This document is meant to explain the theory and rationale behind the code found in the `BezierSpline.recalculate` function. Understanding the math requires some basic calculus and linear algebra as well as good spacial reasoning. Given that, let's start from the top.

Our goal here is to create a mathematical description for a spline. The reason I started this project was to draw a 2-dimensional spline to try to represent hills, so the ultimate goal and the examples given will be in that vein, but there shouldn't be much in the way of understanding other uses (pathing in a game, for instance). A spline is a piecewise polynomial parametric curve that goes through specified points, which we call knots. Generally, we want these curves to look natural, so a linear solution is not satisfying. The Bezier part of "Bezier Spline" is a Bezier curve. Specifically, a cubic Bezier curve. These curves are pretty well-known in computing, and it's not uncommon that they're supported in graphics software, which means it would be pretty easy to turn the result of this spline into something visible. So, the first thing to do is define a cubic Bezier curve, and then we'll worry about connecting these curves.

A cubic Bezier curve is defined by

$$\mathbf{B}(t) = \mathbf{P}_0(1-t)^3 + 3\mathbf{P}_1(1-t)^2t + 3\mathbf{P}_2(1-t)t^2 + \mathbf{P}_3t^3$$

and lives on the interval $t \in [1, 0]$. $\mathbf{P}_0$ is the start point of the curve, and $\mathbf{P}_3$ is the end point of the curve. $\mathbf{P}_1$ and $\mathbf{P}_2$ are the control points of the curve, and do not lie on the curve.

We're building a spline from these bezier curves, so naturally the endpoints of each curve should be the knots of the spline, and the endpoint of the previous curve should be the start point for the next curve. The spline will be described by n knots K_i where $i \in \{1, 2, \ldots, n\}$. From this, I'll describe the ith curve of the spline as

$$\mathbf{B}_i(t) = \mathbf{K}_i(1-t)^3 + 3\mathbf{P}_{1,i}(1-t)^2t + 3\mathbf{P}_{2,i}(1-t)t^2 + \mathbf{K}_{i+1}t^3 \tag{1}$$

where $i \in \{1, 2, \ldots, n-1\}$

We want this spline to be C^2 continuous, meaning the first and second derivatives are equal at the transition between curves (much more on continuity later), so we need the first and second derivatives.

$$\mathbf{B}'_i(t) = -3\mathbf{K}_i(1-t)^2 + 3\mathbf{P}_{1,i}(1-4t+3t^2) + 3\mathbf{P}_{2,i}(2-3t)t + 3\mathbf{K}_{i+1}t^2 \tag{2}$$

$$\mathbf{B}''_i(t) = 6\mathbf{K}_i(1-t) + 6\mathbf{P}_{1,i}(-2+3t) + 6\mathbf{P}_{2,i}(1-3t) + 6\mathbf{K}_{i+1}t \tag{3}$$

The next thing we want to do is to constrain everything to be equal at the endpoints. We already know the position is the same at the endpoints because those are our knots, So first we find the values of the derivatives at the endpoints:

$$\begin{aligned}\mathbf{B}'_i(0) &= -3\mathbf{K}_i + 3\mathbf{P}_{1,i}\\ \mathbf{B}'_i(1) &= -3\mathbf{P}_{2,i} + 3\mathbf{K}_{i+1}\end{aligned}$$

$$\begin{aligned}\mathbf{B}''_i(0) &= 6\mathbf{K}_i - 12\mathbf{P}_{1,i} + 6\mathbf{P}_{2,i}\\ \mathbf{B}''_i(1) &= 6\mathbf{P}_{1,i} - 12\mathbf{P}_{2,i} + 6\mathbf{K}_{i+1}\end{aligned}$$

