```
> #Title: FormalLimit
  #Author: Gord Clement, July 2010
  #Description: For a given function, point and epsilon value, calculate the limit of the function at

     #                the point, the maximum delta value for the finite limit and displays an
     animation demonstrating
  #                all possible delta values
  #Usage:
  #Call: FormalLimit(function, variable, point, epsilon (optional), direction(optional))
  #function: function to use in animation
  #variable: the variable the limit is taken with respect to
  #point: point at which the limit is taken (numeric)
  #epsilon: given epsilon value used to calculate delta, default=0.1

     #direction: one of 'left', 'right' or 'center' indicating whether the limit is from the left, right or
     two-sided respectively

  FormalLimit := proc(expr, var, pt, epsilon := 0.1, dir := center)

  #define local variables
   local del, output, value, limval, yupper, ylower, xupper, xlower, xrange, i, leftend, rightend;

    #two-sided limit code
   if (dir = center) then
        #calculate limit
       limval := limit(expr, var = pt);
       #initialize delta
        del := 0;
        if (not (type(evalf(limval), numeric))) then
            #Error message
            output := "Error: Limit does not exist or is not finite at given point";
       elif (epsilon ≤ 0) then
            #Error message
            output := "Error: epsilon value must be positve";
       else
            #Calculate y bounds
            yupper := limval + epsilon;
            ylower := limval − epsilon;
           #Find all x ranges that satisfy the y bounds
           xrange := solve(abs(expr − limval) < epsilon);
           #if just one range
           if (nops([xrange]) = 1) then
                #extract end-points
                xlower := op(1, xrange);
                xlower := op(1, xlower);
                xupper := op(2, xrange);
                xupper := op(1, xupper);
            #many x ranges
           else
                for i from 1 to nops([xrange]) do
                     #extract end-points
                    leftend := op(1, xrange[i]);
```

```
                    leftend := op(1, leftend);
                    rightend := op(2, xrange[i]);
                   rightend := op(1, rightend);
                    #check if limit point is one of the end-points
                   if (simplify(pt − leftend) = 0) then
                         xupper := rightend;
                   elif (simplify(pt − rightend) = 0) then
                         xlower := leftend;
                   end if;
                  #check if limit point is in the range
                  if (is(pt in xrange[i])) then
                       #extract end-points
                       xlower := op(1, xrange[i]);
                       xlower := op(1, xlower);
                       xupper := op(2, xrange[i]);
                       xupper := op(1, xupper);
                   end if
              end do;
         end if;
         #calculate delta value
         del := min(abs(xupper − pt), abs(xlower − pt));
          #create animation
          plots[animate](plot, [[[var, expr, var = pt − 3 ·del..pt + 3·del], [t, yupper, t = pt − 3
    ·del..pt + 3·del], [t, ylower, t = pt − 3·del..pt + 3·del], [pt − delta, t, t = limval − epsilon
    ..limval + epsilon], [pt + delta, t, t = limval − epsilon..limval + epsilon], [pt, t, t = limval
    − 3·epsilon..limval + 3·epsilon], [t, subs(var = pt − delta, expr), t = pt − delta..pt
    + delta], [t, subs(var = pt + delta, expr), t = pt − delta..pt + delta]], horizontal = pt − 3 ·del
    ..pt + 3·del, vertical = limval − 3·epsilon..limval + 3·epsilon, labels = ["", ""], thickness
    = [3, 2, 2, 2, 2, 1, 1, 1], color = [black, red, red, blue, blue, black, red, red], linestyle
    = [solid, solid, solid, solid, solid, dash, dash, dash]], delta = 0..del);
     end if;
#one-sided limit from the left, logic very similar to two sided limit code
 elif (dir = left) then
      limval := limit(expr, var = pt, left);
      del := 0;
      if (not (type(evalf(limval), numeric))) then
