Simulation of *n*-r robots

POPESCU MARIUS-CONSTANTIN, BORCOŞI IILIE, OLARU ONISIFOR, ANTONIE NICOLAE
Department of Electromechanics, University of Craiova, Romanian,
Department of Automatics and Informatics
University of C-tin Brancusi
str. Geneva nr.3, Tg-Jiu, Gorj
ROMANIA
popescu_ctin2006@yahoo.com, ilie_b@utgjiu.ro,olaru@utgjiu.ro, nicolae.antonie@utgjiu.ro,
http://www.em.ucv.ro, http://www.utgjiu.ro/ing

*Abstract***.** Modelling and simulation of *n*-DOF robots with revolute joints are presented in the paper. First, numerical effective equations to solve the direct kinematics of *n*-degree-of-freedom robots are derived. Following this, the dynamic model based on Lagrangian equation is developed. In the model the characteristics of robots are utilized and therefore the complexity of the model is increasing slower with the number of degrees of freedom compared with the general type of manipulators. Next, some cost functions for *n*-R robots and their gradients are presented. The derived models represent the basis of the software package "Robotic Toolbox" implemented in Matlab. The package allows the user to create and simulate different systems considering robots. Matlab functions and Simulink blocks provided include forward kinematics, dynamics and several utility functions.

Key-words: Modelling; Redundant robots; Simulation *n*-r robots.

1. INTRODUCTION

One of the important issues of the new generation of robotic manipulators is redundancy. Most of the study of the redundant mechanisms has been performed without considering any particular mechanism [9,12]. A theoretical approach to the control problem usually incorporates a dynamic model of a mechanism [8,5]. Furthermore, the schemes with pure cinematic redundancy resolution also often use dynamic models or their parts in performance criterions [9,12]. Some authors have proposed a seven-degree-of-freedom (DOF) mechanism [4], or a four-DOF mechanism [6], but most of them use a simple three-DOF planar mechanism [5], [16], [8], [7], [2], [11]. Actually, most authors use in their analysis mechanical systems with only one degree of redundancy. The reason is in the complexity of mechanisms models, which increases rapidly with the number of DOF. For the evaluation of theoretical results it would be of benefit to have dynamic models of mechanisms with more than one degree of redundancy

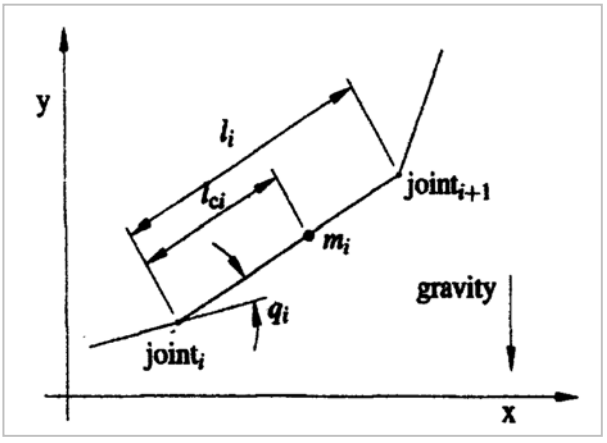

Fig. 1. Structure of the n-DOF planar robot.

In this paper we present cinematic and dynamic models of *n*-DOF planar manipulator (robot) with revolute joints, the *n*-R manipulator. First, numerical effective equations to solve the direct kinematics of *n*-DOF robot are developed utilizing the characteristics of planar manipulators. Namely, the components of the Jacobian matrix and the Hessian matrix can be specified by the previous obtained results without any additional calculation.

Next, the dynamic model based on Lagrangian equation is derived. In the next section some commonly used cost functions for *n*-R robot and their gradients are derived. In the second part, a software package for simulation of n-R robot is presented. The package is implemented in Matlab/Simulink and consists of several M-files and Simulink blocks for the calculation of the robot model and several utility functions derived from the robot model like cost functions and functions for the representation of the robot in the task plane [15]. At the end, some examples showing how to use the derived functions and blocks in Matlab and in Simulink are presented.

2 Problem Formulation

In the following subsections, we derive the cinematic and the dynamic model of the *n*-DOF robot with revolute joints. The manipulator is supposed to move in the vertical plane *x*-*y* as shown in Fig.1.The task coordinates $\boldsymbol{x}$ are the positions in *x*-*y* plane, $\boldsymbol{x}=[x, y]^T$.

2.1. Kinematics of *n*-R robot

With respect to *n* joint coordinates $\boldsymbol{q}$, and *m* task coordinates $\boldsymbol{x}$, the kinematics of the manipulator can be described with the following equations [1]:

$$\boldsymbol{x}=\boldsymbol{p}(\boldsymbol{q}),\ \dot{\boldsymbol{x}}=\mathbf{J}(\boldsymbol{q})\,\dot{\boldsymbol{q}}\,,\ \ddot{\boldsymbol{x}}=\mathbf{J}(\boldsymbol{q})\,\dot{q}+\mathbf{J}(\boldsymbol{q},\dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}}\,, \qquad (1)$$

where $\boldsymbol{p}$ is an *m*-dimensional vector function representing direct kinematics, **J** is the Jacobian matrix and $\dot{\boldsymbol{J}}$ is its time derivative, $\dot{\boldsymbol{J}}=\mathrm{d}\mathbf{J}/\mathrm{dt}$;. As we deal with redundant manipulators, $n>m$ and **J** is an *m*x*n* matrix. Let φ be an *n*-dimensional vector with components:

$$\varphi_i=\varphi_{i-1}+q_i \qquad (2)$$

for $i=1,...,n$ and initial value $\phi_0=0$, and l_i be the length of the *i*th link. Following this, in the case of a planar robot ($m=2$) with revolving joints, the end effectors positions $\boldsymbol{x}$, $\boldsymbol{x}=[x_1, y_1]^T$, can be expressed by the following recursive equations [10]:

$$x_i=x_{i+1}+l_i\cos(\varphi_i)\ \text{ and }\ y_i=y_{i+1}+l_i\sin(\varphi_i) \qquad (3)$$

for $i=n-1,...,1$ and initial values $x_n=l_n\cos(\varphi_n)$ and $y_n=l_n\sin(\varphi_n)$. The pairs $[x_1, y_1]^T$ represent the position of the end of the manipulator measured from the joint *i*. Next we calculate the Jacobian matrix **J**. In the planar case, the Jacobian **J** is a 2x*n* matrix:

$$\mathbf{J}=\begin{bmatrix}\frac{\partial x_1}{\partial q_1} & \cdots & \frac{\partial x_1}{\partial q_n}\\ \frac{\partial y_1}{\partial q_1} & \cdots & \frac{\partial y_1}{\partial q_n}\end{bmatrix}.$$