And then we set them equal to each other:

$$\begin{aligned}\mathbf{B}'_i(0) &= \mathbf{B}'_{i-1}(1)\\ -3\mathbf{K}_i + 3\mathbf{P}_{1,i} &= 3\mathbf{K}_i - 3\mathbf{P}_{2,i-1}\\ \mathbf{P}_{2,i-1} &= 2\mathbf{K}_i - \mathbf{P}_{1,i} \qquad (4)\end{aligned}$$

$$\begin{aligned}\mathbf{B}''_i(0) &= \mathbf{B}''_{i-1}(1)\\ 6\mathbf{K}_i - 12\mathbf{P}_{1,i} + 6\mathbf{P}_{2,i} &= 6\mathbf{P}_{1,i-1} - 12\mathbf{P}_{2,i-1} + 6\mathbf{K}_i\\ -\mathbf{P}_{1,i-1} + 2\mathbf{P}_{2,i-1} - 2\mathbf{P}_{1,i} + \mathbf{P}_{2,i} &= \mathbf{0} \qquad (5)\end{aligned}$$

Now, we have equations (4) and (5) that describe constraints on each curve. Remember, we already know the knots, so this problem is about solving for all the $\mathbf{P}_1$'s and $\mathbf{P}_2$'s. You'll notice that equation (4) is particularly handy if we happen to have all the $\mathbf{P}_1$'s, so let's try solving for just those for now. We'll need an equation containing only $\mathbf{P}_1$'s and known values, so let's do some substitution.

$$\begin{aligned}
-\mathbf{P}_{1,i-1} + 2\mathbf{P}_{2,i-1} - 2\mathbf{P}_{1,i} + \mathbf{P}_{2,i} &= \mathbf{0}\\
-\mathbf{P}_{1,i-1} + 2(2\mathbf{K}_i - \mathbf{P}_{1,i}) - 2\mathbf{P}_{1,i} + (2\mathbf{K}_{i+1} - \mathbf{P}_{1,i+1}) &= \mathbf{0}\\
\mathbf{P}_{1,i-1} + 4\mathbf{P}_{1,i} + \mathbf{P}_{1,i+1} &= 4\mathbf{K}_i + 2\mathbf{K}_{i+1} \qquad (6)
\end{aligned}$$

Equation (6) is going to be very useful in creating a matrix. We have $n-1$ unknowns (for $n-1$ curves), and equation (6) is actually $n-3$ valid equations when you vary i. Where we can't use equation (6) is the first and last row of our matrix; if $i = 1$, we would need to use the value of $\mathbf{P}_{1,0}$, which doesn't exist. The same thing with $i = n-1$ and $\mathbf{P}_{1,n}$. So we need constraints for the endpoints of the spline. Rather arbitrarily, we can impose the restriction that the second derivative must be equal to $\mathbf{0}$ at the endpoints. That is, $\mathbf{B}_1''(0) = \mathbf{B}_{n-1}''(1) = \mathbf{0}$. We'll use equation (4) again to turn $\mathbf{P}_2$'s into $\mathbf{P}_1$'s and $\mathbf{K}$'s.

$$\begin{aligned}
\mathbf{B}_1''(0) = 6\mathbf{K}_1 - 12\mathbf{P}_{1,1} + 6\mathbf{P}_{2,1} &= \mathbf{0}\\
\mathbf{K}_1 - 2\mathbf{P}_{1,1} + \mathbf{P}_{2,1} &= \mathbf{0}\\
\mathbf{K}_1 - 2\mathbf{P}_{1,1} + (2\mathbf{K}_2 - \mathbf{P}_{1,2}) &= \mathbf{0}\\
2\mathbf{P}_{1,1} + \mathbf{P}_{1,2} &= \mathbf{K}_1 + 2\mathbf{K}_2 \qquad (7)
\end{aligned}$$

And now we have a valid equation for the first row of our matrix.

