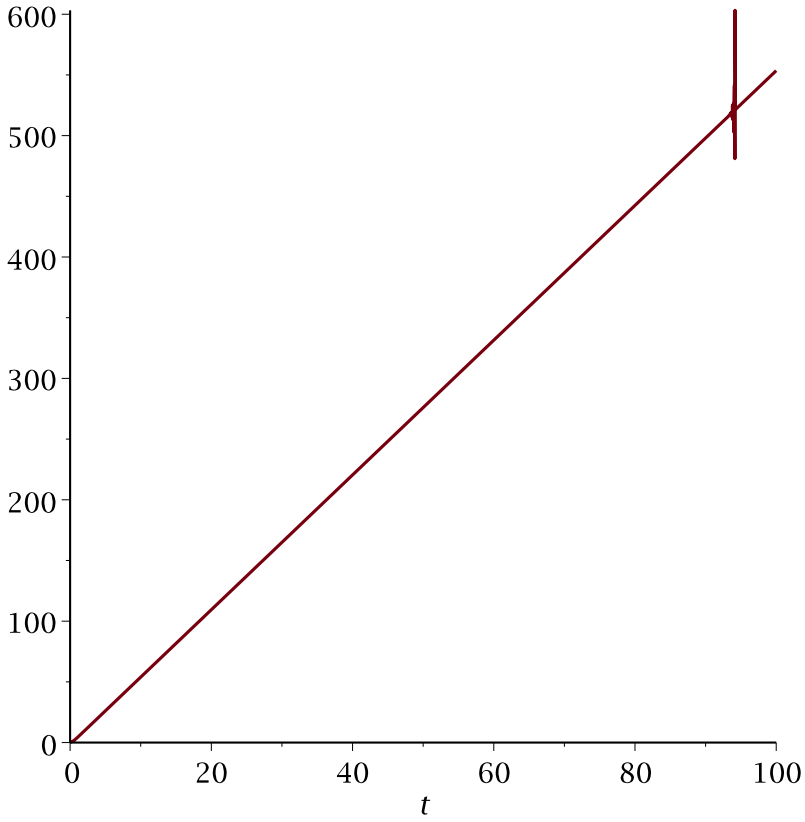
Bewegingsvergelijking

Maak van de bewegingsvergelijking een functie in t. (Experimenteel bepaald op basis van een eerste ontwerp).

```
> qr:=subs(overbr=18,x(t)):
> q:=t->-(151665298680333059 73/21550000000000000)*t-
  (1/2586000000000000 0)*t*sqrt
  (336480053650691284069309025878805966489 76)-(266000/1293)*ln
  (134592021460276513627723610351522386595904/
  (181998358416399671676*exp((1/532000000000000000 0)*t*sqrt
  (336480053650691284069309025878805966489 76))+sqrt
  (336480053650691284069309025878805966489 76)*exp(
  (1/532000000000000000 0)*t*sqrt
  (336480053650691284069309025878805966489 76))
  -181998358416399671676+sqrt
  (336480053650691284069309025878805966489 76))^2):
> speed := evalf(diff(q(t), t));
```

$$speed := -1413.117884 + \frac{5.184275238\,10^{23}\,e^{3.448006135\,t}}{3.654322848\,10^{20}\,e^{3.448006135\,t} + 1.4355680\,10^{18}} \qquad \textbf{(1.1)}$$

```
> plot(q(t), t = 0..100); #Quasi-lineaire bewegingsvergelijking
```



Animatiefilm

```
> restart:
> with(LinearAlgebra): with(VectorCalculus): with(plots): with(ArrayTools):
```

Animatie in 3D van het karretje dat een parcours volgt bestaande uit enkel rechte hoeken.