From Eqs. (2) and (3), it follows that:

$$\frac{\partial x_i}{\partial q_j}=-y_k\ \text{ and }\ \frac{\partial y_i}{\partial q_j}=x_k\,, \qquad (4)$$

where $k=\max(i, j)$. {Note that $x_i=\sum_{k=i}^{n}\left[l_k\cos\left(\sum_{j=i}^{k}q_j\right)\right]$} Hence, the components of **J** can be specified without any further calculations as:

$$\mathbf{J}=\begin{bmatrix}-y_1 & \cdots & -y_n\\ x_1 & \cdots & x_n\end{bmatrix}. \qquad (5)$$

For the complete model, we have to derive $\dot{\boldsymbol{J}}$, i.e. to differentiate **J** with respect to time:

$$\dot{\boldsymbol{J}}=\sum_{k=1}^{n}\left(\frac{\partial\mathbf{J}}{\partial q_k}\dot{q}_k\right). \qquad (6)$$

Differentiating Eq. (4) with respect to $\boldsymbol{q}$ yields:

$$\frac{\partial^2 x_i}{\partial q_j\partial q_k}=-x_r\ \text{ and }\ \frac{\partial^2 y_i}{\partial q_j\partial q_k}=-y_r \qquad (7)$$

where $r=\max(i,j,k)$.Substituting Eq.(7) into Eq.(6) yields:

$$\dot{\boldsymbol{J}}=\begin{bmatrix}\dot{\mathbf{q}}^T\begin{bmatrix}-x_1 & -x_2 & \cdots & -x_n\\ -x_2 & -x_2 & \cdots & -x_n\\ \cdots & \cdots & \cdots & \cdots\\ -x_n & -x_n & \cdots & -x_n\end{bmatrix}\\ \hline \dot{\mathbf{q}}^T\begin{bmatrix}-y_1 & -y_2 & \cdots & -y_n\\ -y_2 & -y_2 & \cdots & -y_n\\ \cdots & \cdots & \cdots & \cdots\\ -y_n & -y_n & \cdots & -y_n\end{bmatrix}\end{bmatrix}. \qquad (8)$$

As we can see from Eqs. (5) and (8), the only elements of a cinematic model of *n*-R robot to be calculated are x_i and y_i. All other parts of the model can be expressed in terms of x_i and y_i.

2.2. Dynamics of *n*-R robot

Before we determine the components of the dynamic model, we have to derive expressions for the position of the centerpoint of the mass and corresponding Jacobian matrices for all segments. Using Eq. (3) the position of the centerpoint of the mass (COM) of the *i*th link can be denned by:

$$\mathbf{x}_{ci}=\begin{bmatrix} x_1 - x_i + l_{ci}\cos(\varphi_i) \\ y_1 - y_i + l_{ci}\sin(\varphi_i) \end{bmatrix}. \qquad (9)$$

Note that $[x_1\text{-}x_i, y_1\text{-}y_i]^T$ represents the position of the joint *i* measured from the base of the robot. Differentiating Eq. (9) with respect to *q* yields:

$$\frac{\partial x_{ci}}{\partial q_j}=\begin{cases} -y_j + y_i + l_{ci}\sin(\varphi_i), & j \le i \\ 0, & j > i \end{cases}$$

$$\frac{\partial y_{ci}}{\partial q_j}=\begin{cases} -x_j + x_i + l_{ci}\cos(\varphi_i), & j \le i \\ 0, & j > i \end{cases} \qquad (10)$$

The Jacobian matrices related to the segments have been divided into two parts:

$$\mathbf{J}=\begin{bmatrix} \mathbf{J_L} \\ \mathbf{J_A} \end{bmatrix},$$

where $\mathbf{J_L}$ and $\mathbf{J_A}$ are parts of $\mathbf{J}$ associated with linear and angular task velocities, respectively. The Jacobian matrices $\mathbf{J_L^{(i)}}$ associated with the center-point of the mass of the *i*th link are defined as:

$$\mathbf{J_L^{(i)}}=\begin{bmatrix} \frac{\partial x_{ci}}{\partial q_1} & \cdots & \frac{\partial x_{ci}}{\partial q_n} \\ \frac{\partial y_{ci}}{\partial q_1} & \cdots & \frac{\partial y_{ci}}{\partial q_n} \end{bmatrix}$$

and can easily be obtained by substituting Eq. (10) into the above equation. Next, the derivatives of $\mathbf{J_L^{(i)}}$ with respect to $\boldsymbol{q}$ are calculated:

$$\frac{\mathbf{J}_L^{(i)}}{\partial q_k}=\begin{bmatrix} \frac{\partial^2 x_{ci}}{\partial q_1 \partial q_k} & \cdots & \frac{\partial^2 x_{ci}}{\partial q_n \partial q_k} \\ \frac{\partial^2 y_{ci}}{\partial q_1 \partial q_k} & \cdots & \frac{\partial^2 y_{ci}}{\partial q_n \partial q_k} \end{bmatrix}$$

using the relationships

$$\frac{\partial^2 x_i}{\partial q_j \partial q_k}=\begin{cases} -x_r + x_i + l_{ci}\cos(\varphi_i), & j \le i \text{ and } k \le i \\ 0, & j > i \text{ or } k > i \end{cases}$$

$$\frac{\partial^2 y_i}{\partial q_j \partial q_k}=\begin{cases} -y_r + y_i + l_{ci}\sin(\varphi_i), & j \le i \text{ and } k \le i \\ 0, & j > i \text{ or } k > i \end{cases}$$

where r=max(j, k). For convenience, the following notation has been introduced:

$$\mathbf{Q^{(i)}}=\mathbf{J_L^{(i)\,T}}\mathbf{J_L^{(i)}};\ \boldsymbol{\Psi}_\mathbf{k}^{(\mathbf{i})}=\left(\frac{\partial J_L^{(i)}}{\partial q_k}\right)^T\mathbf{J_L^{(i)}}+\mathbf{J_L^{(i)\,T}}\left(\frac{\partial \mathbf{J_L^{(i)}}}{\partial \mathbf{q_k}}\right) \qquad (11)$$

