```
> #Title: MVT
  #Author: Gord Clement, May 2011
  #Description: For a given function and interval, this procedure determines if the conditions

      #                of the mean value theorem hold. If so, all possible values c on the interval which
      satisfy the conclusion

      #                of the theorem are found. An animation shows the function plotted on the
      interval with the tangent line
    #                moving across the function, highlighting the c values when they are reached
  #Usage:
  #Call : MVT ( function, interval)
  #function : function to be used for the theorem, note : must be continuous on the given interval [ a,
      b ] and differentiable on ( a, b )

      #interval: interval to be used in the theorem, entered in standard Maple notation, ie [a,b]
      would be entered x=a..b

  MVT := proc(expr, range)
     #local variable declarations
     local slope, dir, clist, fullList, temp, i, j, found, output, var, a, b, step, tanslope, tempx, tany,
      tanline, aplot, bplot, secplot, funcplot, fAta, tanplot, fullplot, miny, maxy, tempc, multiplec,
      previousStop, nextStop, length, maxdir, ctanplot;

      #extract variable and end-points of interval
      var := op( 1, range);
      a := evalf(op(1, op(2, range ) ));
      b := evalf(op(2, op(2, range) ));
      miny := minimize(expr, var = a ..b);
      maxy := maximize(expr, var = a ..b);

       #calculate length of interval, used for scaling in final plots
       length := evalf( (b - a) / 5 );
      #calculate slope of the line joining (a,f(a)) to (b,f(b))
      slope := (subs(var = b, expr) - subs(var = a, expr) ) / (b - a);
       #calculate f(a)
       fAta := subs(var = a, expr);
      #obtain derivative
      dir := diff(expr, var);
      #obtain highest magnitude of derivative on interval
      maxdir := maximize(|dir|, var = a ..b);

       #dummy variable to indicate if c value is found
     found := false;
     output := -1;

       #find all potential c values
     fullList := evalf(solve(dir = slope, var, dropmultiplicity = true ));
     #dummy variable to indicate multiple c values were found
     multiplec := false;
```

```
    #confirm interval given is valid
    if (a ≥ b) then
       output := "Error: a must be less than b";
    #confirm function does not have vertical tangent lines
    elif (is(type(maxdir, infinity))) then
       output
     := "Error:  Function does not satisfy the conditions of the Mean Value Theorem since it is
     not differentiable on (a,b)";
    #confirm function is continous on given range
   elif (not(iscont(expr, var=a..b,'closed'))) then
        output
     := "Error:  Function does not satisfy the conditions of the Mean Value Theorem since it is
     not continuous on [a,b]";
    #confirm function is not linear, which would be a trivial case of the theorem
    elif (diff(expr, var, var) = 0) then
      output
     := "Error: Do not use a linear function, as all values between a and b satisfy the Mean Value
     Theorem";

   else
      #if only one potential c, confirm it is in the interval
      if (nops([fullList]) = 1) then
        if (is(fullList < b) and is(fullList > a)) then
            clist := fullList;
        else
            output
     := "Error:  Function does not satisfy the conditions of the Mean Value Theorem on the given
     range";
        end if;
       else
           #find all of the potential c values that are in the interval
          multiplec := true;
          clist := [ ];
          for i from 1 to nops([fullList]) do
            temp := fullList[i];
             if (is(temp < b) and is(temp > a)) then
                found := true;
                clist := [op(clist), temp];
            end if;
          end do;
        if (not(found)) then
             output
     := "Error:  Function does not satisfy the conditions of the Mean Value Theorem on the given
     range";
        end if;

   end if;
end if;
 #if there was an error, display error message
   if (not (output=-1)) then
     output;
```

```
else
   #animation code for single c value
  if (not(multiplec)) then
        #animate up to c value
        step := evalf((clist - a)/25);
        output := [ ];
        tempx := a;

         #plot a, b the function and the secant line
        aplot := plot([a, t, t = miny - 0.2 ..maxy + 0.2], linestyle = dash, color = black) :
        bplot := plot([b, t, t = miny - 0.2 ..maxy + 0.2], linestyle = dash, color = black) :
        funcplot := plot([var, expr, var = a - 0.2 ..b + 0.2], thickness = 2, color = black) :
        secplot := plot([t, slope·(t - a) + fAta, t = a ..b], thickness = 2, color = red) :

        #plot tangent lines
        for i from 1 to 25 do
            tempx := tempx + step;
            tanslope := evalf(subs(var = tempx, dir));
            tany := evalf(subs(var = tempx, expr));
            tanline := tanslope·(t - tempx) + tany;
            tanplot := plot([t, tanline, t = tempx - length ..tempx + length], color = black,
  thickness = 2) :
            fullplot := plots[display]([aplot, bplot, tanplot, funcplot, secplot], title
  = typeset(var, "=", evalf(tempx)), labels = ["", ""]) :
           output := [op(output), fullplot] :
        end do;
        #plot tangent line for c value
        tanslope := evalf(subs(var = clist, dir));
        tany := evalf(subs(var = clist, expr));
        tanline := tanslope·(t - clist) + tany;
        ctanplot := plot([t, tanline, t = clist - length ..clist + length], color = red, thickness
  = 2) :
       #freeze plot for 10 frames
