Taylor Polynomials

Taylor's Theorem is used extensively in Numerical Analysis and is the basis for the development of several important techniques.

We begin by restarting and loading the plots package for some extra plotting commands.

```
> restart; with(plots);
[animate, animate3d, animatecurve, arrow, changecoords, complexplot, complexplot3d, conformal, conformal3d, contourplot, contourplot3d, coordplot, coordplot3d, densityplot, display, dualaxisplot, fieldplot, fieldplot3d, gradplot, gradplot3d, implicitplot, implicitplot3d, inequal, interactive, interactiveparams, intersectplot, listcontplot, listcontplot3d, listdensityplot, listplot, listplot3d, loglogplot, logplot, matrixplot, multiple, odeplot, pareto, plotcompare, pointplot, pointplot3d, polarplot, polygonplot, polygonplot3d, polyhedra_supported, polyhedraplot, rootlocus, semilogplot, setcolors, setoptions, setoptions3d, spacecurve, sparsematrixplot, surfdata, textplot, textplot3d, tubeplot]
```

We next load the package called NumericalAnalysis in the Student package. It can be accessed in two ways. The first way is

```
> with(Student);
[Calculus1, LinearAlgebra, MultivariateCalculus, NumericalAnalysis, Precalculus, SetColors, VectorCalculus]
```

```
> with(NumericalAnalysis);
[AbsoluteError, AdamsBashforth, AdamsBashforthMoulton, AdamsMoulton, AdaptiveQuadrature, AddPoint, ApproximateExactUpperBound, ApproximateValue, BackSubstitution, BasisFunctions, Bisection, CubicSpline, DataPoints, Distance, DividedDifferenceTable, Draw, Euler, EulerTutor, ExactValue, FalsePosition, FixedPointIteration, ForwardSubstitution, Function, InitialValueProblem, InitialValueProblemTutor, Interpolant, InterpolantRemainderTerm, IsConvergent, IsMatrixShape, IterativeApproximate, IterativeFormula, IterativeFormulaTutor, LeadingPrincipalSubmatrix, LinearSolve, LinearSystem, MatrixConvergence, MatrixDecomposition, MatrixDecompositionTutor, ModifiedNewton, NevilleTable, Newton, NumberOfSignificantDigits, PolynomialInterpolation, Quadrature, RateOfConvergence, RelativeError, RemainderTerm, Roots, RungeKutta, Secant, SpectralRadius, Steffensen, Taylor, TaylorPolynomial, UpperBoundOfRemainderTerm, VectorLimit]
```

Most of these new commands are appropriate to our class. Unless one is using several packages in Student simultaneously, a more compact way of opening our package is:

```
> with(Student[NumericalAnalysis]);
[AbsoluteError, AdamsBashforth, AdamsBashforthMoulton, AdamsMoulton, AdaptiveQuadrature, AddPoint, ApproximateExactUpperBound, ApproximateValue, BackSubstitution, BasisFunctions, Bisection, CubicSpline, DataPoints, Distance, DividedDifferenceTable, Draw, Euler, EulerTutor, ExactValue, FalsePosition, FixedPointIteration, ForwardSubstitution, Function,]
```
InitialValueProblem, InitialValueProblemTutor, Interpolant, InterpolantRemainderTerm, IsConvergent, IsMatrixShape, IterativeApproximate, IterativeFormula, IterativeFormulaTutor, LeadingPrincipalSubmatrix, LinearSolve, LinearSystem, MatrixConvergence, MatrixDecomposition, MatrixDecompositionTutor, ModifiedNewton, NevilleTable, Newton, NumberOfSignificantDigits, PolynomialInterpolation, Quadrature, RateOfConvergence, RelativeError, RemainderTerm, Roots, RungeKutta, Secant, SpectralRadius, Steffensen, Taylor, TaylorPolynomial, UpperBoundOfRemainderTerm, VectorLimit

To help understand Taylor's Theorem, we look at several Taylor polynomials $P_n(x)$ for the function $f(x) = \frac{1}{x}$ about $x_0 = 1$. We begin by entering the function as an expression.

```maple
> restart; with(plots): with(Student[NumericalAnalysis]):
> f := 1/x;

We use the command TaylorPolynomial to form the Taylor polynomials for the function. The first argument is the function expression, the second is the $x_0$, the third is the list of the orders we want.

```maple
> t := TaylorPolynomial(f, x = 1, order=[1, 2, 3, 4, 5, 6, 30]);
```

Note that, besides a list, order= can be followed by a single number or a range as 3..7. $t$ is now a Maple list of 7 polynomials. We use a for loop to expand the polynomials.