The dynamic model is based on the Lagrangian formulation. The equations of the motion of the robot are given by:

$$\frac{d}{dt}\frac{\partial \mathbf{T}}{\partial \dot{\mathbf{q}}}-\frac{\partial \mathbf{T}}{\partial \mathbf{q}}=\boldsymbol{\tau},$$

where T is the total energy of the system and $\boldsymbol{\tau}$ are generalized forces corresponding to the joint coordinates $\boldsymbol{q}$. After some calculations, the dynamic equations of the rigid robot can be given in the closed form:

$$\boldsymbol{\tau}=\mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\boldsymbol{h}(\boldsymbol{q},\dot{\boldsymbol{q}})+\mathbf{B}\dot{\boldsymbol{q}}+\boldsymbol{g}(\boldsymbol{q}), \qquad (12)$$

where $\mathbf{H}$ is the inertia matrix, $\boldsymbol{h}$ is the vector of Coriolis and centrifugal forces, $\mathbf{B}$ is the matrix of viscous friction coefficients and $\boldsymbol{g}$ is the vector of gravity forces. The detailed derivation of the above equations is described in [1]. In Eq. (12), the matrix $\mathbf{H}$ is given by:

$$\mathbf{H}=\sum_{i=1}^{n}(m_i\ \mathbf{J_L^{(i)\,T}}\mathbf{J_L^{(i)}}+\mathbf{J_A^{(i)\,T}}\mathbf{J_A^{(i)}}),$$

where m_i is the mass of the *i*th link. In the case of a planar manipulator with revolute joints, it can be proved that terms $\mathbf{J_L^{(i)\,T}}\mathbf{I}_\mathbf{i}\mathbf{J_L^{(i)}}$ simplify to:

$$\mathbf{J_L^{(i)\,T}}\mathbf{I}_\mathbf{i}\mathbf{J_L^{(i)}}=I_i\begin{bmatrix} \mathbf{1}_{i\times i} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{n\times n},$$

where I_i is the moment of inertia of the link *i* and $\mathbf{1}$ and $\mathbf{0}$ are matrices of ones and zeros of corresponding dimensions, respectively. Next, the components of the vector of Coriolis and centrifugal forces $\boldsymbol{h}$ can be expressed by: $h_i=\sum_{j=1}^{n}\sum_{k=1}^{n}h_{ijk}\dot{q}_j\dot{q}_k$ and $h_{ijk}=\frac{\partial \mathbf{H}_{ij}}{\partial q_k}-\frac{1}{2}\frac{\partial \mathbf{H}_{jk}}{\partial q_i}$.

Using Eq. (11) the vector ***h*** can rewritten into a more suitable form:

$$\mathbf{h} = \sum_{i=1}^{n} m_i \left(\left(\sum_{j=1}^{n} \psi_j^{(i)} \dot{q}_j \right) \dot{q} - \frac{1}{2} \begin{bmatrix} \dot{q}^T \psi_1^{(i)} \\ \dot{q}^T \psi_2^{(i)} \\ \cdots \\ \dot{q}^T \psi_n^{(i)} \end{bmatrix} \dot{q} \right).$$

As a last step, the components of the vector of gravity forces g are calculated. In general, they are defined as:

$$g_i = \sum_{j=1}^{n} m_j \mathbf{g}_{acc}^T \mathbf{J}_{Li}^{(j)},$$

where $\mathbf{g}_{acc}^T$ is the vector representing the acceleration of gravity, but in the case of *n*-R planar robots it is of benefit to calculate the components of the vector ***g*** recursively. Namely, the gravity force of link *i* is equal to the gravity force of the link *i*+1 increased by the contribution of the link *i*. Thus, the components g_i are:

$$g_i = g_{i+1} + g_{acc} \left(m_i l_{ci} + \sum_{k=i+1}^{n} m_k l_i \right) \cos(\varphi_i) \quad (13)$$

for *i*= *n* -1,..., 1 and the initial value $g_n = g_{acc} m_n l_{cn} \cos(\varphi_n)$ and g_{acc} is the acceleration of gravity.

3 Problem Solution

Optimization is often part of the motion planning and the control design of redundant mechanisms. A very common approach is redundancy resolution at the velocity level based on the equation [12]:

$$\dot{\mathbf{q}} = \mathbf{J}^{+} + \dot{\mathbf{x}}_d + k(\mathbf{I} - \mathbf{J} + \mathbf{J})\dot{\Phi}, \quad (14)$$

where $\dot{\mathbf{x}}_d$ is the desired task velocity, $\mathbf{J}^{+}$ is the generalized inverse, $\dot{\Phi}$ is an arbitrary joint velocity vector and $(\mathbf{I}\text{-}\mathbf{J}^{+}\mathbf{J})\dot{\Phi}$ is its projection into the null space of **J**, corresponding to the self motion of the mechanism and *k* is an arbitrary constant. The first part of Eq. (14) is the least normal solution and assures the motion along the path. The second part is the homogeneous solution which moves the robot in the null space of **J**, e.g. toward the optimal configuration. The scalar gain *k* is used to tune the null space motion. In the following, we present some cost functions which are common in the study of the redundant systems.

