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 posistions
  data := [ T = 1, n_OP = 3, epsilon = 0.05, n_P = 50 ];
  Obs_Pos_Data := [ xp__1 = 0.2, yp__1 = 0.2, xp__2 = 0.3, yp__2 =
  0.7, xp__3 = 0.8, yp__3 = 0.8 ]: <%>:
```

$$data := [T = 1, n\_OP = 3, \varepsilon = 0.05, n\_P = 50] \quad (1)$$

Model definition

```
> eq_tr := ;
Error, `;` unexpected
> eq_v := []: <%>;
```

$$[\ ] \quad (2)$$

```
> in_cond := [0, 0, Pi/4];   #curvatura iniziale parametrica
  fi_cond := [1, 1, Pi/4];
```

$$in\_cond := \left[0, 0, \frac{1}{4}\pi\right]$$

$$fi\_cond := \left[1, 1, \frac{1}{4}\pi\right] \quad (3)$$

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);
```

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

$$fi\_tr\_cond := [eq\_tr] \quad (4)$$

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

```
> u__1(t) := ; #symbolic formula
  u__2(t) :=  ;

Error, `;` unexpected
> diff_ui(t) := convert(<diff(x__1(t),t) = u__1(t), diff(x__2(t),t)
  = u__2(t)>,list):
  x__i(t) := convert(<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])>,list):
Error, invalid input: rhs received eq_tr, which is not valid for
its 1st argument, expr
> convert(<diff(x__3(t),t) = x__2(t)*u__1(t)>,list):
  diff_u3(t) := subs(x__i(t),%):
  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)]: <%>;
```

$$\begin{bmatrix} \frac{d}{dt} x_1(t) = u_1(t) \\ \frac{d}{dt} x_2(t) = u_2(t) \\ t \end{bmatrix}$$

$$\begin{bmatrix} t \\ x_{i3} \end{bmatrix} \quad (5)$$

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])]): <%>;
```

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

$$\begin{bmatrix} t \\ x_{i3} \end{bmatrix} \quad (6)$$

4) Preparation of optimization: Target is minimization of trajectory and then the path, bounds as final conditions and minimum distance from obstacles

```
> tr_to_gen := op(solve(convert(eq_tr,list),[x(t),y(t),delta(t)])):
  <%>;
  tr_sub := subs(pos_gen,tr_to_gen): <%>;
```

$$\begin{bmatrix} \quad \end{bmatrix}$$

$$\begin{bmatrix} t \\ x_{i3} \end{bmatrix} \quad (7)$$

Definition of target to minimize

```
> DTraj := ( );
  Sym_Target := int(DTraj,t=0..T);
  DTraj_sub := subs(tr_sub,DTraj):
  #Target := sum(subs(t = subs(data,T/n_P*k), DTraj_sub * (subs
  (data,T/n_P))), k = 0..subs(data,n_P-1)):
  Target := sum(subs(t = subs(data,T/n_P*k), DTraj_sub), k = 0..
  subs(data,n_P-1)):
```

$$DTraj :=$$

Error, invalid input: VectorCalculus:-int expects its 1st argument, f, to be of type {algebraic, Vector(algebraic)}, but received t = 0 .. T

Definition of obstacle bounds

```
> 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]:

  Phi := []:
  OPTOT := []:
  for i from 1 by 1 to subs(data,n_OP) do
    OPTOT := [op(OPTOT), OP[i]]:
    Phi := [ op(Phi),  (  (x(t) - OPTOT[i][1])^2 + (y(t) - OPTOT[i]
  [2])^2 ) - subs(data,epsilon)^2  >=  0 ]:
  end do:

