Gabriele Minotto
Exercise: OPTIMIZATION of monocycle model in chained form, using polinomial functions

```
> restart: with(plots): with(LinearAlgebra): with(Threads): with
  (Optimization): with(VectorCalculus):
> # Data Inputs & Obstacle positions.
  data := [ T = 1, n_OP = 6, epsilon = 0.0001, n_P = 10]:
  Obs_Pos_Data := [ xp__1 = 0.2, yp__1 = 0.5, xp__2 = 0.2, yp__2 =
  0.3 , xp__3 = 0.5, yp__3 = 0.4, xp__4 = 0.6, yp__4 = 0.2, xp__5 =
  0.8, yp__5 = 0.5, xp__6 = 1.0, yp__6 = 0.5 ]: <%>:
  T__step := subs(data,[T/4, T/4, 2*T/4, 2*T/4, 3*T/4, 3*T/4]):
```

Model definition

```
> eq_v := [v__1(t) = x__3(t)*u__1(t) + u__2(t), v__2(t) = -u__1(t)]
  : <%>;
```

$$\begin{bmatrix} v_1(t) = x_3(t)\,u_1(t) + u_2(t) \\ v_2(t) = -u_1(t) \end{bmatrix} \quad (1)$$

```
> in_cond := [0, 0, delta__in];
  fi_cond := [1, 1, delta__fin];
```

$$in\_cond := \left[0, 0, \delta_{in}\right]$$

$$fi\_cond := \left[1, 1, \delta_{fin}\right] \quad (2)$$

1) Initial transformed conditions

```
> in_tr_cond := convert(subs(x(t)=in_cond[1],y(t)=in_cond[2],delta
  (t)=in_cond[3],eq_tr),list);
  fi_tr_cond := convert(subs(x(t)=fi_cond[1],y(t)=fi_cond[2],delta
  (t)=fi_cond[3],eq_tr),list);
```

$$in\_tr\_cond := [eq\_tr]$$

$$fi\_tr\_cond := [eq\_tr] \quad (3)$$

2)Input u__1 e u__2 in symbolic form (convert operation is useful to extract information from function equations)

```
> u__1(t) :=
  u__2(t) :=

Error, `:=` unexpected
> diff_ui(t) := [diff(x__1(t),t) = u__1(t), diff(x__2(t),t) = u__2
  (t)]: <%>:
  x__i(t) := [x__1(t) = int(rhs(diff_ui(t)[1]),t) + rhs(in_tr_cond
  [1]), x__2(t) = int(rhs(diff_ui(t)[2]),t) + rhs(in_tr_cond[2])]:
  <%>:
Error, invalid input: rhs received eq_tr, which is not valid for
its 1st argument, expr
> diff_3 := [diff(x__3(t),t) = x__2(t)*u__1(t)]: <%>:
  diff_u3(t) := subs(x__i(t),diff_3):
  x__i3 := x__3(t) = int(rhs(op(diff_u3(t))),t) + rhs(in_tr_cond[3]
  ):
Error, invalid input: subs received x__i(t), which is not valid
for its 1st argument
```

```
Error, invalid input: rhs received t, which is not valid for its
1st argument, expr
```

```
> # Generalized velocities and positions.
  vel_gen := [op(diff_ui(t)), op(diff_u3(t))]: <%>;
  pos_gen := [op(x__i(t)), simplify(x__i3)]: <%>;
```

$$\left[\begin{array}{c} \frac{d}{dt} x_1(t) = u_1(t) \\ \frac{d}{dt} x_2(t) = u_2(t) \\ t \end{array}\right]$$

$$\left[\begin{array}{c} t \\ x_{i3} \end{array}\right] \quad (4)$$

3) Constraint initial and final conditions - linear system.

```
> Bounds_1 := subs(t=subs(data,T),[rhs(pos_gen[1]) = rhs(fi_tr_cond
  [1]), rhs(pos_gen[2]) = rhs(fi_tr_cond[2]), rhs(pos_gen[3]) = rhs
  (fi_tr_cond[3])]): <%>;
  Bounds_2  #If you want, you can impose that tracking velocity
  must be grater than zero........
```

```
Error, invalid input: rhs received t, which is not valid for its
1st argument, expr
```

$$\left[\begin{array}{c} t \\ x_{i3} \end{array}\right]$$

```
Warning, premature end of input, use <Shift> + <Enter> to avoid
this message.
```

4) Preparation of Optimization: Target is the distance between path and the obstacles' positions. Bound is the linear system with final conditions.

```
> tr_to_gen :=
  tr_sub := subs(pos_gen,tr_to_gen): <%>:
```

```
Error, `:=` unexpected
```