```maple
> for i from 1 to 6 do
    P[i] := expand(t[i])
end do;
P[30] := expand(t[7]);
```

$P_1 := 2 - x$

$P_2 := 3 - 3x + x^2$

$P_3 := 4 - 6x + 4x^2 - x^3$

$P_4 := 5 - 10x + 10x^2 - 5x^3 + x^4$

$P_5 := 6 - 15x + 20x^2 - 15x^3 + 6x^4 - x^5$

$P_6 := 7 - 21x - 35x^2 + 21x^4 - 7x^5 + x^6 + 35x^2$

$P_{30} := 31 - 465x - 31465x^3 + 169911x^4 - 736281x^5 + 2629575x^6 - 7888725x^7 + 20160075x^8$
\[ + 4495 x^2 - 141120525 x^{11} + 206253075 x^{12} - 265182525 x^{13} + 300540195 x^{14} \\
- 300540195 x^{15} + 265182525 x^{16} - 206253075 x^{17} + 141120525 x^{18} - 84672315 x^{19} \\
+ 44352165 x^{20} - 20160075 x^{21} - 44352165 x^9 + 84672315 x^{10} + 7888725 x^{22} - 2629575 x^{23} \\
+ 736281 x^{24} - 169911 x^{25} + 31465 x^{26} - 4495 x^{27} + 465 x^{28} - 31 x^{29} + x^{30} \]

Since \texttt{TaylorPolynomial} can only take positive integers for \texttt{order}, we define \( P_0 = 1 \).
\[
> P[0]:=1;
\]
\[
P_0 := 1
\]

Now let's plot \( P_0(x) \) (in blue) and \( f \) (in red) together.
\[
> \text{plot0:=plot}(f,x=0..4,y=-5..10,\text{color}=\text{red});
> \text{plot1:=plot}(P[0],x=0..4,y=-5..10,\text{color}=\text{blue});
> \text{display}(\text{plot0,plot1});
\]

Note that the function and the degree 0 Taylor polynomial have the same value at \( x = 1 \). However, as you vary from \( x = 1 \), the constant polynomial is hardly a good approximation to the function. Let's see what happens as we move up a degree to a first order Taylor polynomial.
\[
> \text{plot1:=plot}(P[1],x=0..4,y=-5..10,\text{color}=\text{blue});
> \text{display}(\text{plot0,plot1});
\]

This is the linear approximation to \( f \) at \( x = 1 \). Here both the function and the degree one Taylor
polynomial have the same value and the same derivative at $x = 1$, so if we stay close enough to $x = 1$, we can use the polynomial, which is always easy to evaluate, to approximate the function. Let's advance to a second order Taylor polynomial.
Now the graphs also have the same second derivative at $x = 1$ (both are concave up), allowing us to move further from $x = 1$ than before and still get good approximations. Using `diff` (for taking the derivative of an expression) and `eval`, let's check that the first and second derivatives for the Taylor polynomial and function are really both the same. For the first derivatives:

```
> eval(diff(f,x),x=1);
-1
> eval(diff(P[2],x),x=1);
-1
```

So the first derivatives are really the same. Note the use of the $\$$_ operator to get the second derivative. For the second derivatives:

```
> eval(diff(f,x$2),x=1);
2
> eval(diff(P[2],x$2),x=1);
2
```

They are also the same. It continues that the number of derivatives matching at the chosen value $x_0$ is the same as the degree of the Taylor polynomial, giving ever better approximations further and further from $x = 1$. For the third degree:

```
> plot1:=plot(P[3],x=0..4,y=-5..10,color=blue):
```
Here the Taylor polynomial looks to be an excellent approximation from .6 to 1.6. For the fourth degree:

```plaintext
> plot1:=plot(P[4], x=0..4, y=-5..10, color=blue):
> display(plot0, plot1);
```
This looks good from .5 to 1.7. For the fifth degree:

```plaintext
> plot1:=plot(P[5],x=0..4,y=-5..10,color=blue):
> display(plot0,plot1);
```
Now we get good approximations from .4 to 1.7. For degree six:

```
> plot1:=plot(P[6],x=0..4,y=-5..10,color=blue):
> display(plot0,plot1);
```
This looks good from .4 to 1.8. Now let's jump to degree 30:

```maple
> plot1:=plot(P[30],x=0..4,y=-5..10,color=blue):
> display(plot0,plot1);
```
This now looks good from .2 to 1.9. Let's check the amount of error in the Taylor polynomial approximation for $x = \frac{18}{10}$.

\[
\text{actual} := \text{evalf(subs(x=18/10,f))};
\]
\[
\text{actual} := 0.5555555556
\]

\[
\text{approx30} := \text{evalf(subs(x=18/10,P[30]))};
\]
\[
\text{approx30} := 0.5561057511
\]

\[
\text{err} := \text{abs(approx30-actual)};
\]
\[
\text{err} := 0.0005501955
\]

We are off by about $\frac{1}{2000}$. Pretty good.