C11 SCARA Robot – MATLAB

Unsegmented Model / Environmental Level

Simulator: MATLAB is a widely used software tool based on numerical vector and matrix manipulation. For this solution "pure" MATLAB (without the graphical simulation blockset SIMULINK) was used.

Model: The model equations have to be programmed, in orderd sequence the right-hand side of the equations has to be provided for an ODE solver:

```
function dydt2 = f(t,y)
global m3l g kt Ra La u P D ......
T = (sqrt(3)/2)*u.*kt.*[y(7); y(8); y(9)]
% right side --------
dydt(1) = y(4); dydt(2) = y(5);
dydt(3) = y(6);
dydt(4) = T(1)+O(2)*(2*y(4)*y(5)+
           y(5)^2)*sin(y(2));
dydt(5) = T(2)-O(2)*y(4)^2*sin(y(2));
dydt(6) = T(3)-m3l*g;
```

Task a: Explicit and Implicit Modelling Techniques: The model was implemented in three various ways. First the implizit equation $M \cdot dy/dt = b(y,t)$ was implemented, as MATLAB's solvers allow implicit models of the type given, using options and providing the function $b(y,t)$ and the Mass matrix $M(y)$:

```
options = odeset('Mass','M(t,y)')
[t y] = ode15s('robotersys_im') ...
function varargout=robotersys_im(t,y,flag)
switch flag
case ''
   varargout{1} = f(t,y);
case 'mass'
   varargout{1} = mass(t,y);
case 'init'
   [varargout{1:3}] = init; end
function dydt2 = f(t,y) ...
function M = mass(t,y) ...
function [tspan,y0,options] = init ...
```

In the second approach the Gaussian algorithm was used to solve the linear equation $M(y) \cdot dy/dt = b(y,t)$ with respect to dy/dt in each integration step numerically.

This may take more time, but an explicit solver can be used, which could be faster then the implizit one necessary in the first approach.

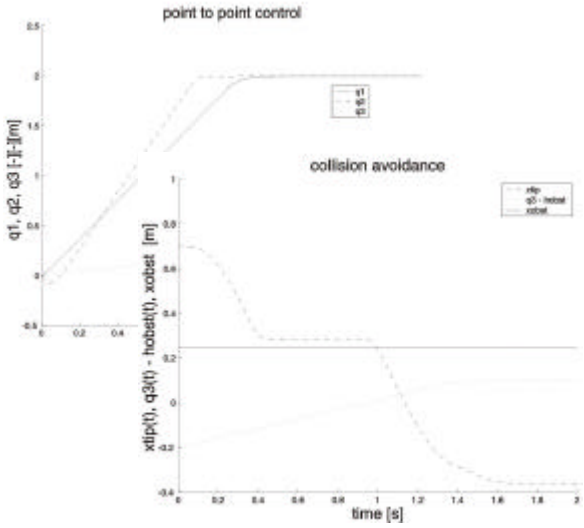
The third approach inverted the mass matrix symbolically. The explicit equation $dy/dt = M^{-1} \cdot b(y,t)$ can easily be solved by an explizit algorithm. The table compares the times for a 10 sec run (normalized):

Model / Algorithm	ode15s	ode23t	ode23tb
implicit	1.00	1.01	1.06
explicit / numerical	1.35	1.31	1.32
Explicit / symbolical	2.17	1.67	1.78

Task b: Pont-to-point control: The PD control was directly integrated in the function for the right side of the implizit implementation. In order to bound the voltage, a simple if assignment was used:

```
if y(6+i) > Imax(i) & dydt(6+i) > 0
dydt(6+i) = 0; end
```

So it was possible to get the current back from the limit without stopping the alorithm by using an event option (simulation results next fig.)



Task c: Collision avoidance: As at MATLAB level no event mechanism is available, for collison handling again voltage was also directly regulated in the function for the right side. In case of emergency, the PD control was used as if the final state was reached permanently, so that only the speed was regulated down. With an if assignment the emergency limits were realised (results see fig. above):

```
if (xtip-xobst) <= dcrit & y(3) < hobst
   U(1) = -D(1) * y(4);
   U(2) = -D(2) * y(5);
```

T. Piechatzek, H. Rekersbrink, TU Clausthal
timo.piechatzek@tu-clausthal.de
henning.\-rekersbrink@\-tu-clausthal.de