Symbolic Algebra for Manipulator Dynamics

Peter I. Corke
CSIRO Division of Manufacturing Technology
Pinjarra Hills, AUSTRLIA. 4069
pic@brb.dmt.csiro.au

August 27, 1996

Abstract

This paper discusses the application of symbolic algebra software to the generation and analysis of manipulator dynamic equations of motion. For robots with more than two degrees of freedom, manual manipulation of these equations is impractical.

Using a popular computer algebra package and a number of simple programs we can readily gain insight into the configuration dependent dynamic terms, and their effect on independent joint axis control strategies. We can also readily determine the manipulator's base parameter set, generate a form of the dynamic equations suitable for linear parameter identification, and generate efficient run-time code for computed-torque or feed-forward control.

1 Introduction

The dynamic characteristics of robot manipulators are significant in determining limits to performance, and also in control design. While the underlying theory and algorithmic techniques are relatively old, it remains difficult to gain insight into the effects and their significance, particularly for manipulators more complex than the two-link examples in textbooks. However such insights are important in the analysis of new high-performance robot designs, as well as the control of existing manipulators. Computer algebra packages allow the individual to generate and manipulate the dynamic equations of motions, and investigate a large number of issues related to manipulator dynamics.

The use of computer packages for the manipulation of robot dynamics equations goes back to early work such as ARM[1], DYMIR[2] and EMDEG[3]. These were generally large custom LISP or Fortran based packages. Since the mid 1980s the increasing power and availability of general purpose computer algebra tools such as Macsyma, REDUCE and MAPLE has reduced the need for specialized packages for manipulator dynamics.

Using a general purpose computer algebra package and a high-performance workstation, it is now quite possible to compute and manipulate the complex torque expressions in symbolic form with relatively simple symbolic manipulation programs.

This paper describes in tutorial form the use of a computer algebra package to derive and manipulate the dynamic equations of motion for a robot manipulator. The discussion is based on the commonly available computer algebra package, MAPLE[4]. MAPLE is widely available on many platforms, and is found at Universities and research institutions worldwide. The examples given are for the Puma 560 manipulator, the dynamics of which have been studied extensively allowing ready cross-checking of the results presented. All work has been conducted using MAPLE V.3 on a SparcStation 5.

A number of simple programs and scripts have been written to facilitate generation and manipulation of the dynamic equations. The manipulations include the visualization of configuration dependance of dynamic effects, various simplifications, intertial parameter identification and base parameter set determination, as well as automatic code generation for efficient online computed-torque or feedforward control. Once only offline symbolic manipulation of the dynamics for a particular manipulator can greatly reduce the run-time load, to something readily dealt with by a modest microprocessor.

Section 2 discusses the generation of the symbolic equations of motion, and issues such as simplification and verification. Section 3 describes a number of useful operations that can be performed on the symbolic equations. In Section 4 we substitute numerical values for inertial and kinematic parameters and examine the configuration dependance of terms in the equations of motion, significance of terms, and methods of culling less significant terms. Section 5 investigates issues concerned with inertial parameter identification, reformulation of the dynamics in regressor form, and determining the manipulator's base parameter set. Section 6 describes the automatic generation of run-time code for computed-torque or feedforward control. Appendix A summarizes the various scripts provided in the package, and Appendix B shown an example MAPLE session.

Method	Multiplications	Additions	For N=6	
			Mul	Add
Lagrangian[10]	$32\frac{1}{2}n^4 + 86\frac{5}{12}n^3$ $+171\frac{1}{4}n^2 + 53\frac{1}{3}n$ -128	$25n^4 + 66\frac{1}{3}n^3$ $+129\frac{1}{2}n^2 + 42\frac{1}{3}n$ -96	66,271	51,548
Recursive NE[10]	$150n - 48$	$131n - 48$	852	738
Kane[7]			646	394
Simplified RNE[1]			224	174

Table 1: Comparison of computational costs for inverse dynamics from various sources. The last entry is achieved by symbolic simplification using the software package ARM. The two rightmost columns are for a 6-axis Puma 560 manipulator.