3.1. Singularity avoidance

It is well known that pseudo-inverse control does not avoid singularities. Furthermore, minimization of some cost functions may push the manipulator to the singularities. Therefore, additional measures for the singularity avoidance should be used. One possibility is the condition number of **J** [16]. The condition number is the ratio between the maximal and minimal singular values of $\mathbf{JJ}^{\mathbf{T}}$:

$$\rho = \frac{\sigma_{\max}}{\sigma_{\min}}. \quad (15)$$

The other commonly used measure is the manipulability measure [16] defined as:

$$. \ w = \sqrt{\sigma_1 \sigma_2 \cdots \sigma_m} = \sqrt{\det\left(\mathbf{JJ}^{\mathbf{T}}\right)}. \quad (16)$$

In the case of planar manipulator $\mathbf{JJ}^{\mathbf{T}}$ is a 2x2 symmetric matrix :

$$\mathbf{JJ}^{\mathbf{T}} = \begin{bmatrix} a^{11} & a^{12} \\ a^{21} & a^{22} \end{bmatrix}$$

and the two singular values of $\mathbf{JJ}^{\mathbf{T}}$ are

$$\sigma_{1,2} = \frac{a^{11} + a^{22} \pm \sqrt{\left(a^{11} - a^{22}\right)^2 + 4\left(a^{12}\right)^2}}{2} \quad (17)$$

Hence, substituting Eq. (17) into Eq. (15) yields:

$$\rho = \frac{a^{11} + a^{22} + \sqrt{\left(a^{11} - a^{22}\right)^2 + 4\left(a^{12}\right)^2}}{a^{11} + a^{22} - \sqrt{\left(a^{11} - a^{22}\right)^2 + 4\left(a^{12}\right)^2}} \quad (18)$$

and into Eq. (16) yields

$$w = \sqrt{a^{11} a^{22} - \left(a^{12}\right)^2}. \quad (19)$$

The components a^{ij}, $a^{ij} = a_1^{ij}$ can be easily obtained using the following equations [17,10]:

$$a_i^{11} = a_{i+1}^{11} + y_i^2$$

$$a_i^{22} = a_{i+1}^{22} + x_i^2$$

$$a_i^{12} = a_{i+1}^{12} - x_i\, y_i \qquad (20)$$

where the initial values are $a_{n+1}^{11} = a_{n+1}^{22} = a_{n+1}^{12} = 0$. To move away from the singularity we have to minimize ρ or maximize w. Following the basic idea the velocity $\dot{\Phi}$ in Eq. (14) should be selected in the case of the condition number as:

$$\dot{\Phi} = \nabla\rho = \frac{\sigma_2 \nabla\sigma_1 - \sigma_1 \nabla\sigma_2}{\sigma_2^2}, \qquad (21)$$

or when the manipulability is used as

$$\dot{\Phi} = \nabla w = \frac{a^{11}\nabla a^{22} + a^{22}\nabla a^{11} - 2\nabla a^{12}\nabla a^{12}}{2\sqrt{a^{11}a^{22} - \left(a^{12}\right)^2}}. \qquad (21)$$

Differentiating a^{ij} yields:

$$\frac{\partial a^{11}}{\partial q_j} = 2\left(x_j \sum_{k=1}^{j-1} y_k - a_j^{12} \right)$$

$$\frac{\partial a^{22}}{\partial q_j} = 2\left(- y_j \sum_{k=1}^{j-1} x_k - a_j^{12} \right)$$

$$\frac{\partial a^{12}}{\partial q_j} = -x_j \sum_{k=1}^{j-1} x_k + j_i \sum_{k=1}^{j-1} y_k + a_j^{11} + a_j^{22}$$

and the results are then used in the calculation of the gradient of the cost functions [in Eq. (20) or Eq. (21)].

3.2. Joint torques optimization

Considering only the gravity term on the right side of Eq. (12), the performance criterion function p representing the weighted norm of joint torques can be expressed as:

$$p = \boldsymbol{g}(\boldsymbol{q})^{\mathrm{T}} \mathbf{W} \boldsymbol{g}(\boldsymbol{q}), \qquad (22)$$

where $\mathbf{W}$ is the matrix of weights. To optimize p, the vector $\dot{\Phi}$ can be selected as:

$$\dot{\Phi} = \nabla p(\mathbf{q}) = 2\left(\frac{\partial \mathbf{g}(\mathbf{q})}{\partial \mathbf{q}} \right)^T \mathbf{W}\mathbf{g}(\mathbf{q}). \qquad (23)$$

Considering Eq. (13), the gradient of gravity forces is calculated as:

$$\frac{\partial g_i}{\partial q_j} = \frac{\partial q_{i+1}}{\partial q_j} - \begin{cases} g_{acc}\left(m_i l_{ci} + \sum_{k=i+1}^{n} m_k l_i \right) \sin(\varphi_i), j \le i \\ 0, j > i \end{cases} \qquad (24)$$

for $i=n$-1,...,1 and initial value

$$\frac{\partial g_n}{\partial q_j} = -g_{acc} m_n l_{cn} \sin(\Phi_n).$$

Thus, using Eqs. (13) and (24) the gradient of the cost function Eq. (23) can be easily obtained.

3.3. Simulation

There exist several simulation packages for robot analysis and control design like "A Robotic Toolbox" [3] and "A Toolbox for Simulation of Robotics Systems" [16] which are implemented in Matlab, or "Robotica" [13], [14] which is based on Mathematica. All these packages support the general manipulator structures and are rather complex. To utilize all the simplifications in the model calculation which are enabled due to the special structure of the planar manipulators, we have decided to develop a special software package "Robotic Toolbox" for dynamic simulation of n-DOF robots with revolute joints. The package has been implemented in Simulink. Matlab has been selected mainly due to its capabilities of solving problems with matrix formulations, easy extensibility and because of the possibility to simulate in real-time.

3.3.1. Implementation in Matlab

The mathematical models derived in the previous sections form the basis of the simulation system. The main functions are:

[x,J,Jd]=kinmodel(q,qd,L) for the calculation of elements of the kinematic model $\{x, \mathbf{J}, \dot{\mathbf{J}}\}$; [links]=kin_link(q,L) for the calculation of task positions of all links; [x,J,Jd,H,h,g]=dynmodel(q,qd,L,Lc,m,ml,ll,B) for the calculation of elements of the complete model (x, $\mathbf{J}$, $\dot{\mathbf{J}}$, $\mathbf{H}$, $\boldsymbol{h}$, $\boldsymbol{g}$); [Man,GradMan]=idxman(q,L) for

the calculation of the manipulability w and its gradient; [CN, GradCN] =idxcn(q,L) for the calculation of the condition number ρ and its gradient and [Grav,GradGrav]=idxgrav(q,L,Lc,m,

ml,grav,W) for the calculation of the norm of joint torques due to the gravity p and its gradient.