The other endpoint is a little more involved. When we substitute (4) in, we'll end up with another $\mathbf{P}_{1,n}$. We'll solve for this one and plug it into equation (6) to get rid of it.

$$\begin{aligned}
\mathbf{B}_{n-1}''(1) = 6\mathbf{P}_{1,n-1} - 12\mathbf{P}_{2,n-1} + 6\mathbf{K}_n &= \mathbf{0}\\
\mathbf{P}_{1,n-1} - 2\mathbf{P}_{2,n-1} + \mathbf{K}_n &= \mathbf{0}\\
\mathbf{P}_{1,n-1} - 2(2\mathbf{K}_n - \mathbf{P}_{1,n}) + \mathbf{K}_n &= \mathbf{0}\\
2\mathbf{P}_{1,n} &= -\mathbf{K}_n + 4\mathbf{K}_n - \mathbf{P}_{1,n-1}\\
\mathbf{P}_{1,n} &= \frac{1}{2}(3\mathbf{K}_n - \mathbf{P}_{1,n-1})
\end{aligned}$$

$$\begin{aligned}
\mathbf{P}_{1,n-2} + 4\mathbf{P}_{1,n-1} + \mathbf{P}_{1,n} &= 4\mathbf{K}_{n-1} + 2\mathbf{K}_n\\
\mathbf{P}_{1,n-2} + 4\mathbf{P}_{1,n-1} + \frac{1}{2}(3\mathbf{K}_n - \mathbf{P}_{1,n-1}) &= 4\mathbf{K}_{n-1} + 2\mathbf{K}_n\\
\mathbf{P}_{1,n-2} + \frac{7}{2}\mathbf{P}_{1,n-1} &= 4\mathbf{K}_{n-1} + \frac{1}{2}\mathbf{K}_n \qquad (8)
\end{aligned}$$

And finally we get to a set of equations that can describe a tridiagonal matrix with (6), (7), and (8):

$$\begin{aligned}
2\mathbf{P}_{1,1} + \mathbf{P}_{1,2} &= \mathbf{K}_1 + 2\mathbf{K}_2\\
\mathbf{P}_{1,i-1} + 4\mathbf{P}_{1,i} + \mathbf{P}_{1,i+1} &= 4\mathbf{K}_i + 2\mathbf{K}_{i+1}, \text{for } i \in \{2, 3, \ldots, n-2\}\\
\mathbf{P}_{1,n-2} + \frac{7}{2}\mathbf{P}_{1,n-1} &= 4\mathbf{K}_{n-1} + \frac{1}{2}\mathbf{K}_n
\end{aligned}$$

This creates the following matrix equation:

$$\begin{pmatrix}
2 & 1 & & & & 0\\
1 & 4 & 1 & & & \\
 & 1 & 4 & 1 & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & 1 & 4 & 1\\
0 & & & & 1 & 7/2
\end{pmatrix}
\begin{pmatrix}
\mathbf{P}_{1,1}\\ \mathbf{P}_{1,2}\\ \vdots\\ \mathbf{P}_{1,n-1}
\end{pmatrix}
=
\begin{pmatrix}
\mathbf{K}_1 + 2\mathbf{K}_2\\
4\mathbf{K}_2 + 2\mathbf{K}_3\\
4\mathbf{K}_3 + 2\mathbf{K}_4\\
\vdots\\
4\mathbf{K}_{n-2} + 2\mathbf{K}_{n-1}\\
4\mathbf{K}_{n-1} + \frac{1}{2}\mathbf{K}_n
\end{pmatrix}$$

This type of matrix is called a tridiagonal matrix, and that's good because we can use the Thomas algorithm to solve it in $O(n)$ operations rather than the $O(n^3)$ operations Gaussian elimination takes.

Once we have all the $\mathbf{P}_1$s, we need to get the $\mathbf{P}_2$s, which is relatively trivial. For $i \in \{1, 2, \ldots, n-1\}$, we can use equation (4). We can't use equation (4) for $\mathbf{P}_{2,n-1}$ because $\mathbf{P}_{1,n}$ doesn't exist. Instead, we'll use what follows from our constraint that the curve's second derivative is equal to $\mathbf{0}$ at the end of the spline.

$$\begin{aligned}
\mathbf{P}_{1,n-1} - 2\mathbf{P}_{2,n-1} + \mathbf{K}_n &= \mathbf{0} \\
\mathbf{P}_{1,n-1} + \mathbf{K}_n &= 2\mathbf{P}_{2,n-1} \\
\frac{1}{2}(\mathbf{P}_{1,n-1} + \mathbf{K}_n) &= \mathbf{P}_{2,n-1} \qquad (9)
\end{aligned}$$

Now we've got all the control points for every segment of our spline. Worth noting is that this type of spline is a global problem. Move one knot, and the whole spline has to be recalculated. This is in contrast to something like a B-spline, which is a local problem.