2 Dynamic equations of motion

The inverse dynamics give the joint torques required for a given set of joint angles, velocities and accelerations. These equations of motion are given by

$$\underline{Q} = \mathbf{M}(\underline{q})\ddot{\underline{q}} + \mathbf{C}(\underline{q}, \dot{\underline{q}})\dot{\underline{q}} + \mathbf{G}(\underline{q}) \tag{1}$$

where

n	is the number of axes,
q_i	is an n-vector of generalized coordinates (θ_i for a revolute joint and d_i for a prismatic joint),
$\mathbf{M}$	is the positive-definite joint-space inertia matrix,
$\mathbf{C}$	represents coriolis and centripetal forces,
$\mathbf{G}$	is the gravity loading, and
$\underline{Q}$	is the vector of generalized forces applied to the joints.

The equations may be derived via a number of techniques, including Lagrangian (energy based), Newton-Euler (NE), d'Alembert[5, 6] or Kane's[7] method. The earliest reported work was by Uicker[8] and Kahn[9] using the Lagrangian approach, which has a computational complexity of $O(n^4)$.

The NE and Lagrange forms can be written generally in terms of the Denavit-Hartenberg parameters — however the specific formulations, such as Kane's, can have lower computational cost for the specific manipulator. Whilst the recursive forms are computationally more efficient, the non-recursive forms compute the individual dynamic terms ($\mathbf{M}$, $\mathbf{C}$ and $\mathbf{G}$) directly. A comparison of computation costs is given in Table 1.

2.1 Recursive Newton-Euler equations

In 1980 Luh et al.[11] provided a recursive formulation of the Newton-Euler equations with linear and angular velocities referred to link coordinate frames, resulting in $O(n)$ complexity.

The outward recursion propagates kinematic information – such as angular velocities, angular accelerations, linear accelerations, total forces and moments exerted at the center of mass of each link – from the base reference (inertial) frame to the end-effector. The inward recursion propagates the forces and moments exerted on each link from the end-effector of the manipulator to the base reference frame.

Outward recursion, $1 \leq i \leq n$**.**

If axis $i+1$ is rotational

$$\begin{aligned}
{}^{i+1}\underline{\omega}_{i+1} &= {}^{i+1}\mathbf{R}_i \left({}^{i}\underline{\omega}_i + \underline{z}_0 \underline{\dot{q}}_{i+1} \right) && (2)\\
{}^{i+1}\underline{\dot{\omega}}_{i+1} &= {}^{i+1}\mathbf{R}_i \left\{ {}^{i}\underline{\dot{\omega}}_i + \underline{z}_0 \underline{\ddot{q}}_{i+1} + {}^{i}\underline{\omega}_i \times \left(\underline{z}_0 \underline{\dot{q}}_{i+1} \right) \right\} && (3)\\
{}^{i+1}\underline{v}_{i+1} &= {}^{i+1}\underline{\omega}_{i+1} \times {}^{i+1}\underline{p}^*_{i+1} + {}^{i+1}\mathbf{R}_i {}^{i}\underline{v}_i && (4)\\
{}^{i+1}\underline{\dot{v}}_{i+1} &= {}^{i+1}\underline{\dot{\omega}}_{i+1} \times {}^{i+1}\underline{p}^*_{i+1} + {}^{i+1}\underline{\omega}_{i+1} \times \left\{ {}^{i+1}\underline{\omega}_{i+1} \times {}^{i+1}\underline{p}^*_{i+1} \right\} \\
&\quad + {}^{i+1}\mathbf{R}_i {}^{i}\underline{\dot{v}}_i && (5)
\end{aligned}$$

