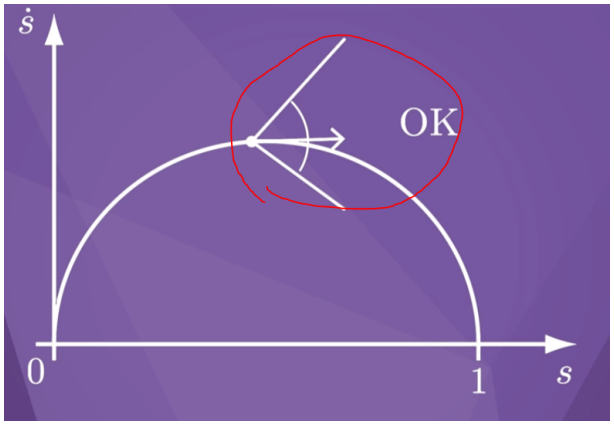
Summing these vertical vectors with the horizonal vector, we get the vectors shown here.



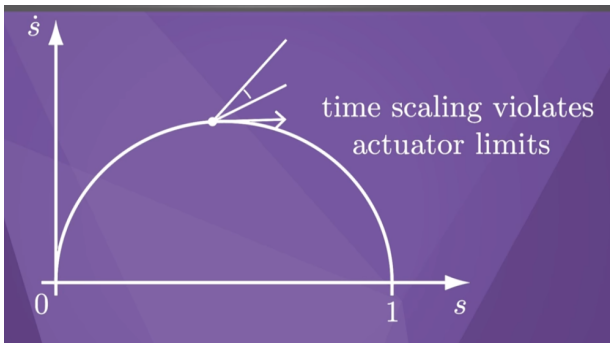
->



At this state (s, s-dot), the tangent vector to the time scaling must be inside this cone to satisfy the actuator limits.

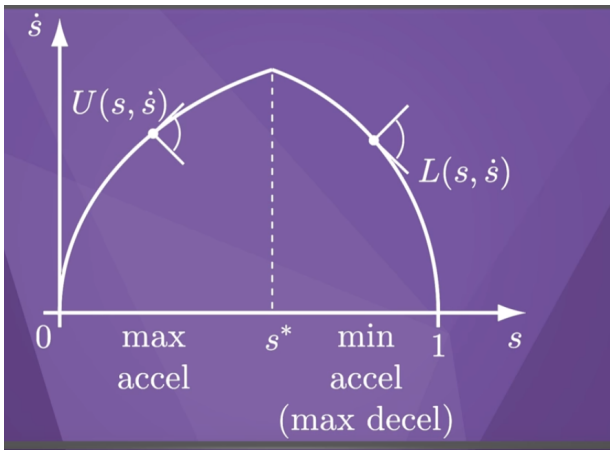


Therefore, a time scaling like this would be OK at this state, as the tangent vector lies inside the feasible motion cone.



If, instead, our feasible motion cone looked like this, the tangent vector is outside the cone, and this time scaling is not possible according to the robot's actuator limits.

You could imagine drawing the motion cone at every point in the plane, and the problem is to get from the start state to the goal state as quickly as possible while keeping the tangent to the time scaling inside all motion cones along the curve.



It is clear why this is time optimal: in the first segment, the robot cannot go any faster,

and in the second segment, if the speed s-dot were any higher at any given s, the robot would not be able to come to a stop

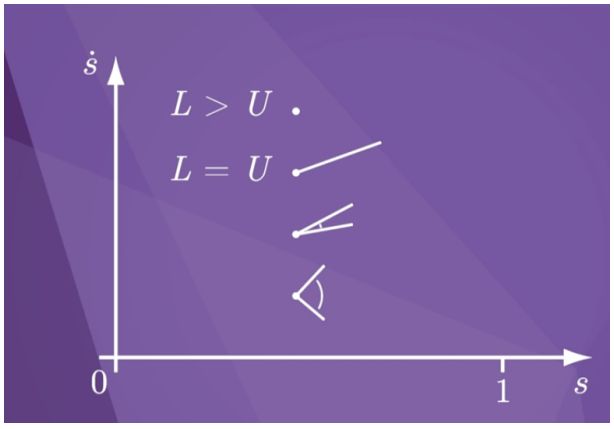
This time scaling keeps the speed s-dot as high as possible at all times, and therefore the duration of the motion is as short as possible.

This is called the:

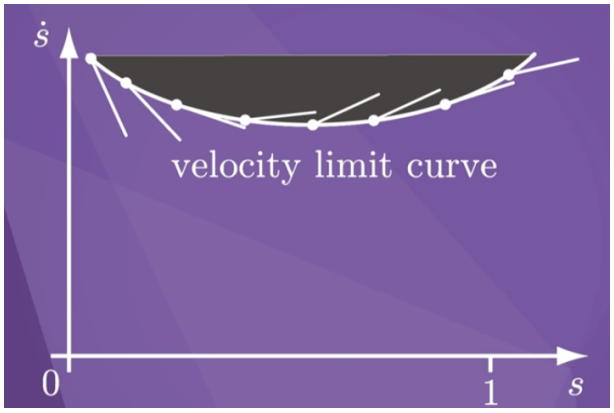
If you keep s constant but increase s-dot, you get a different motion cone.