Now, the above solution is mathematically pretty, and C^2 continuity is all you'll need for most applications. I'll actually argue though that it's too much in some cases. In animation, splines might be used for smoothly moving a component from one location to another with stops in between. In this sort of case, C^2 continuity is necessary or else things look weird and jumpy. In these sorts of instances, t represents time, and it takes 1 unit of time to traverse from one knot to another. We'll simplify that use case to a particle moving along a path. In graphical use cases, time isn't relevant; only the shape of the curve. So for particles, we care that the first derivative of the path (velocity) is equivalent at the knots. For graphics, we only want the curve to be pointing in the same direction at the knots. That's where G^1 continuity comes in.

Let's take a set of knots where $\mathbf{K}_1$ and $\mathbf{K}_2$ are far apart relative to $\mathbf{K}_2$ and $\mathbf{K}_3$. It takes the particle 1 second to get to $\mathbf{K}_2$, so it needs to accelerate a good amount and get going pretty fast. Once it hits $\mathbf{K}_2$, it's got a lot of speed, but now it has to take an entire second to get to $\mathbf{K}_3$. Typically, this means it'll take a loop and come back around to $\mathbf{K}_3$. You can see this in the math pretty easily too. If we go back to equation (4) and rewrite it a bit

$$\begin{aligned}\mathbf{P}_{2,i-1} &= 2\mathbf{K}_i - \mathbf{P}_{1,i}\\ \mathbf{P}_{1,i} &= 2\mathbf{K}_i - \mathbf{P}_{2,i-1}\\ \mathbf{P}_{1,i} &= \mathbf{K}_i + (\mathbf{K}_i - \mathbf{P}_{2,i-1})\end{aligned}$$

Our first control point is as far from the first knot as the previous second control point was, so the curve gets pulled far from the first knot. But the path needs to get back to the second knot, and if the second knot is "behind" the first control point, it does a loop. If we're in animation land or particle land, it's good that the particle doesn't instantaneously slow down between knots. To solve the problem, we need to adjust our knots. In graphics land, we didn't want a loop; we wanted our spline to make a nice smooth-looking curve that happens to have to stop through two nearby points. So, for the graphics use case, we'll relax our C^2 continuity to G^2 continuity, and use that to pull the control points back in. Where C^1 continuity requires the first derivatives at the connections to be identical, G^1 continuity only requires the tangent vectors to point in the same direction. C^2 continuity requires C^1 continuity and that the second derivatives at the connections be identical while G^2 continuity requires G^1 continuity and that curvature is continuous, which also means the second derivatives point in the same direction. This also means that C^2 continuity is G^2 continuous, so we can work off what we've already done.

To start, we'll rewrite our constraints to be more forgiving:

$$\begin{aligned}c_i\mathbf{B}_i'(0) &= \mathbf{B}_{i-1}'(1)\\ d_i\mathbf{B}_i''(0) &= \mathbf{B}_{i-1}''(1)\end{aligned}$$

Continuous tangency is satisfied here because the first derivatives are just scalar multiples of each other, so we can choose c_i to be anything we want. It can even be different for every i. d_i could also be whatever we want, but there's a relationship between c_i and d_i that we must enforce to keep curvature constant. We'll use the definition of curvature to find d_i in terms of c_i.