           output := "Error: Limit does not exist or is not finite at given point";
     elif (epsilon ≤ 0) then
          output := "Error: epsilon value must be positve";
     else
          yupper := limval + epsilon;
          ylower := limval − epsilon;
          xrange := solve(abs(expr − limval) < epsilon);
          if (nops([xrange]) = 1) then
               xlower := op(1, xrange);
               xlower := op(1, xlower);
          else
               for i from 1 to nops([xrange]) do
                    leftend := op(1, xrange[i]);
                    leftend := op(1, leftend);
                    rightend := op(2, xrange[i]);
                   rightend := op(1, rightend);
```

```
                    if (pt = rightend) then
                        xlower := leftend;
                    end if;
                    if (is(pt in xrange[i])) then
                        xlower := op(1, xrange[i]);
                        xlower := op(1, xlower);
                    end if;
                end do;
            end if;
            del := pt - xlower;
            plots[animate](plot, [[[var, expr, var = pt - 5 ·del..pt + del], [t, yupper, t = pt - 5·del
        ..pt + del], [t, ylower, t = pt - 5·del..pt + del], [pt - delta, t, t = limval - epsilon..limval
        + epsilon], [pt, t, t = limval - 3·epsilon..limval + 3·epsilon], [t, subs(var = pt - delta,
        expr), t = pt - delta..pt], [t, limval, t = pt - delta..pt]], horizontal = pt - 5 ·del..pt + del,
        vertical = limval - 3·epsilon..limval + 3·epsilon, labels = ["", ""], thickness = [3, 2, 2, 2, 1,
        1, 1], color = [black, red, red, blue, black, red, red], linestyle = [solid, solid, solid, solid, dash,
        dash, dash]], delta = 0..del);
        end if;
    #one-sided limit form the right code, logic very similar to two-sided limit code
    elif (dir = right) then
        limval := limit(expr, var = pt, right);
        del := 0;
        if (not (type(evalf(limval), numeric))) then
            output := "Error: Limit does not exist or is not finite at given point";
        elif (epsilon ≤ 0) then
            output := "Error: epsilon value must be positve";
        else
            yupper := limval + epsilon;
            ylower := limval - epsilon;
            xrange := solve(abs(expr - limval) < epsilon);
            if (nops([xrange]) = 1) then
                xupper := op(2, xrange);
                xupper := op(1, xupper);
            else
                for i from 1 to nops([xrange]) do
                    leftend := op(1, xrange[i]);
                    leftend := op(1, leftend);
                    rightend := op(2, xrange[i]);
                    rightend := op(1, rightend);
                    if (pt = leftend) then
                        xupper := rightend;
                    end if;
                    if (is(pt in xrange[i])) then
                        xupper := op(2, xrange[i]);
                        xupper := op(1, xupper);
                    end if;
                end do;
            end if;
            del := xupper - pt;
            plots[animate](plot, [[[var, expr, var = pt - del..pt + 5·del], [t, yupper, t = pt - del
        ..pt + 5·del], [t, ylower, t = pt - del..pt + 5· del], [pt + delta, t, t = limval - epsilon..limval
        + epsilon], [pt, t, t = limval - 3·epsilon..limval + 3·epsilon], [t, subs(var = pt + delta,
```

```
    expr), t = pt..pt + delta], [t, limval, t = pt..pt + delta]], horizontal = pt - del..pt + 5·del,
     vertical = limval - 3·epsilon..limval + 3·epsilon,  labels = ["", ""], thickness = [3, 2, 2, 2, 1,
    1, 1], color = [black, red, red, blue, black, red, red], linestyle = [solid, solid, solid, solid, dash,
    dash, dash]], delta = 0..del);
      end if;
  else
        output := ("Error: Direction must be one of 'left', 'center' or 'right'");
  end if;
  end proc:
>
```

$$> \mathit{FormalLimit}\left(\frac{1}{x-3}, x, 5, 0.4\right)$$

$$> \mathit{FormalLimit}\left(\sin^2(3\cdot x), x, 2, 0.3\right)$$

$$> \mathit{FormalLimit}\left(x^3 - \frac{1}{3}x^2 + \frac{1}{4}x + 2, x, \frac{-1}{4}, 0.3\right)$$

```
>
```