If you increase s-dot further, then the motion cone may reduce to a single vector, where the lower acceleration limit is the same as the upper acceleration limit.

If you increase s-dot further, then no motions are feasible, and this means the robot is traveling too fast for the actuators to keep the robot on the path.:



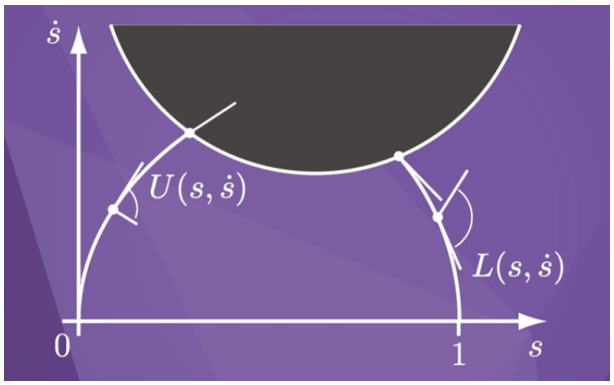
In general, we could plot a velocity limit curve: at states on this curve, only a single acceleration is possible, and at states above this curve, the robot leaves the path immediately:



Now, considering the existence of a speed limit, we might end up with a situation as illustrated here: the maximum acceleration curve and the maximum deceleration curve do not intersect, but instead run into the velocity limit curve.

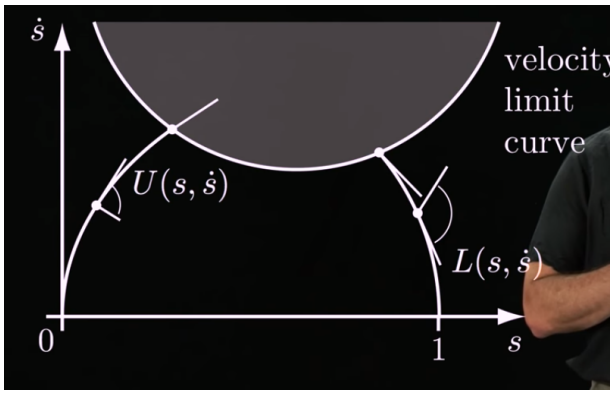
Therefore, bang-bang control is not possible.

What to do in this case is the subject of the next, and final, video of Chapter 9.

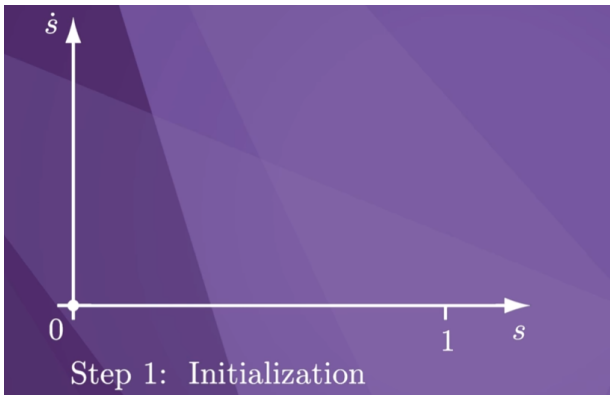


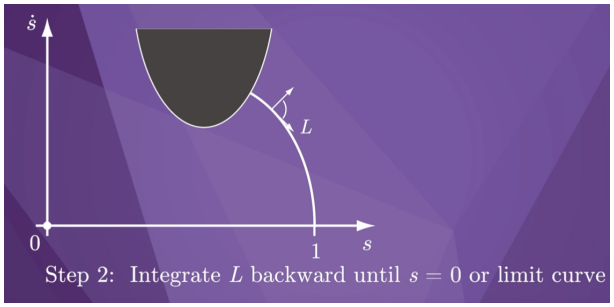
V. Time-Optimal Time Scaling (Part 3 of 3)

this section is the solution of solving the problem of :