Curvature, κ, is defined in terms of t as

$$\kappa(t) = \frac{\|\mathbf{T}'(t)\|}{\|\mathbf{B}'(t)\|}$$

Where

$$\mathbf{T}(t) = \frac{\mathbf{B}'(t)}{\|\mathbf{B}'(t)\|}$$

Using the quotient rule, we can find that

$$\mathbf{T}'(t) = \frac{\mathbf{B}''(t)}{\|\mathbf{B}'(t)\|} - \frac{(\mathbf{B}'(t)\cdot\mathbf{B}''(t))\mathbf{B}'(t)}{\|\mathbf{B}'(t)\|^3}$$

(The last page of this document details finding $\frac{d}{dt}[\|\mathbf{x}(t)\|]$.)
Which lets us write κ in terms of derivatives of $\mathbf{B}$:

$$\kappa_i(t) = \frac{\mathbf{B}_i''(t)}{\|\mathbf{B}_i'(t)\|^2} - \frac{(\mathbf{B}_i'(t)\cdot\mathbf{B}_i''(t))\mathbf{B}_i'(t)}{\|\mathbf{B}_i'(t)\|^4}$$

If our spline is to be G^2 continuous, we have the condition

$$\kappa_i(0) = \kappa_{i-1}(1)$$

We'll use this to find a relationship between c_i and d_i.

$$\begin{gathered}
\kappa_i(0) = \kappa_{i-1}(1) \\
\frac{\mathbf{B}_i''(0)}{\|\mathbf{B}_i'(0)\|^2} - \frac{(\mathbf{B}_i'(0)\cdot\mathbf{B}_i''(0))\mathbf{B}_i'(0)}{\|\mathbf{B}_i'(0)\|^4} = \frac{\mathbf{B}_{i-1}''(1)}{\|\mathbf{B}_{i-1}'(1)\|^2} - \frac{(\mathbf{B}_{i-1}'(1)\cdot\mathbf{B}_{i-1}''(1))\mathbf{B}_{i-1}'(1)}{\|\mathbf{B}_{i-1}'(1)\|^4} \\
\frac{\mathbf{B}_i''(0)}{\|\mathbf{B}_i'(0)\|^2} - \frac{(\mathbf{B}_i'(0)\cdot\mathbf{B}_i''(0))\mathbf{B}_i'(0)}{\|\mathbf{B}_i'(0)\|^4} = \frac{d_i\mathbf{B}_i''(0)}{\|c_i\mathbf{B}_i'(0)\|^2} - \frac{((c_i\mathbf{B}_i'(0))\cdot(d_i\mathbf{B}_i''(0)))(c_i\mathbf{B}_i'(0))}{\|c_i\mathbf{B}_i'(0)\|^4} \\
\frac{\mathbf{B}_i''(0)}{\|\mathbf{B}_i'(0)\|^2} - \frac{(\mathbf{B}_i'(0)\cdot\mathbf{B}_i''(0))\mathbf{B}_i'(0)}{\|\mathbf{B}_i'(0)\|^4} = \frac{d_i\mathbf{B}_i''(0)}{c_i^2\|\mathbf{B}_i'(0)\|^2} - \frac{c_i^2 d_i(\mathbf{B}_i'(0)\cdot\mathbf{B}_i''(0))\mathbf{B}_i'(0)}{c_i^4\|\mathbf{B}_i'(0)\|^4} \\
\|\mathbf{B}_i'(0)\|^2\mathbf{B}_i''(0) - (\mathbf{B}_i'(0)\cdot\mathbf{B}_i''(0))\mathbf{B}_i'(0) = \frac{d_i}{c_i^2}\|\mathbf{B}_i'(0)\|^2\mathbf{B}_i''(0) - \frac{d_i}{c_i^2}(\mathbf{B}_i'(0)\cdot\mathbf{B}_i''(0))\mathbf{B}_i'(0) \\
c_i^2(\|\mathbf{B}_i'(0)\|^2\mathbf{B}_i''(0) - (\mathbf{B}_i'(0)\cdot\mathbf{B}_i''(0))\mathbf{B}_i'(0)) = d_i(\|\mathbf{B}_i'(0)\|^2\mathbf{B}_i''(0) - (\mathbf{B}_i'(0)\cdot\mathbf{B}_i''(0))\mathbf{B}_i'(0)) \\
c_i^2 = d_i
\end{gathered}$$

So whatever we decide to set our constants to, that relationship must hold for our spline to be G^2 continuous.

