Comparison 11– MATLAB/SIMULINK
Numerical Inversion / Hybrid Approach

MATLAB is a widely used software tool based on numerical vector and matrix manipulation, SIMULINK is MATLAB’s extension for graphical modelling and numerical simulation of dynamic systems.

For this solution a hybrid modeling approach was chosen for task c: Execution of the SIMULINK model is controlled from the MATLAB environment.

Model Description (Task a): The robot model itself was implemented in two ways. First SIMULINK’s `Algebraic Constraint` block was used and the implicit equation $b(q,\dot{q}) - M(q_2)\cdot\ddot{q} = 0$ directly implemented (including control):

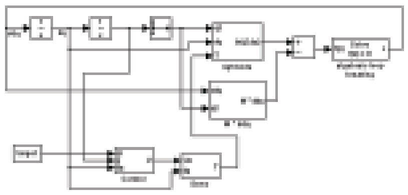

Fig.1: Implicit equations, algebraic loop breaking

For every integration step SIMULINK’s Algebraic Constraint block 'searches' for a solution of the implicit equation. This procedure is comfortable and does also work in the presence of a Hit Crossing block (which was needed for task c)!

For the second (explicit) solution the systems mass matrix *M* was inverted symbolically outside MATLAB and the explicit equation $\ddot{q} = M(q_2)^{-1}\cdot b(q,\dot{q})$ implemented in SIMULINK (including control), resulting in a SIMULINK model without a block for algebraic loop breaking.

Both SIMULINK models are controlled at the MATLAB environment by an m-file. All parameters and constants are defined and all plots produced by that m-file.

Point to Point Control (Task b): For point to point movement both solutions use the same controller. A target vector is the input for submodel `Control` which contains the PD-Controller. The output `u` is fed into submodel `Servo` which models the servo drives. Finally the resulting torque `T` is provided for the calculation of the right hand side `b`. `U` is bounded by a `Saturation` block inside the submodel `Control`. Figure 2 shows the results graphically, the following table gives the processing times (using the commands 'tic' and 'toc'):

Model description	Norm. CPU-time
Task b) explicit - inverted matrix	1 (2.23s at PII 400)
Task b) implicit - algebraic loop breaking	4.09

Obstacle Avoidance (Task c): For collision avoidance a hybrid approach was chosen: MATLAB takes control over two phases (save and danger), changes parameters when necessary and starts the appropriate SIMULINK model.

Submodel `Control` was extended for both solutions. The distance between the obstacle and the tool tip is permanently checked. Depending on the status `save` execution of the model is stopped if the robot enters or leaves the restricted area (`Hit-Crossing` block). Control is then handed back to MATLAB. On entering, the active m-file changes the target positions for the state-variables to the current position and allows the emergency maximum voltages. The SIMULINK model is restarted with the changed parameters for the emergency manoeuvre. Just after the tool tip of the robot has reached an admissible height the SIMULINK model stops again. At MATLAB the original target position is reactivated and the SIMULINK model restarted again. This procedure could be repeated. All initial conditions for the integrators are updated with each restart of the SIMULINK model.

A part of the m-file: Change of parameters and start of the SIMULINK model for emergency manoeuvre

```
% prepare parameters for emergency manoeuvre
index=length(Ti); time=T(index); targetold=target;
target=[position(1:2),target(3)];
Umax=[U1mm,U2mm,...
dq10=DQ(index,1); ...q10=Q(index,1);
...I10=I(index,1);...
save=0; danger=1;
% start model for emergency manoeuvre
[Ti]=sim('C_11v2C_EX',[time,2]);
position=Q(length(Ti),1:3); ...
```

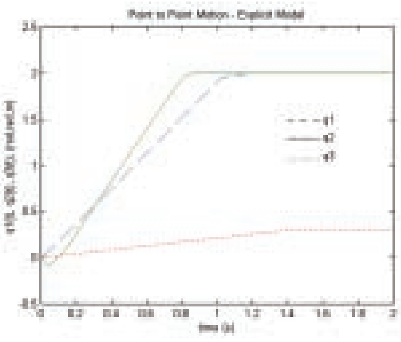

Fig. 2: Joint positions

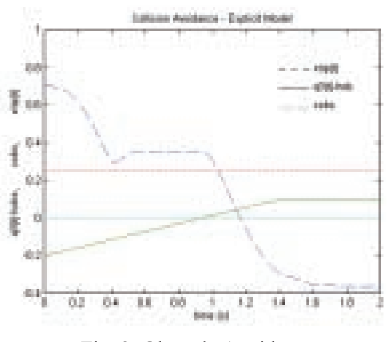

Fig. 3: Obstacle Avoidance

J. Scheikl, SIMTECH / ARGESIM, TU Vienna, Wiedner Hauptstr. 8-10, A-1040 Vienna, email: joxg@osiris.tuwien.ac.at