Definition of target to minimize

```
> OP[1] := [xp__1, yp__1]:
  OP[2] := [xp__2, yp__2]:
  OP[3] := [xp__3, yp__3]:
  OP[4] := [xp__4, yp__4]:
  OP[5] := [xp__5, yp__5]:
  OP[6] := [xp__6, yp__6]:

  TarTot:= []:
  Tar1  := []:
  for i from 1 by 1 to subs(data,n_OP) do
```

```
    TarTot := [op(TarTot), OP[i]]:
     Tar1 := [ op(Tar1),  (  (x(t) - TarTot[i][1])^2 + (y(t) -
  TarTot[i][2])^2 )  ]:
  end do:

  Tar1;
  numelems(Tar1):
  Tar_sub := subs(tr_sub,Obs_Pos_Data,Tar1):
  numelems(Tar_sub):
  Targe := []:
  for i from 1 by 1 to subs(data,n_OP) do
   Targe := [op(Targe), subs(t = T__step[i],Tar_sub[i])]:
  end do:
  Targe: numelems(%):
  Tar := sum(Targe[k],k=1..numelems(Targe)):

  Bounds_2_sub := []:
  for k from 1 to 50 do
  Bounds_2_sub := [op(Bounds_2_sub), subs(t = subs(data,T/50*k),
  Bounds_2)]:
  end do:
  Bounds_2_sub:
```

$$\left[\left(x(t)-xp_1\right)^2+\left(y(t)-yp_1\right)^2,\left(x(t)-xp_2\right)^2+\left(y(t)-yp_2\right)^2,\left(x(t)-xp_3\right)^2+\left(y(t)-yp_3\right)^2,\left(x(t)-xp_4\right)^2+\left(y(t)-yp_4\right)^2,\left(x(t)-xp_5\right)^2+\left(y(t)-yp_5\right)^2,\left(x(t)-xp_6\right)^2+\left(y(t)-yp_6\right)^2\right]$$

```
Error, invalid input: subs received tr_sub, which is not valid
for its 1st argument
Error, invalid input: numelems expects its 1st argument, t, to
be of type indexable, but received Tar_sub
```

Optimization

```
> Optim := :      # Optimization WITH obstacles
  Optim_set := Optim[2];
Error, `:` unexpected
```

5) Obtain x__i(t) from optimization: original coordinates

```
> x__i_final := simplify(evalf(subs(Optim_set,pos_gen))): <%>:
Error, invalid input: subs received Optim_set, which is not
valid for its 1st argument
```

6) Obtain generalized coordinate x, y, delta

```
> gen_coord := evalf(subs(Optim_set,tr_sub)): <%>:
  subs(t=1,%%): <%>:
Error, invalid input: subs received Optim_set, which is not
valid for its 1st argument
```

7) Find the two inputs

```
> the_controls := simplify(subs(Optim_set,x__i_final,eq_v)): <%>;

Error, invalid input: subs received Optim_set, which is not
valid for its 1st argument
```

(5)

50 x 1 Matrix
Data Type: anything
Storage: rectangular
Order: Fortran_order (5)

8) Plot trajectories & controls

```
> P1 := plot([subs(gen_coord[1],x(t)), subs(gen_coord[2],y(t)), t =
  0 .. subs(data,T)], color="DarkOrange",gridlines=true,labels=["x
  (t) [m]","y(t) [m]"],title="Trajectory in time, WITH obstacles",
  axes=boxed):
  P2 := plot([subs(the_controls[1],v__1(t)), subs(the_controls[2],
  v__2(t))], t = 0 .. subs(data,T),color=["DarkOrange",
  "DarkMagenta"],gridlines=true,labels=["t [s]","v(t)"],title=
  "Controls in time WITH obstacles",axes=boxed,legend=["v__1(t)
  [m/s]","v__2(t) [rad/s]"]):
  P3 := pointplot({seq([op(subs(Obs_Pos_Data,OP[i]))], i = 1 ..
  subs(data,n_OP))}, color = blue, symbolsize = 20,gridlines=true,
  labels=["x [m]","y [m]"], view=[0..1, 0..1],color = ["Blue",
  "Blue","Green","Green","DarkMagenta","DarkMagenta"],title=
  "Position of coupled obstacles"):
  P13 := display([P1,P3]):
  PY := plot(subs(gen_coord,x(t)), t = 0.. subs(data,T), color =
  "DarkMagenta", legend=["x(t) [m]"]):
  PX := plot(subs(gen_coord,y(t)), t = 0.. subs(data,T), color =
  "DarkOrange", legend=["y(t) [m]"]):
  PXY := display([PX,PY],gridlines=true, title = "X(t) & Y(t) in
  time WITH obstacles", labels = ["t [s]", "xy_pos(t) [m]"],
  thickness = 1):
  PTRAG := plot(subs(gen_coord,sqrt((x(t))^2 + (y(t))^2)), t = 0..
  subs(data,T),color="DarkOrange",gridlines=true,labels=["t [s]",
  "Traj(t) [m]"],title="Trajectory in time WITH obstacles",axes=
  boxed):

  display(Array(1..2,[P13,P2, PXY, PTRAG]));
Error, invalid input: subs received gen_coord[1], which is not
valid for its 1st argument
Error, invalid input: subs received the_controls[1], which is
not valid for its 1st argument
Error, (in plots:-display) expecting plot structures but
received: [P1]
Error, invalid input: subs received gen_coord, which is not
valid for its 1st argument
Error, invalid input: subs received gen_coord, which is not
valid for its 1st argument
Error, (in plots:-display) expecting plot structures but
received: [PX, PY]
Error, invalid input: subs received gen_coord, which is not
valid for its 1st argument
Error, (in plots:-display) element 1 of the rtable is not a
valid plot structure
> ################################################
```

SUMMARY NOTES OF PROCEDURE:

1) Curvature in polynomial form with parametric coefficients to be optimized do the trajectory more elastic and efficient: results are strongly improved, especially for many more initial and

final conditions and many more sampling points.
2) Unfortunately if bounds are too much procedure become very long or cannot reach a possible solution.
3) However it is possible that there is a limit in efficiency in the basic model. Probably methods that use clothoids are better than ours.