  Phi;
  Phi_sub := subs(tr_sub,Obs_Pos_Data,Phi):
  Bounds_Phi := []:
  for i from 0 by 1 to subs(data,n_P) do
    for j from 1 by 1 to subs(data,n_OP) do
      Bounds_Phi := [ op(Bounds_Phi), subs(t = subs(data,T/n_P*i),
  Phi_sub[j])]:
    end do:
  end do:
  Bounds_Phi_1 := [ sum(Bounds_Phi[k], k = 1..subs(data,n_P)) ]:
  op(Bounds_Phi):
```

$$\left[0 \le \left(x(t) - xp_1\right)^2 + \left(y(t) - yp_1\right)^2 - 0.0025,\ 0 \le \left(x(t) - xp_2\right)^2 + \left(y(t) - yp_2\right)^2 - 0.0025,\right.$$
$$\left. 0 \le \left(x(t) - xp_3\right)^2 + \left(y(t) - yp_3\right)^2 - 0.0025\right]$$

```
Error, invalid input: subs received [t, x__i3], which is not
valid for its 1st argument
```

Optimization

```
> Optim :=
  Optim_Free := #Optimization WITHOUT obstacles
  Optim_set := Optim[2];
  Optim_set_Free := Optim_Free[2];
Error, `:=` unexpected
```

5) Obtain x__i(t) from optimization

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

$$\begin{bmatrix} 1 \,..\, 153\ Vector_{column} \\ Data\ Type: anything \\ Storage: rectangular \\ Order: Fortran\_order \end{bmatrix}$$

```
Error, invalid input: subs received Optim_set_Free, 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,%%): <%>:
  gen_coord_Free := evalf(subs(Optim_set_Free,tr_sub)): <%>:
  subs(t=1,%%): <%>:
Error, invalid input: subs received Optim_set, which is not
```

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

7) Find the two inputs

```
> the_controls := simplify(subs(Optim_set,x__i_final,eq_v)): <%>:
  the_controls_Free := simplify(subs(Optim_set_Free,
  x__i_final_Free,eq_v)): <%>:
Error, invalid input: subs received Optim_set, which is not
valid for its 1st argument
Error, invalid input: subs received Optim_set_Free, which is not
valid for its 1st argument
```

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) [m/s]"],
  title="Controls in time",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):
  P13 := display([P1,P3]):
  PY := plot(subs(gen_coord,x(t)), t = 0.. subs(data,T), color =
  "DarkMagenta"):
  PX := plot(subs(gen_coord,y(t)), t = 0.. subs(data,T), color =
  "DarkOrange"):
  PXY := display([PX,PY],gridlines=true, title = "X(t) & Y(t) in
  time WITH obstacles", labels = ["t [s]", "x(t),y(t) [m]"],
  thickness = 1):
  PTRAG := plot(subs(gen_coord_Free,sqrt((x(t))^2 + (y(t))^2)), t =
  0.. subs(data,T),color="DarkOrange",gridlines=true,labels=["t",
  "Traj(t) [m]"],title="Trajectory in time",axes=boxed):

  P1Free := plot([subs(gen_coord_Free[1],x(t)), subs(gen_coord_Free
  [2],y(t)), t = 0 .. subs(data,T)], color="DarkOrange",gridlines=
  true,labels=["x(t) [m]","y(t) [m]"],title="Trajectory in time,
  WITHOUT obstacles",axes=boxed):
  P2Free := plot([subs(the_controls_Free[1],v__1(t)), subs
  (the_controls_Free[2],v__2(t))], t = 0 .. subs(data,T),color=
  ["DarkOrange","DarkMagenta"],gridlines=true,labels=['t','v(t)'],
  title="Controls in time WITHOUT obstacles",axes=boxed,legend=
  ["v__1(t) [m/s]","v__2(t) [rad/s]"]):
  display(Array(1..2,[P1Free,P13]));
  display(Array(1..3,[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_Free, which is not
valid for its 1st argument
Error, invalid input: subs received gen_coord_Free[1], which is
not valid for its 1st argument
Error, invalid input: subs received the_controls_Free[1], which
is not valid for its 1st argument
Error, (in plots:-display) element 1 of the rtable is not a
valid plot structure
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 does 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 becomes 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 tha use clothoids are better than ours;

4) Maybe the problem is computazionally overdetermined in some parts, which makes the path non smooth and the shortest as possible. It may be better to not impose an initial and final attitude, taking it as another unknown to be optimized.