Table 1 Parameters of the robot

Parameter	Example for 4-R robot
Initial joint position q	q = [0.5;1;-1;-1]*pi/2
Initial joint velocity $\dot{q}$	qd= [0;0;0;0l
Link lengths l	L = [1 1 1 1]
Link COM/c	L_c = [0.4 0.3 0.3 0.25]
Link masses m	m = [2 1 1 0.5]
Load mass mt	m_1 = [1 1 1 1]
Link inertias I	ll= [0.3 0.2 0.2 0.11]
Viscous friction coefficient **B**	Bv = [0.05 0.04 0.04 0.04]

The parameters of the robot needed in the calculation of the above functions are given in the Table 1. These functions can be used together with the standard Matlab functions in different areas like kinematics, dynamic or path planning. For example, to calculate the direct kinematics the following commands should be used:

```
>> nj= ;L=[1 1 1 1]; q=[0.5;1;-1;-1]*pi/2;

>> qd=[1; 1; 1; 1]; qdd=[1;0;-1;2];

>> [x,J,Jd]=kinmodel(q,qd,L); xd=J*qd;

>> xdd=J*qdd; disp([x xd xdd])
```

To find the configuration of the robot with maximal manipulability for the same task position the cinematic control with the Adams (ode113) integration is used:

```
>> while norm(qd)>0.001 [x,J]=kinmodel (q,L);

J1=J'/(J*J');  NS=(eye(nj)-J1*J);

[Man,GradMan]=idxman (q,L);

qd=10*NS*GradMan; q=q+qd*0.001; end

>> disp (q)
```

As no task motion is desired only the second term of Eq. (14) is applied for the control. The figure of the optimal manipulator configuration is obtained by using the following commands (see Fig. 2)

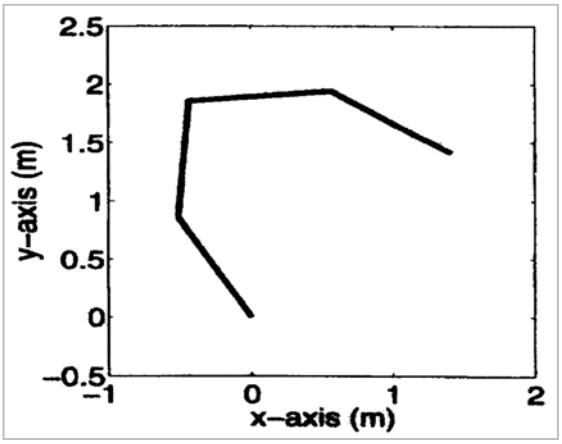

Fig. 2. Configuration of 4-R robot with maximal manipulability.

```
>> xx=kin_link (q,L);ll=plot(xx(1,:),xx(2,:)); grid

>> axis equal, axis square, axis([ —1 252.5])

>> xlabel('x-axis'), ylabel('y-axis')
```

3.3.2. Computational complexity

As described in the previous sections, most of the equations of the model use recursion and are in the form of loops. To obtain the most speed of the Matlab it is recommended to vectorize all the loops. For example, Eqs. (2) and (3) can be transformed into the form of

```
fi=(robll*q)'; cL=L*cos(fi); sL=L*sin(fi);

x=cL*robll; y=sL*robll; X=[x(l),y(l)];
```

where robll is a lower-left triangular matrix of ones. As expected, the vectorized form is faster. Unfortunately, the increase in the calculation speed is not significant because it is not possible to vectorize all the loops and because the vectorization itself increases the number of numeric operations. On the other hand, the significant performance improvements can be achieved by using MEX-files instead of M-files. Using the Matlab Compiler, M-functions can be compiled to MEX-files. As the

Matlab compiler compiles loops very efficiently it is better not to vectorize those parts where the vectorization introduces additional operations. Table 2 shows the execution time t_e versus number of DOF of the robot cinematic and dynamic model realized as M-file and MEX-file. It is obvious that the MEX-file version is significantly faster than the M-file version. We can also see that the execution time t_e for calculation of the cinematic model increases with the number of DOF as $O(n^2)$ and the dynamic model as $O(n^3)$ for both M-file and MEX-file.

Table 2. The execution time t_c versus number of DOF for cinematic and complete model for M-file and MEX- file

	Dynmodel		Kinmodel	
	M-file	MEX-file	M-file	MEX-file
n	t_e(ms)	t_e(ms)	t_e(ms)	t_e(ms)
2	11	1,1	5,0	0,38
3	27	1,7	7,2	0,44
4	38	2,7	9,8	0,50
5	55	3,8	14,3	0,60
10	215	23,1	44,0	1,10
20	1197	276,9	159,0	3,30

3.3.3. Integration into Simulink

As an extension to Matlab, Simulink adds many features for the easier simulation of the dynamic systems, e.g. graphical model building and selection of the integration method and parameters. To exploit additional features, we have developed several blocks and functions needed to create cinematic and dynamic models and to simulate the motion of *n*-R robots in Simulink. Fig. 3 shows the Simulink part of the "Robotic Toolbox". To gain the transparency the system should be represented by the block structure with several hierarchical levels. In the case of robot systems the top level represents the close loop model. The corresponding block diagram is given in Fig. 4. Under the Robot block, the cinematic or dynamic model of the robot is modelled. For example, using Eq. (12) the dynamic model of the robot can be built as it is shown in Fig.5.

The block Model is realized as the S-function (calling the M-function dynmodel) and includes the calculation of model matrices and vectors $\boldsymbol{x}$, $\mathbf{J}$, $\dot{\mathbf{J}}$, $\mathbf{H}$, $\boldsymbol{h}$ and $\boldsymbol{g}$. The two integrators are used to obtain the joint velocities and positions. As it can be seen from Fig. 5 the model calculations require some matrix operations for the calculation of the joint accelerations and for the direct cinematic transformations (inversion, product, etc.). As the standard Simulink does not provide any blocks for the matrix operations we had to write the necessary blocks. To make them fast they are all C-MEX files. Other special blocks in the Simulink library provide functions specific for robots which cannot be created using the standard Simulink blocks. Two special blocks are used for the representation of the end-effector position in the task space and for the on-line animation of the manipulator motion (Fig. 4). They plot the signals in separate windows as it is shown in Fig. 6.