```
> translate_polygon := proc(polygon, translation)
    description "translate polygon";
    local result, i;
    result := LinearAlgebra[Copy](polygon):
    for i from 1 to Size(polygon, 1) do result[i] := polygon[i] + translation: end
       do;
    result;
    end proc:
> #De dimensies van het grondvlak van het karretje.
    front_width := 4:
    board_width := 7:
    board_length := 11:
```

```
board_height := 1:
front_length := 2:
board_matrix := Matrix([[-front_width/2, board_length/1, 0],
[front_width/2, board_length/1, 0],
[front_width/2, board_length/1 - front_length, 0],
[board_width/2, board_length/1 - front_length, 0],
[board_width/2, 0, 0],
[-board_width/2, 0, 0],
[-board_width/2, board_length/1 - front_length, 0],
[-front_width/2, board_length/1 - front_length, 0]]):
car_polygons := []:
#top_board: bovenkant
#bottom_board: onderkant
top_board := translate_polygon(board_matrix, Vector([0, 0, board_height/2],
    orientation = row)):
bottom_board := translate_polygon(board_matrix, Vector([0, 0,
    -board_height/2], orientation = row)):
car_polygons := [op(car_polygons), top_board]:
car_polygons := [op(car_polygons), bottom_board]:
#zijvlakken van het karretje (verbinding tussen boven- en ondervlak)
for i from 0 to Size(board_matrix, 1) - 1 do car_polygons
    := [op(car_polygons), Matrix([[top_board[i+1]], [top_board[(i+1)
    mod Size(board_matrix, 1) + 1]], [bottom_board[(i+1)
    mod Size(board_matrix, 1) + 1]], [bottom_board[i+1]]])]: end do:
> plot_wagentje :=proc(translation, rotation)
description "plot het wagentje in 3d, getransleerd + ter plaatse geroteerd";
local tmp_matrix, i, rotation_transposed, j, current_polygon;
rotation_transposed := rotation^+ :
tmp_matrix := []:
for j from 1 to Size(car_polygons, 2) do
  current_polygon := LinearAlgebra[Copy](car_polygons[j]);
  for i from 1 to Size(car_polygons[j], 1) do
  current_polygon[i] := VectorMatrixMultiply(car_polygons[j][i],
    rotation_transposed) + translation:
```

```
    end do;
    tmp_matrix := [op(tmp_matrix), current_polygon];
  end do;
  polygonplot3d(tmp_matrix, scaling = constrained, axes = none);
end proc:
> plot_wagentje_angle := proc(translation, angle)

    description "plot het wagentje in 3d, gegeven een translatievector en de
    hoek rond de z-as waarrond het wagentje geroteerd wordt";
  local rotation_matrix, theta;
  theta := angle- Pi/2 :
  rotation_matrix := Matrix([[-cos(theta), -sin(theta), 0], [-sin(theta),
    cos(theta), 0], [0, 0, 1]]) :
  plot_wagentje(translation, rotation_matrix);
  end proc:
>
> unit := 15 :#relatieve dimensie van het het parcours
  plot_wall_part := proc(p1, p2)
  description "plot een stuk van een muur van het parcours"
  local translation, result :
  translation := Vector([0, 0, 2], orientation = row);
  result := Matrix(4, 3) :
  result[1] := p1 :
  result[2] := p2:
  result[3] := p2 + translation:
  result[4] := p1 + translation:
  polygonplot3d(result);
  end proc:
  parcours := [ ]:
  #rand van het parcours
  border := Matrix([[0, 0, 0], [6·unit, 0, 0], [6·unit, 4·unit, 0], [0, 4·unit, 0], [0,
    0, 0]]) :
  #binnenmuren
  inner1 := Matrix([[0, 3·unit, 0], [5·unit, 3·unit, 0]]) :
  inner2 := Matrix([[unit, 3·unit, 0], [unit, unit, 0], [2·unit, unit, 0]]) :
  inner3 := Matrix([[2· unit, 2· unit, 0], [3·unit, 2· unit, 0], [3·unit, 0, 0]]) :
  inner4 := Matrix([[4·unit, 3·unit, 0], [4·unit, unit, 0], [5·unit, unit, 0]]) :
  inner5 := Matrix([[5·unit, 2·unit, 0], [6·unit, 2·unit, 0]]) :