If axis $i+1$ is translational

$$\begin{aligned}
{}^{i+1}\underline{\omega}_{i+1} &= {}^{i+1}\mathbf{R}_i {}^{i}\underline{\omega}_i && (6)\\
{}^{i+1}\underline{\dot{\omega}}_{i+1} &= {}^{i+1}\mathbf{R}_i {}^{i}\underline{\dot{\omega}}_i && (7)\\
{}^{i+1}\underline{v}_{i+1} &= {}^{i+1}\mathbf{R}_i \left(\underline{z}_0 \underline{\dot{q}}_{i+1} + {}^{i}\underline{v}_i \right) + {}^{i+1}\underline{\omega}_{i+1} \times {}^{i+1}\underline{p}^*_{i+1} && (8)\\
{}^{i+1}\underline{\dot{v}}_{i+1} &= {}^{i+1}\mathbf{R}_i \left(\underline{z}_0 \underline{\ddot{q}}_{i+1} + {}^{i}\underline{\dot{v}}_i \right) + {}^{i+1}\underline{\dot{\omega}}_{i+1} \times {}^{i+1}\underline{p}^*_{i+1} \\
&\quad + 2\, {}^{i+1}\underline{\omega}_{i+1} \times \left({}^{i+1}\mathbf{R}_i \underline{z}_0 \underline{\dot{q}}_{i+1} \right) \\
&\quad + {}^{i+1}\underline{\omega}_{i+1} \times \left({}^{i+1}\underline{\omega}_{i+1} \times {}^{i+1}\underline{p}^*_{i+1} \right) && (9)
\end{aligned}$$

$$\begin{aligned}
{}^{i}\dot{\overline{\underline{v}}}_i &= {}^{i}\underline{\dot{\omega}}_i \times \underline{s}_i + {}^{i}\underline{\omega}_i \times \left\{ {}^{i}\underline{\omega}_i \times \underline{s}_i \right\} + {}^{i}\underline{\dot{v}}_i && (10)\\
{}^{i}\underline{F}_i &= m_i {}^{i}\dot{\overline{\underline{v}}}_i && (11)\\
{}^{i}\underline{N}_i &= \mathbf{J}_i {}^{i}\underline{\dot{\omega}}_i + {}^{i}\underline{\omega}_i \times \left(\mathbf{J}_i {}^{i}\underline{\omega}_i \right) && (12)
\end{aligned}$$

Inward recursion, $n \geq i \geq 1$**.**

$$
\begin{aligned}
{}^{i}\underline{f}_i &= {}^{i}\mathbf{R}_{i+1}\,{}^{i+1}\underline{f}_{i+1} + {}^{i}\underline{F}_i && (13)\\
{}^{i}\underline{n}_i &= {}^{i}\mathbf{R}_{i+1}\left\{{}^{i+1}\underline{n}_{i+1} + \left({}^{i+1}\mathbf{R}_i\,{}^{i}\underline{p}_i^*\right) \times {}^{i+1}\underline{f}_{i+1}\right\} \\
&\quad + \left({}^{i}\underline{p}_i^* + \underline{s}_i\right) \times {}^{i}\underline{F}_i + {}^{i}\underline{N}_i && (14)\\
\underline{Q}_i &= \begin{cases} \left({}^{i}\underline{n}_i\right)^T \left({}^{i}\mathbf{R}_{i+1}\underline{z}_0\right) & \text{if link } i+1 \text{ is rotational} \\ \left({}^{i}\underline{f}_i\right)^T \left({}^{i}\mathbf{R}_{i+1}\underline{z}_0\right) & \text{if link } i+1 \text{ is translational} \end{cases} && (15)
\end{aligned}
$$

where

i	is the link index, in the range 1 to n
$\mathbf{J}_i$	is the moment of inertia of link i about its COM
$\underline{s}_i$	is the position vector of the COM of link i with respect to frame i
$\underline{\omega}_i$	is the angular velocity of link i
$\underline{\dot{\omega}}_i$	is the angular acceleration of link i
$\underline{v}_i$	is the linear velocity of frame i
$\underline{\dot{v}}_i$	is the linear acceleration of frame i
$\underline{\overline{v}}_i$	is the linear velocity of the COM of link i
$\underline{\dot{\overline{v}}}_i$	is the linear acceleration of the COM of link i
$\underline{n}_i$	is the moment exerted on link i by link $i-1$
$\underline{f}_i$	is the force exerted on link i by link $i-1$
$\underline{N}_i$	is the total moment at the COM of link i
$\underline{F}_i$	is the total force at the COM of link i
$\underline{Q}_i$	is the force or torque exerted by the actuator at joint i
${}^{i-1}\mathbf{R}_i$	is the orthonormal rotation matrix defining frame i orientation with respect to frame $i-1$. It is the upper 3×3 portion of the link transform matrix given in (**??**).