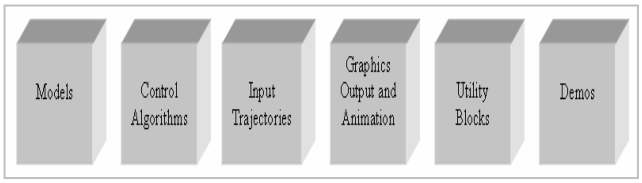

Fig. 3. Robotic Toolbox : Simulink block library

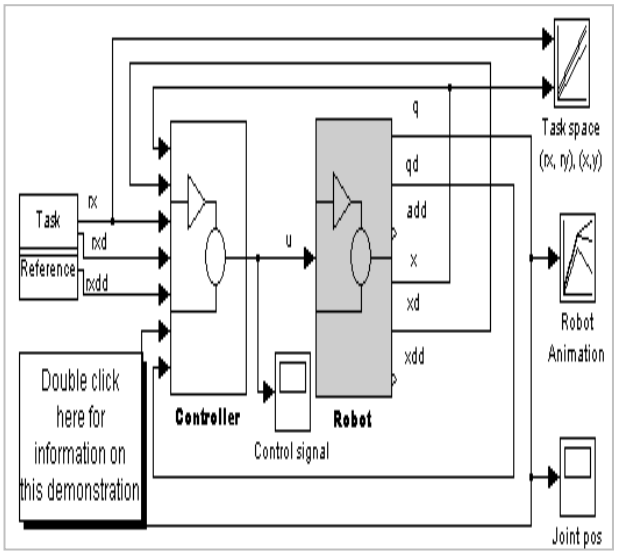

Fig. 4 Simulink block diagram for general close loop system.

3.3.4. Simulation example

For illustration of the capabilities of the "Robotic Toolbox", we have chosen a dynamic simulation of a 7-R manipulator moving along a straight line.
The redundancy resolution has been performed at the acceleration level and the control law is given in the form [12]:

$$\tau = \mathbf{H}(\mathbf{J}^{+}(\ddot{\mathbf{x}}_d+\mathbf{K}_v\dot{\mathbf{e}}-\dot{\mathbf{J}}\dot{\mathbf{q}})+(\mathbf{I}-\mathbf{J}+\mathbf{J}\Phi)+\boldsymbol{h}+\boldsymbol{g}. \quad (25)$$

where $\boldsymbol{e}=\boldsymbol{x}_d-\boldsymbol{x}$ is the tracking error, $\dot{x}_d$ is the desired task space acceleration, $\mathbf{K}_v$ and $\mathbf{K}_p$ are constant gain matrices and $\mathbf{J}^{+}$ is the inertia weighted pseudo-inverse, $\dot{\mathbf{J}}^{+}=\mathbf{H}^{-1}\mathbf{J}^{T}(\mathbf{J}\mathbf{H}^{-1}\mathbf{J}^{T})^{-1}$.
The null space vector Φ is used to stabilize the null space motion and is defined as [7]:

$$\Phi = -\mathbf{K}_n\dot{q} - \frac{d\left(I-J^{+}J\right)}{dt}\dot{q}, \quad (26)$$

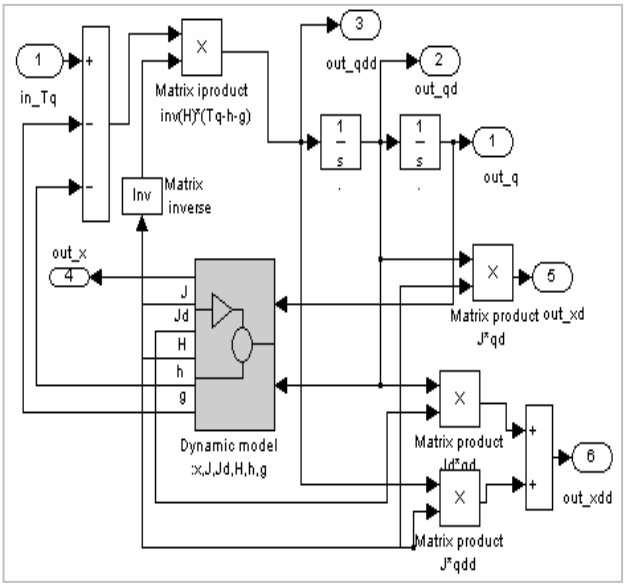

Fig. 5 Simulink block diagram for the calculation of the dynamic model of the robot

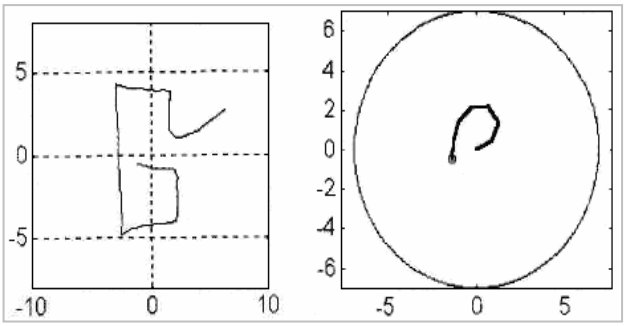

Fig. 6. Task space (.v, y) and manipulator animation windows for the presentation of the manipulator motion in the task plane.

where $\mathbf{K}_n$ is a gain matrix. Fig. 7 shows the SIMULINK block structure of the system. As we can see, the matrix blocks enable straightforward model generation from the Eqs. (25) and (26).

