Piecewise Polynomials and Matlab

First, we see how to define piecewise polynomials and evaluate these polynomials in Matlab. Here are two polynomials:

\[ P_1(x) = \begin{cases} 
3x^2 - 5x + 1, & \text{if } 0 \leq x \leq 5 \\
3x - 2, & \text{if } 5 < x \leq 10 
\end{cases} \]

\[ P_2(x) = \begin{cases} 
x^2 - 5x, & \text{if } -2 \leq x \leq 0 \\
3x - 2, & \text{if } 0 < x \leq 2 \\
-x^2 + 1, & \text{if } 2 < x \leq 4 
\end{cases} \]

In Matlab, we need to build a structure that contains the “break” points, and the polynomial coefficients. To build \( P_1(x) \), use \texttt{mkpp}:

\[
\text{Coefs}=[3,-5,1;0,3,-2]; \\
\text{breaks}=[0 5 10]; \\
\text{P1}=\text{mkpp}(\text{breaks},\text{Coefs});
\]

To evaluate this polynomial, say at equally spaced points between -3 and 12 and plot the result, use:

\[
\text{xx}=\text{linspace}(-3,12); \\
\text{yy}=\text{ppval(xx,P1)}; \\
\text{plot(xx,yy)}
\]

Notice that if you evaluate past the intervals on which the polynomial is defined, Matlab will use the first or last function.

Now build \( P_2(x) \), and plot it on the interval from \(-3\) to \(5\).

\[
\text{Coefs}=[1,-5,0;0,3,-2;-1,0,1]; \\
\text{breaks}=[-2 0 2 4]; \\
\text{P1}=\text{mkpp}(\text{breaks},\text{Coefs}); \\
\text{xx}=\text{linspace}(-3,5); \\
\text{yy}=\text{ppval(xx,P1)}; \\
\text{plot(xx,yy)}
\]

Cubic Splines and PPVAL

In this example, we create a cubic spine approximating \( \sin(x) \) for \(-2 \leq x \leq 2\) on 12 equally spaced points:

\[
\text{x}=\text{linspace}(-2,2,12); \\
\text{y}=\text{sin(x)}; \\
\text{P}=\text{spline(x,y)};
\]
The coefficients are stored in `P.coefs` and the breaks are stored as `P.breaks`. The coefficients are stored just as we did previously - That is, if a row of `P.coefs` looks like: `[1 2 3 4]`, that corresponds to

\[ 1 \cdot x^3 + 2 \cdot x^2 + 3 \cdot x + 4 \]

Now, we would like to plot the derivative of our splines. Note that each spline has the form:

\[ ax^3 + bx^2 + cx + d \]

so the derivative is \( 3ax^2 + 2bx + c \), so the coefficients are \( 3a, 2b, c \). We can re-form the coefficient matrix:

\[
T = P.coefs; \quad %\text{Temporary storage} \\
Coefs = [3*T(:,1), 2*T(:,2), T(:,3)]; \\
P = mkpp(P.breaks, Coefs); \\
clear Coefs T
\]

To plot the result, we could say:

```matlab
T = P.coefs; %Temporary storage 
Coefs = [3*T(:,1), 2*T(:,2), T(:,3)]; 
P = mkpp(P.breaks, Coefs); 
clear Coefs T 

t = linspace(-2,2); 
newY = ppval(t, P); 
plot(t, newY);
```