        fullplot := plots[display]([aplot, bplot, ctanplot, funcplot, secplot], title
  = typeset("c=", clist), labels = ["", ""]) :
       for i from 1 to 10 do
         output := [op(output), fullplot] :
       end do;
        #plot tangent lines after c value
       step := evalf((b - clist)/25);
       tempx := clist;
      for i from 1 to 25 do
            tempx := tempx + step;
            tanslope := evalf(subs(var = tempx, dir));
            tany := evalf(subs(var = tempx, expr));
            tanline := tanslope·(t - tempx) + tany;
            tanplot := plot([t, tanline, t = tempx - length ..tempx + length], color = black,
  thickness = 2) :
            fullplot := plots[display]([aplot, bplot, tanplot, funcplot, secplot, ctanplot], title
```

```
= typeset(var, "=", evalf(tempx)), labels = ["", ""]) :
            output := [op(output), fullplot] :
        end do;
         #add final frame and animate
        fullplot := plots[display]([aplot, bplot, funcplot, secplot, ctanplot], title = typeset(var,
"=", evalf(tempx)), labels = ["", ""]) :
        output := [op(output), fullplot] :
       plots[display]([op(output)], insequence = true, view = [a − 0.2 ..b + 0.2, miny − 0.2
..maxy + 0.2]);
 else
      #multiple c value animation code

      #order c values from low to high
       clist := sort(clist);
      #plots for a,b, f and the secant
      aplot := plot([a, t, t = miny − 0.2 ..maxy + 0.2], linestyle = dash, color = black) :
      bplot := plot([b, t, t = miny − 0.2 ..maxy + 0.2], linestyle = dash, color = black) :
      funcplot := plot([var, expr, var = a − 0.2 ..b + 0.2], thickness = 2, color = black) :
      secplot := plot([t, slope·(t − a) + fAta, t = a ..b], thickness = 2, color = red) :
      output := [ ];
      tempx := a;
     #variables to determine which to values the tangents are plotting between
      previousStop := a;
      nextStop := evalf(clist[1]);

       ctanplot := [ ];

#loop plots tangents between stopping values (endpoints or c values) and pauses at c values
      for i from 1 to nops(clist) do
                tempc := evalf(clist[i]);
                step := evalf( (nextStop − previousStop)/25 );
                 #plot tangents
                 for j from 1 to 25 do
                    tempx := tempx + step;
                    tanslope := evalf(subs(var = tempx, dir));
                    tany := evalf(subs(var = tempx, expr));
                    tanline := tanslope·(t − tempx) + tany;
                    tanplot := plot([t, tanline, t = tempx − length ..tempx + length], color
= black, thickness = 2) :
                    fullplot := plots[display]([aplot, bplot, tanplot, funcplot, secplot,
op(ctanplot)], title = typeset(var, "=", evalf(tempx)), labels = ["", ""]) :
                   output := [op(output), fullplot] :
                end do;
                 #plot tangent for current c value
                tanslope := evalf(subs(var = tempc, dir));
                tany := evalf(subs(var = tempc, expr));
               tanline := tanslope·(t − tempc) + tany;
               ctanplot := [op(ctanplot), plot([t, tanline, t = tempc − length ..tempc
+ length], color = red, thickness = 2)] :
               fullplot := plots[display]([aplot, bplot, op(ctanplot), funcplot, secplot], title
= typeset("c=", tempc), labels = ["", ""]) :
```

```
                    #pause animation for 10 frames
                    for j from 1 to 10 do
                        output := [op(output), fullplot]:
                    end do;
                    #determine next stopping value
                    if (not(i = nops(clist))) then
                        previousStop := nextStop;
                        nextStop := evalf(clist[i + 1]);
                    else
                        previousStop := nextStop;
                        nextStop := b;
                    end if;
                end do;
                #plot from final c value to b
                step := evalf((nextStop - previousStop)/25);

                for j from 1 to 25 do
                    tempx := tempx + step;
                    tanslope := evalf(subs(var = tempx, dir));
                    tany := evalf(subs(var = tempx, expr));
                    tanline := tanslope·(t - tempx) + tany;
                    tanplot := plot([t, tanline, t = tempx - length..tempx + length], color = black,
        thickness = 2):
                    fullplot := plots[display]([aplot, bplot, tanplot, funcplot, secplot,
        op(ctanplot)], title = typeset(var, "=", evalf(tempx)), labels = ["", ""]):
                    output := [op(output), fullplot]:
                end do;

                #add final frame and animate
                fullplot := plots[display]([aplot, bplot, funcplot, secplot, op(ctanplot)], title
        = typeset(var, "=", evalf(tempx)), labels = ["", ""]):
                output := [op(output), fullplot]:
                plots[display]([op(output)], insequence = true, view = [a - 0.2..b + 0.2, miny - 0.2
        ..maxy + 0.2]);
            end if;
        end if;
        end proc:
```

```
> MVT(x^2 + 3 x + 1, x = -2..1)
```

```
> MVT(x^3 + 5, x = -1..1)
```

```
> MVT(tan(x), x = 0..3)
"Error:  Function does not satisfy the conditions of the Mean Value Theorem since it is not continuous on [a,b]"
```

(1)

```
> MVT(surd(x, 3), x = -1..1)
"Error:  Function does not satisfy the conditions of the Mean Value Theorem since it is not differentiable on (a,b)"
```

(2)

```
> MVT(abs(x), x = -2 ..3)
```
"Error: Function does not satisfy the conditions of the Mean Value Theorem on the given range" **(3)**

```
>
>
```