Our ultimate goal is to make sure if we come into a spline too fast, we don't overshoot it. In that sense, we should reduce our speed if the next spline is short in comparison to the previous spline. To that end, we'll try

$$c_i = \frac{w_{i-1}}{w_i} = \frac{\|\mathbf{K}_i - \mathbf{K}_{i-1}\|}{\|\mathbf{K}_{i+1} - \mathbf{K}_i\|}$$

And our final constraints become

$$\begin{gathered}
c_i\mathbf{B}_i'(0) = \mathbf{B}_{i-1}'(1) \\
c_i^2\mathbf{B}_i''(0) = \mathbf{B}_{i-1}''(1)
\end{gathered}$$

Now that we have our constraints fully defined, we basically just do the same as before. The first step then is to find a replacement for equation (4)

$$\mathbf{P}_{2,i-1} = \mathbf{K}_i - c_i(\mathbf{P}_{1,i} - \mathbf{K}_i) \tag{10}$$

And a replacement for equation (5):

$$\begin{aligned}
c_i^2(\mathbf{K}_i - 2\mathbf{P}_{1,i} + \mathbf{P}_{2,i}) &= \mathbf{P}_{1,i-1} - 2\mathbf{P}_{2,i-1} + \mathbf{K}_i \\
c_i^2\mathbf{K}_i - 2c_i^2\mathbf{P}_{1,i} + c_i^2\mathbf{P}_{2,i} &= \mathbf{P}_{1,i-1} - 2\mathbf{P}_{2,i-1} + \mathbf{K}_i \\
-\mathbf{P}_{1,i-1} + 2\mathbf{P}_{2,i-1} + (c_i^2 - 1)\mathbf{K}_i - 2c_i^2\mathbf{P}_{1,i} + c_i^2\mathbf{P}_{2,i} &= \mathbf{0} \qquad (11)
\end{aligned}$$

And now we basically follow through with the same algebra.

$$\begin{gathered}
-\mathbf{P}_{1,i-1} + 2\mathbf{P}_{2,i-1} + (c_i^2 - 1)\mathbf{K}_i - 2c_i^2\mathbf{P}_{1,i} + c_i^2\mathbf{P}_{2,i} = \mathbf{0} \\
-\mathbf{P}_{1,i-1} + 2(\mathbf{K}_i - c_i(\mathbf{P}_{1,i} - \mathbf{K}_i)) + (c_i^2 - 1)\mathbf{K}_i - 2c_i^2\mathbf{P}_{1,i} + c_i^2(\mathbf{K}_{i+1} - c_{i+1}(\mathbf{P}_{1,i+1} - \mathbf{K}_{i+1})) = \mathbf{0} \\
-\mathbf{P}_{1,i-1} + 2\mathbf{K}_i - 2c_i\mathbf{P}_{1,i} + 2c_i\mathbf{K}_i + (c_i^2 - 1)\mathbf{K}_i - 2c_i^2\mathbf{P}_{1,i} + c_i^2\mathbf{K}_{i+1} - c_{i+1}c_i^2\mathbf{P}_{1,i+1} + c_{i+1}c_i^2\mathbf{K}_{i+1} = \mathbf{0} \\
\mathbf{P}_{1,i-1} + 2c_i(1 + c_i)\mathbf{P}_{1,i} + c_{i+1}c_i^2\mathbf{P}_{1,i+1} = (1 + 2c_i + c_i^2)\mathbf{K}_i + c_i^2(1 + c_{i+1})\mathbf{K}_{i+1} \qquad (12)
\end{gathered}$$

That looks messy, but it's our replacement for equation (6). We'll also need to replace equation (7).