  parcours := [border, inner1, inner2, inner3, inner4, inner5]:

  plot_wall := proc( )
  description "plot alle muren van het parcours"
  local i, all_parts, j;
  all_parts := [ ]:
  for j from 1 to Size(parcours, 2) do
  for i from 1 to Size(parcours[j], 1) - 1 do
  all_parts := [op(all_parts), plot_wall_part(parcours[j][ i ], parcours[j][ i
```

```
        +1])]:
    end do:
    end do:
    return all_parts:
    end proc:
    wall_plot := plot_wall( ):
> #de checkpoints van het pad dat het wagentje moet afleggen
    path := [[unit/2, 7/2 unit, 0], [11/2 unit, 7/2 unit, 0], [11/2 unit, 5/2 unit, 0], [9/2 unit,
        5/2 unit, 0], [9/2 unit, 3/2 unit, 0], [11/2 unit, 3/2 unit, 0], [11/2 unit, 1/2 unit, 0],
        [7/2 unit, 1/2 unit, 0], [7/2 unit, 5/2 unit, 0], [3/2 unit, 5/2 unit, 0], [3/2 unit, 3/2 unit, 0],
        [5/2 unit, 3/2 unit, 0], [5/2 unit, 1/2 unit, 0], [1/2 unit, 1/2 unit, 0], [1/2 unit, 5/2 unit
        - board_length, 0], [1/2 unit, 5/2 unit, 0]]:
> path_plot := pointplot3d(path, connect = true, color = black):
> speed := t→1:#Quasi-lineaire bewegingsvergelijking (zie bovenaan de file)
> current_step := 1:
    current_position := Vector(path[1], orientation = row):
    displays := [ ]:
    distance_left := 0:
    current_speed := 0:
    current_rotation := 0:
    #het wagentje beweegt zich aan constante snelheid

        #het midden tussen de achterwielen en het midden tussen de voorwielen
        bevinden zich altijd op de lijnen van het pad

        #in de bochten roteert het wagentje zich zo dat aan bovenstaande
        voorwaarde voldaan blijft
    while current_step < nops(path) do
     if distance_left = 0 then
       displays := [op(displays), display([path_plot, op(wall_plot),
         plot_wagentje_angle(current_position, current_rotation)], scaling
         = constrained)]:
       distance_left := speed(current_speed):
       current_speed := distance_left:
     else
       distance_to_go := Student[Precalculus][Distance](Vector(path[current_step
         +1], orientation = row), current_position):
       if distance_to_go < distance_left then
         #in the next part
         current_step := current_step+1:
         current_rotation := arctan(path[current_step+1][2]
         - path[current_step][2], path[current_step+1][1]
```

```
        - path[current_step][1]) :
        distance_left := distance_left - distance_to_go :
        current_position := Vector(path[current_step], orientation = row) :
      else
        current_position := current_position + (distance_left
       · (Vector(path[current_step + 1] - path[current_step], orientation = row)))
        / (Norm(Vector(path[current_step + 1] - path[current_step], orientation
        = row))) :

        if distance_to_go > distance_left + board_length then
          #still straight in this part

        else
          #at the corner
          back_corner_distance := distance_to_go - distance_left:
          next_angle := CrossProduct(Vector(path[current_step + 1]
        - path[current_step]), Vector(path[current_step + 2]
        - path[current_step]))[3]:
          d_angle := arccos( back_corner_distance / board_length ) · signum(next_angle) :
          current_rotation := arctan(path[current_step + 1][2]
        - path[current_step][2], path[current_step + 1][1]
        - path[current_step][1]) + d_angle;
        end if;
        distance_left := 0 :

      end if:
    end if:
  end do:
```

```
Error, invalid subscript selector
```

```
> display(displays, insequence = true);
```