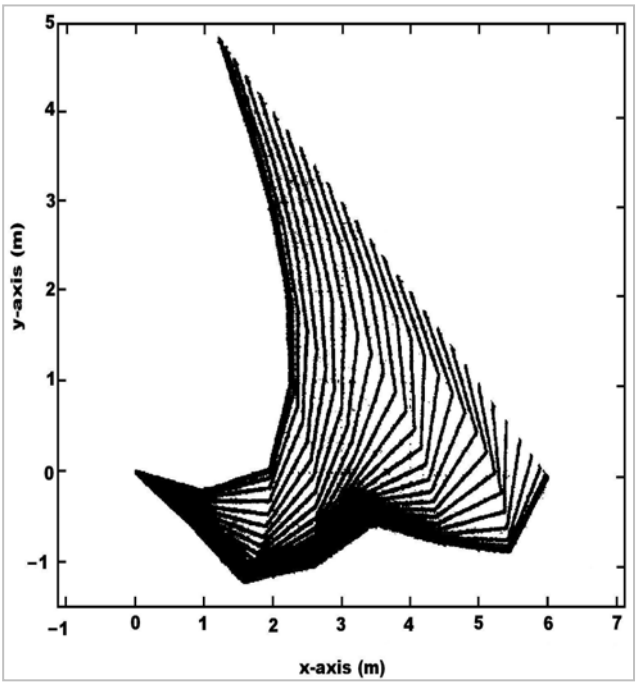

Fig. 8. Tracking of a line: configurations of the robot in the task plane.

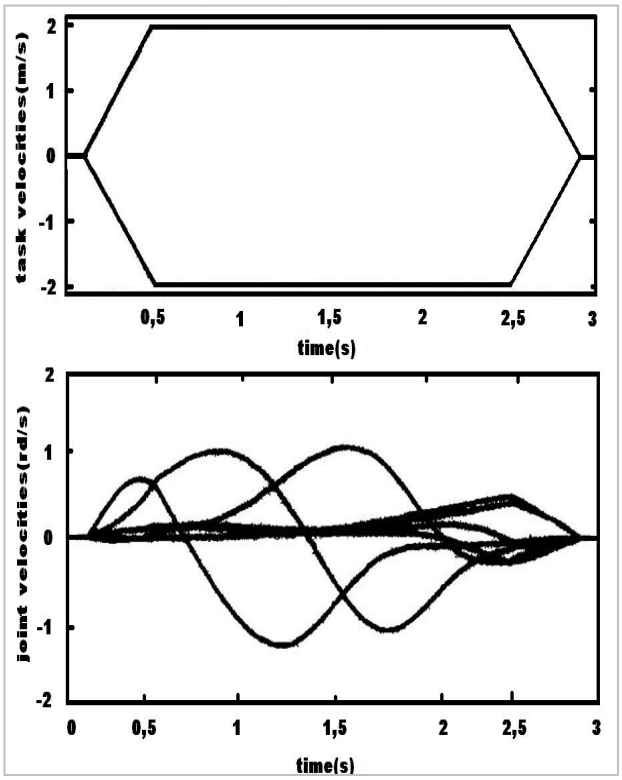

Fig. 9. Tracking of a line: task and joint velocities.

Two dynamic models are used in the system: one for the calculation of the robot model and one in the control algorithm. As they can have different parameters, the user can simulate the errors in the model parameters. The simulation can be performed from the Simulink menu or from the command line by using

>> [t,x] = ode45('trace',3,[],[0.001 0.01 0.001])

where trace is the name of the Simulink model.

The selected system states are saved in the variables T, rxt, xt, qt and ut using To Workspace blocks. The analysis of the simulation results is then made in Matlab. For example, by using the following commands:

>> xx=kin_link(qt(1,:),L); ll=plot(xx(1,:),xx(2,:));
>> axis equal, axis([-0.5 6.5 -1.5 5]), grid
>>xlabel('x-axis'),ylabel('y-axis');set(11,'Linewidth',2);
>>Hold on; plot(rxt(:,1),rxt(:,2),':')
>>for ii=1:10:length(qt) xx=kinJink(qt(ii,:),L); li=plot(xx(1r:),xx(2,:)); set(li,'Linewidth',1); end

the motion in the task plane can be plotted as shown in Fig. 8. The next commands plot the task and joint velocities versus time (Fig. 9):

>>subplot(211); plot(T,xdt);
>>xlabel(ltime');ylabel('task velocities')
>> subplot(212); plot(T,qdt);
>>xlabel('time');ylabel('joint velocities')

Actually, the user can use all the tools available in Matlab and Simulink for the control design or the analysis of simulation results.