$$\begin{aligned}
\mathbf{K}_1 - 2\mathbf{P}_{1,1} + \mathbf{P}_{2,1} &= \mathbf{0} \\
\mathbf{K}_1 - 2\mathbf{P}_{1,1} + (\mathbf{K}_2 - c_2(\mathbf{P}_{1,2} - \mathbf{K}_2)) &= \mathbf{0} \\
\mathbf{K}_1 - 2\mathbf{P}_{1,1} + \mathbf{K}_2 - c_2\mathbf{P}_{1,2} + c_2\mathbf{K}_2 &= \mathbf{0} \\
2\mathbf{P}_{1,1} + c_2\mathbf{P}_{1,2} &= \mathbf{K}_1 + (1 + c_2)\mathbf{K}_2 \qquad (13)
\end{aligned}$$

And equation (8).

$$\begin{aligned}
\mathbf{P}_{1,n-1} - 2\mathbf{P}_{2,n-1} + \mathbf{K}_n &= \mathbf{0}\\
\mathbf{P}_{1,n-1} - 2(\mathbf{K}_n - c_n(\mathbf{P}_{1,n} - \mathbf{K}_n)) + \mathbf{K}_n &= \mathbf{0}\\
\mathbf{P}_{1,n-1} - 2\mathbf{K}_n + 2c_n\mathbf{P}_{1,n} - 2c_n\mathbf{K}_n + \mathbf{K}_n &= \mathbf{0}\\
\mathbf{P}_{1,n} &= \frac{2c_n+1}{2c_n}\mathbf{K}_n - \frac{1}{2c_n}\mathbf{P}_{1,n-1}
\end{aligned}$$

$$\begin{aligned}
\mathbf{P}_{1,n-2} + 2c_{n-1}(1+c_{n-1})\mathbf{P}_{1,n-1} + c_n c_{n-1}^2\mathbf{P}_{1,n} &= (1+2c_{n-1}+c_{n-1}^2)\mathbf{K}_{n-1} + c_{n-1}^2(1+c_n)\mathbf{K}_n\\
\mathbf{P}_{1,n-2} + 2c_{n-1}(1+c_{n-1})\mathbf{P}_{1,n-1} + c_n c_{n-1}^2(\frac{2c_n+1}{2c_n}\mathbf{K}_n - \frac{1}{2c_n}\mathbf{P}_{1,n-1}) &= (1+2c_{n-1}+c_{n-1}^2)\mathbf{K}_{n-1} + c_{n-1}^2(1+c_n)\mathbf{K}_n\\
\mathbf{P}_{1,n-2} + 2c_{n-1}(1+c_{n-1})\mathbf{P}_{1,n-1} + c_{n-1}^2(c_n+\frac{1}{2})\mathbf{K}_n - \frac{1}{2}c_{n-1}^2\mathbf{P}_{1,n-1} &= (1+2c_{n-1}+c_{n-1}^2)\mathbf{K}_{n-1} + c_{n-1}^2(1+c_n)\mathbf{K}_n\\
\mathbf{P}_{1,n-2} + (2c_{n-1}+\frac{3}{2}c_{n-1}^2)\mathbf{P}_{1,n-1} &= (1+2c_{n-1}+c_{n-1}^2)\mathbf{K}_{n-1} + \frac{1}{2}c_{n-1}^2\mathbf{K}_n \qquad (14)
\end{aligned}$$

Those are some gross-looking equations, but they're not all that complex. We'll use equations (13), (12), and (14) for our new matrix.

$$\begin{aligned}
2\mathbf{P}_{1,1} + c_2\mathbf{P}_{1,2} &= \mathbf{K}_1 + (1+c_2)\mathbf{K}_2\\
\mathbf{P}_{1,i-1} + 2(c_i+c_i^2)\mathbf{P}_{1,i} + c_{i+1}c_i^2\mathbf{P}_{1,i+1} &= (1+2c_i+c_i^2)\mathbf{K}_i + c_i^2(1+c_{i+1})\mathbf{K}_{i+1}\\
\mathbf{P}_{1,n-2} + (2c_{n-1}+\frac{3}{2}c_{n-1}^2)\mathbf{P}_{1,n-1} &= (1+2c_{n-1}+c_{n-1}^2)\mathbf{K}_{n-1} + \frac{1}{2}c_{n-1}^2\mathbf{K}_n
\end{aligned}$$

An important note here is that these equations should decompose into (7), (6), and (8) respectively when all of our constants are equal to 1 because that was our original problem. It's not too hard to verify that that is the case.