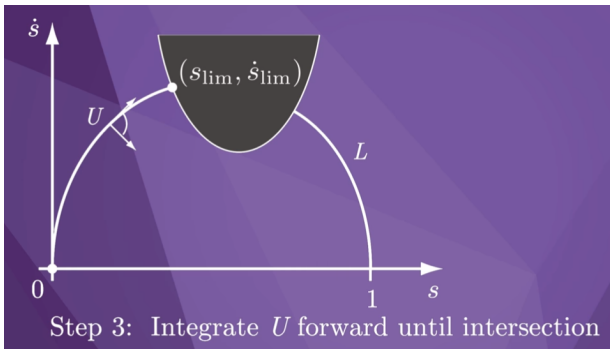


which bangbang cannot handle



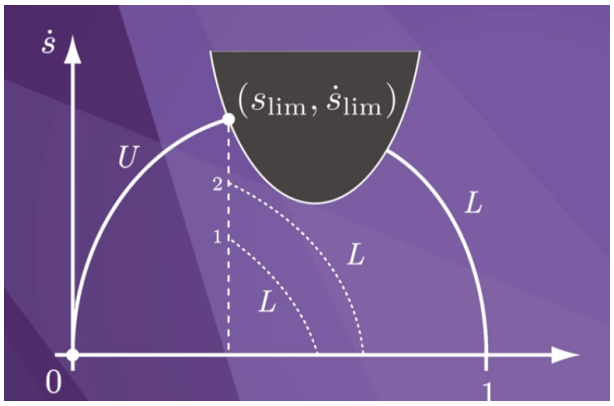


the figure shows above is integrate L backward until reach the limit curve

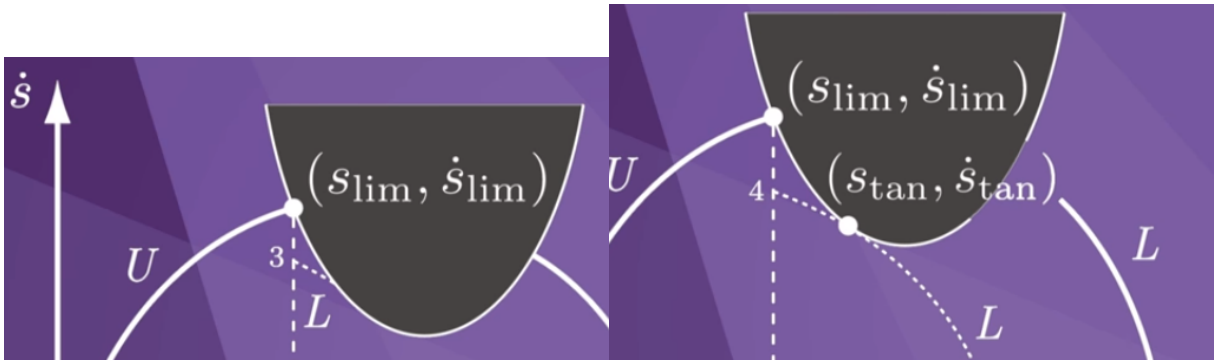


Then we need to find a point where we switch from U to L.

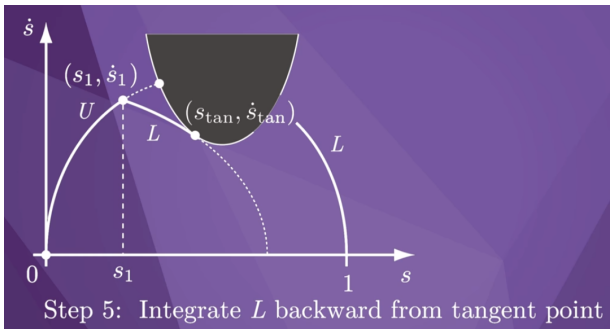
One way to find the switch point is to decrease the velocity s-dot from the point where we hit the velocity limit curve, and to integrate forward the minimum acceleration L. Depending on how much we decrease s-dot, the forward integration will either hit the s-axis, as it does with our first two guesses:



or it will intersect the velocity limit curve again, as with our third guess; or it will just touch the velocity limit curve tangentially.

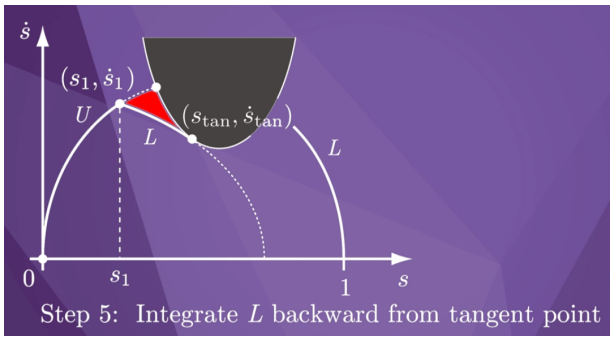


It is this tangential point we are looking for, and we'll call this point (s_tan, s_tan-dot).

