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

      #              of the mean Rolle's 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 : RollesTheorem( 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) and f(a) = f(b)

      #interval: interval to be used in the theorem, entered in standard Maple notation, ie [a,b]
      would be entered x=a..b
 #For fully commented code, see MVT as logic is the same
  RollesTheorem := proc(expr, range)
     local dir, clist, fullList, temp, i, j, found, output, var, a, b, step, tanslope, tempx, tany, tanline,
      aplot, bplot, secplot, funcplot, fAta, fAtb, tanplot, fullplot, miny, maxy, tempc, multiplec,
      previousStop, nextStop, length, maxdir, ctanplot;

   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);
   length := evalf( (b - a) / 5 );
   fAta := subs(var = a, expr);
  fAtb := subs(var = b, expr);

   dir := diff(expr, var);
  maxdir := maximize(|dir|, var = a ..b);
   found := false;
  output := -1;
    fullList := evalf(solve(dir = 0, var, dropmultiplicity = true ));
  multiplec := false;
  if (fAta ≠ fAtb) then
        output :=
     "Error: This function does not satisfy the conditions of Rolle's Theorem on the given range as f(a) does no
  elif (a ≥ b) then
        output := "Error: a must be less than b";
   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)";
  elif (not(iscont(expr, var = a ..b,'closed')) ) then
        output
       := "Error:  Function does not satisfy the conditions of the Rolle's Theorem on the given
      range";
```

```
 elif (dir = 0) then
      output
     := "Error: Do not use a constant function, as all values between a and b satisfy the Rolle's
    Theorem";
else
  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 Rolle's Theorem on the given
    range";
    end if;
  else
       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 Rolle's Theorem on the given
    range";
        end if;

  end if;
end if;
 if (not (output = -1)) then
    output;
else

    if (not(multiplec)) then
        step := evalf( (clist - a) / 25 );
          output := [ ];
          tempx := a;
          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) :
          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;

        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) :
        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;

        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;
           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
        clist := sort(clist);
      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;
      previousStop := a;
     nextStop := evalf(clist[1]);
     ctanplot := [ ];
      for i from 1 to nops(clist) do
            tempc := evalf(clist[i]);
            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;

            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, tanplot, funcplot, secplot,
op(ctanplot)], title = typeset("c=", tempc), labels = ["", ""]):
            for j from 1 to 10 do
              output := [op(output), fullplot]:
            end do;
            if (not(i = nops(clist))) then
                previousStop := nextStop;
                nextStop := evalf(clist[i + 1]);
            else
                previousStop := nextStop;
                nextStop := b;
            end if;
        end do;
        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;
        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:
```

```
> RollesTheorem(x^2 − 4 x + 1, x = 0..4)
```

> $RollesTheorem(x^3 - 3x^2 + 2, x = 0..2)$

"Error: This function does not satisfy the conditions of Rolle's Theorem on the given range as f (a) does not equal f(b)" **(1)**

> $RollesTheorem\left((x-2)\cdot(x-1)\cdot(x)\cdot\left(x+\frac{1}{2}\right)\cdot(x+3), x = -3..2\right)$

> $RollesTheorem(\text{surd}(x^2, 3), x = -1..1)$

"Error: Function does not satisfy the conditions of the Mean Value Theorem since it is not differentiable on (a,b)" **(2)**

>