Now we put everything into matrix form:

$$\begin{pmatrix}
2 & c_2 & & & & 0\\
1 & 2(c_2+c_2^2) & c_3c_2^2 & & & \\
 & 1 & 2(c_3+c_3^2) & c_4c_3^2 & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & 1 & 2(c_{n-2}+c_{n-2}^2) & c_{n-1}c_{n-2}^2\\
0 & & & & 1 & 2c_{n-1}+\frac{3}{2}c_{n-1}^2
\end{pmatrix}
\begin{pmatrix}
\mathbf{P}_{1,1}\\ \mathbf{P}_{1,2}\\ \vdots\\ \mathbf{P}_{1,n-1}
\end{pmatrix} =
\begin{pmatrix}
\mathbf{K}_1 + (1+c_2)\mathbf{K}_2\\
(1+2c_2+c_2^2)\mathbf{K}_2 + c_2^2(1+c_3)\mathbf{K}_3\\
(1+2c_3+c_3^2)\mathbf{K}_3 + c_3^2(1+c_4)\mathbf{K}_4\\
\vdots\\
(1+2c_{n-2}+c_{n-2}^2)\mathbf{K}_{n-2} + c_{n-2}^2(1+c_{n-1})\mathbf{K}_{n-1}\\
(1+2c_{n-1}+c_{n-1}^2)\mathbf{K}_{n-1} + \frac{1}{2}c_{n-1}^2\mathbf{K}_n
\end{pmatrix} \qquad (15)$$

A little messy-looking, but fairly easy to implement. We're still using the Thomas algorithm because we still have a tridiagonal matrix.

And of course, once we have all the $\mathbf{P}_1$'s, we plug them into (11) and (9) to get the $\mathbf{P}_2$'s. And finally we have a good-looking spline that is built provided only knots.

$$\frac{d}{dt}\|\mathbf{x}(t)\|$$

$$\frac{d}{dt}\sqrt{\sum_{i=0}^{n} x_i(t)^2}$$

$$\frac{d}{dt}(\sum_{i=0}^{n} x_i(t)^2)^{\frac{1}{2}}$$

$$\frac{1}{2}(\sum_{i=0}^{n} x_i(t)^2)^{-\frac{1}{2}} \cdot \frac{d}{dt}\sum_{i=0}^{n} x_i(t)^2$$

$$\frac{1}{2}\frac{1}{\|\mathbf{x}(t)\|} \cdot \frac{d}{dt}\sum_{i=0}^{n} x_i(t)^2$$

$$\frac{1}{2}\frac{1}{\|\mathbf{x}(t)\|} \cdot \sum_{i=0}^{n} \frac{d}{dt}[x_i(t)^2]$$

$$\frac{1}{2}\frac{1}{\|\mathbf{x}(t)\|} \cdot \sum_{i=0}^{n} 2x_i(t)x_i'(t)$$

$$\frac{1}{2}\frac{1}{\|\mathbf{x}(t)\|} \cdot 2\sum_{i=0}^{n} x_i(t)x_i'(t)$$

$$\frac{1}{\|\mathbf{x}(t)\|} \cdot \sum_{i=0}^{n} x_i(t)x_i'(t)$$

$$\frac{\mathbf{x}(t) \cdot \mathbf{x}'(t)}{\|\mathbf{x}(t)\|}$$

$$\frac{d}{dt}\|\mathbf{x}(t)\| = \frac{\mathbf{x}(t) \cdot \mathbf{x}'(t)}{\|\mathbf{x}(t)\|}$$