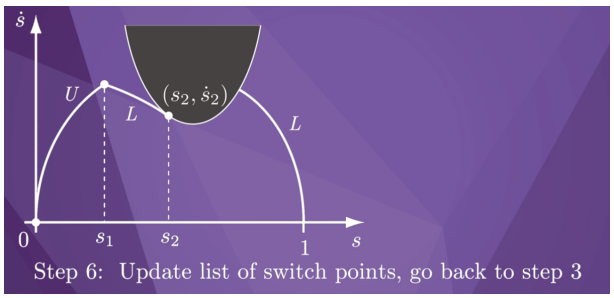


In step 5, we integrate the minimum acceleration L backward from the tangent point until it intersects the previous U segment.

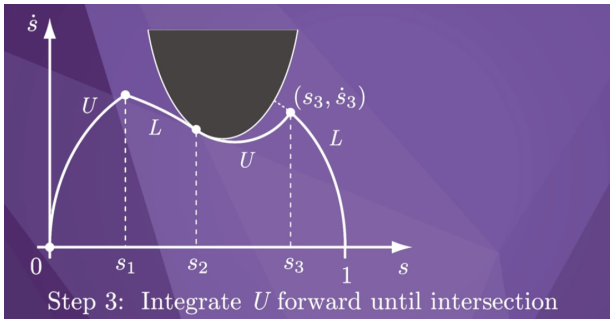
This is the switch point s_1, from maximum acceleration to minimum acceleration.



States in the region shaded red would eventually collide with the velocity limit curve, so they have to be avoided.



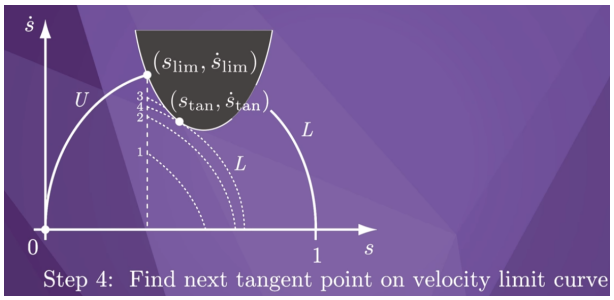
In step 6, we mark the tangent point as the switch point s_2, where we switch again from minimum acceleration L to maximum acceleration U



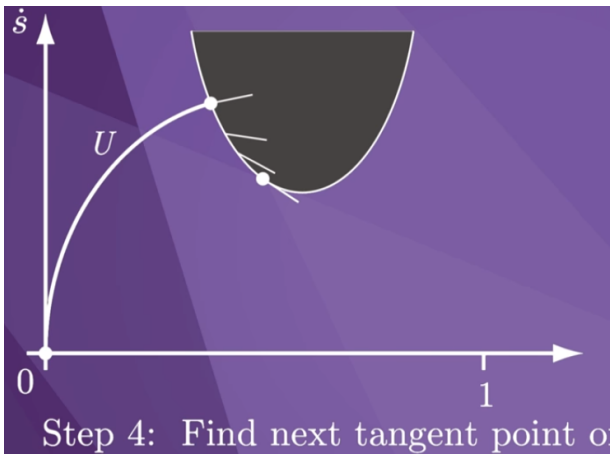
We then go back to step 3, where we integrate U forward again. This segment will either intersect the velocity limit curve again, in which case we repeat the process just described, or it will intersect the final L segment, and the algorithm is complete.

In the figure shown, the algorithm completes with three switching points, and the time-optimal time scaling consists of maximum acceleration until s_1, followed by minimum acceleration until s_2, followed by maximum acceleration until s_3, followed by minimum acceleration This time scaling keeps the velocity as high as possible at all points on the path while assuring the trajectory is feasible for the actuators.

A key step in this algorithm is step 4, finding the next tangent point on the velocity limit:



Instead of using a binary search guess-and-check approach, a more computationally efficient approach is to numerically construct the velocity limit curve and search for the next point where the motion ray is tangent to the curve.

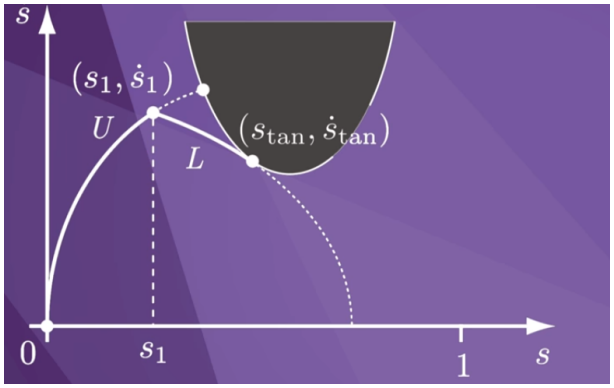


As shown above:

Clearly, at the point of intersection the motion ray is into the region of inadmissible

As we search along the velocity limit curve, we eventually find a point on the curve where the motion ray is tangent to the limit curve

We can now proceed with the algorithm as before:


$\dot{s}$
$(s_1, \dot{s}_1)$
U
L
$(s_{\tan}, \dot{s}_{\tan})$
0
s_1
1
s