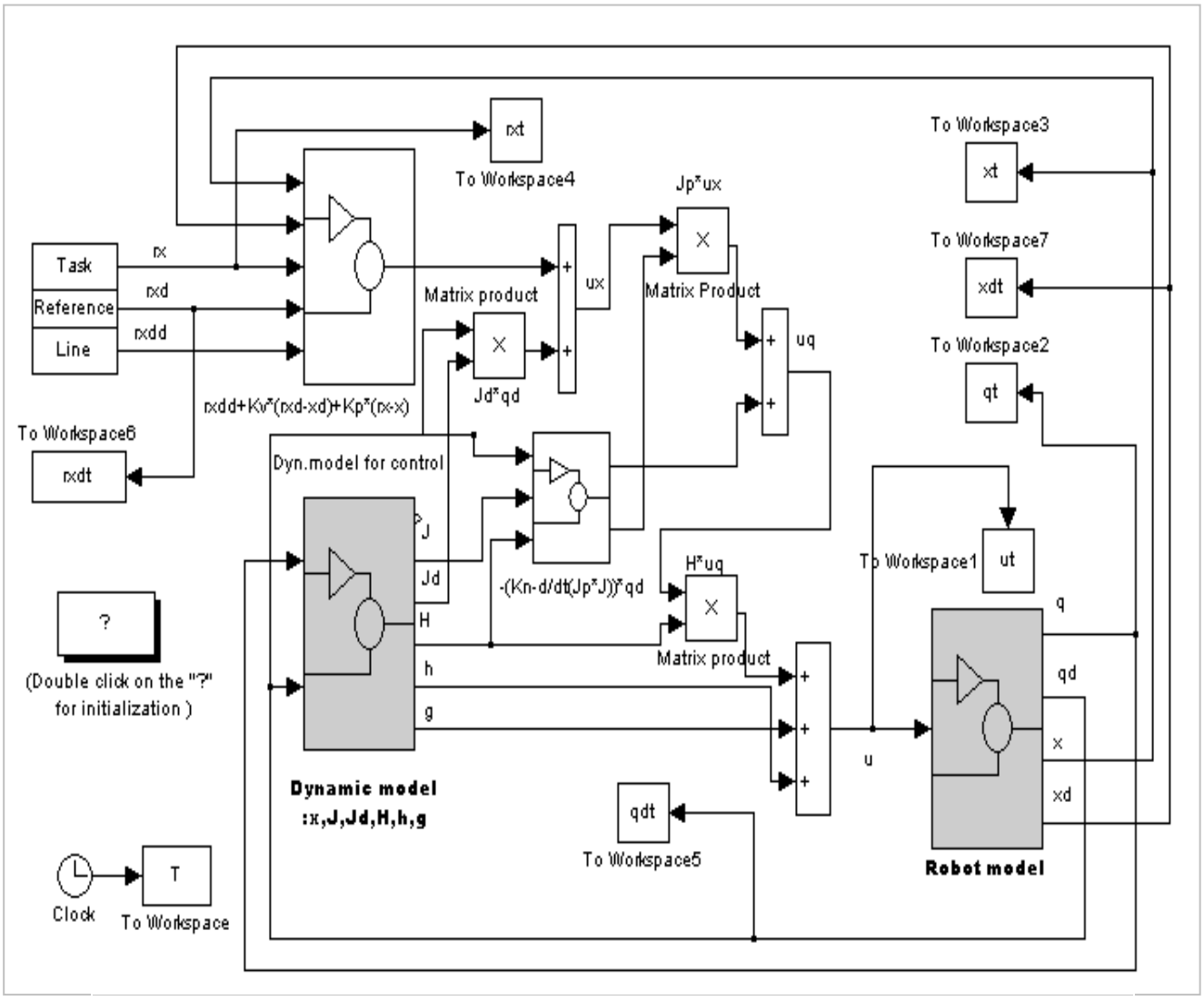

Fig. 7. SIMULINK block diagram for a robot system with the redundancy resolution at the acceleration level [control algorithm: E(25)].

4. Conclusions

In the paper, cinematic and dynamic models of *n*-DOF planar manipulators (robot) with revolving joints are presented. Numerical effective equations to solve the direct kinematics and dynamics are developed. Their basic benefit is to utilize partial results in further calculations. Additionally, some cost functions for *n*-R manipulators and their gradients are presented. Based on the derived models, a software package "Robotic Toolbox" for cinematic and dynamic simulation of *n*-R planar manipulators with revolute joints is derived.
The package is implemented in Matlab/Simulink. It consists of several M-files for the calculation of the model and other functions for planar manipulators which cannot be created using the standard ones.
To make the simulation easier, the modules were also integrated in Simulink environment as S-function blocks. Additional blocks for matrix operation in Simulink enable a straightforward building of the simulation model.
Current experience with the toolbox has confirmed that it is very useful and effective tool for many purposes: cinematic simulation, dynamic simulation, analysis and synthesis of control systems, trajectory generation, etc. As the complexity of the model increases slower with the number of DOF than in the case of general manipulators, the derived toolbox permits simulation of manipulators with many DOF within reasonable simulation times. It is very easy to extend and to adapt the simulation package to different requirements.
Thus, it should be of interest to the researchers involved in the development of advanced robot control systems. Our further work will consider the real-time simulation. We intend to rewrite all the functions into C-MEX files so that the automatic code generation will be possible.
The final goal is the development of an integrated environment for the robot control design and testing by using the "robot in the loop" simulation where the model of the robot in the simulation scheme is replaced by the real manipulator.

References:

[1] H. Asada H., Slotine J.J.E.., (1986) *Robot Analysis and Control*, Wiley, Chichester.
[2] Carignan C.R., (1991) *Trajectory optimization for cinematically redundant arms*, Robotic Syst. 4.
[3] Corke P.I., (1996) *A robotics toolbox for MATLAB*, IEEE Robotics Automn Mag. 3.
[4] Euler J.A., Dubey R.V., (1989) *A comparison of two real-time control scemes for redundant manipulators with bounded joint velocities*, Proceedings of the IEEE Conference on Robotics and Automation, Scottsdale.
[5] Ivănescu M., (1994) *Roboți industriali,* Editura Universitaria, Craiova.
[6] Hsia T.C., Guo Z.Y, (1989) *Joint trajectory generation forredundant robots*, Proceedings of the IEEE Conf Robotics and Automation, Scottsdale.
[7] Hsu P., Hauser J., Sastry S., (1989) *Dynamic control of redundant manipulators*, Robotic Syst. 6.
[8] Khatib O., (1987) *A unified approach for motion and force control of robot manipulators: the operational space formulation*, IEEE Trans. Robotics Automn 3.
[9] Klein C.A., Huang C.H., (1983) *Review of pseudo inverse control for use with cinematically redundant manipulators*, IEEE Trans. Systems, Man, Cybernetics SMC 13.
[10] Lenarcic J., (1993) *Optimum configurations of planar n-R hyper-redundant mechanism*, International Conference on Advanced Robotics, Tokyo.
[11] Martin D.P., Baillieul J., Hollerbach J.M., (1989) *Resolution of cinematic redundancy using optimization techniques*, IEEE Trans. Robotic Autom 5.
[12] Nenchev D.N., (1989) *Redundancy resolution through local optimization*, Robotic Syst. 6.
[13] Nethery J. E. Spong M.W., (1994) *Robotica: a Mathematica package for robot analysis*, IEEE Robotic Automn Mag. 1.
[14] Popescu M.C., (1996) *Simularea numerică a proceselor*, Tipografia Universității Contantin Brâncuşi, Târgu-Jiu.
[15] Popescu M.C., (2006) *2D Optimal Control Algorithms Implementation,* WSEAS Transactions on System and Control, Issue 1, Vol. 1, Veneția.
[16] Surdilovic D., Lizama E., Kirchhof J., (1995) *A toolbox for simulation of robotic systems*, E. Breitenecker, I. Husinsky (Eds.), EUROSIM '95 Simulation Congress, Vienna, Elsevier, Oxford.
[17] Zlajpah L., Lenarcic J., (1994) *On-line minimum joint torque motion generation for redundant manipulators*, 25th Interntional Symposium on Industrial Robots, Hanover.