$$
\begin{aligned}
{}^{i-1}\mathbf{R}_i &= \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i \end{bmatrix} && (16)\\
{}^{i}\mathbf{R}_{i-1} &= \left({}^{i-1}\mathbf{R}_i\right)^{-1} = \left({}^{i-1}\mathbf{R}_i\right)^T && (17)
\end{aligned}
$$

${}^{i}\underline{p}_i^*$	is the displacement from the origin of frame $i-1$ to frame i with respect to frame i.

$$
{}^{i}\underline{p}_i^* = \begin{bmatrix} a_i \\ d_i \sin\alpha_i \\ d_i \cos\alpha_i \end{bmatrix} \qquad (18)
$$

It is the negative translational part of $\left({}^{i-1}\mathbf{A}_i\right)^{-1}$.

Axis	RNE	Simple	Numeric substit.
1	25 728	16 222	1 145
2	11 046	6 738	488
3	8 919	5 421	348
4	3 552	2 329	81
5	2 421	1 674	90
6	1 331	982	18
	52 997	33 366	2 170

Table 2: Number of terms in torque expres sions after RNE computation, trigonometric simplification, and numeric parameter substitution.

$\underline{z}_0$ is a unit vector in Z direction, $\underline{z}_0 = [0\ 0\ 1]$

Note that the COM linear velocity given by equation (4) or (8) does not need to be computed since no other expression depends upon it. Boundary conditions are used to introduce the effect of gravity by setting the acceleration of the base link

$$\dot{v}_0 = -\underline{g} \tag{19}$$

where $\underline{g}$ is the gravity vector in the reference coordinate frame, generally acting in the negative Z direction, downward. Base velocity is generally zero

$$\begin{aligned} v_0 &= 0 && (20) \\ \omega_0 &= 0 && (21) \\ \dot{\omega}_0 &= 0 && (22) \end{aligned}$$

2.2 Kinematic conventions

Unfortunately there are two methods for describing manipulator kinematics, and the affixment of coordinate frames to manipulator links. These are the so called standard Denavit-Hartenberg notation of Paul[12], and the modified Denavit-Hartenberg form of Craig[13]. The advantage of one form over the other is a subject for debate, but what is important is for authors of papers to clearly state the kinematic conventions used, and for readers to be aware that there can be significant variation in kinematic conventions adopted, even for the same robot. Each method has a different form for the link transform matrices, $\mathbf{A_i}$, and the RNE algorithm — the version given above is for the standard Denavit-Hartenberg notation, as does the remainder of this paper.

Method	Time (s)	Memory (Mbyte)
RNE 3 axes	5.1	1.0
RNE 6 axes	530	23
Lagrangian 3 axes 51	2.4	

Table 3: Execution time for symbolic computation of equations of motion using Lagrange and RNE methods. Note that the 6-axis Lagrangian calculation caused page thrashing (despite 64Mbyte RAM) and did not complete.

3 A MAPLE implementation

The RNE algorithm is straightforward to program and efficient to execute in the general case, but considerable savings can be made for the specific manipulator case. The general form inevitably involves many additions with zero and multiplications with 0, 1 or -1, in the various matrix and vector operations. The zeros and ones are due to the trigonometric terms[1] in the orthonormal link transform matrices (16) as well as zero-valued kinematic and inertial parameters. Symbolic simplification gathers common factors and eliminates operations with zero, reducing the run-time computational load, at the expense of a once-only off-line symbolic computation. Symbolic manipulation can also be used to gain insight into the effects and magnitudes of various components of the equations of motion.

In this work a general purpose symbolic algebra package, MAPLE[14], has been used to compute the torque expressions in symbolic form via a straightforward implementation of the RNE algorithm. Compared to symbolic computation using the Lagrangian approach, computation of the torque expressions is very efficient, see Table 3.

These expressions are in sum of product form, and can be extremely long. For example the expression for the torque on the first axis of a Puma 560 is

$$\begin{aligned}
\tau_1 \quad = \quad & -C_{23}\, I_{yz3}\, \ddot{q}_2 \\
& + {s_{x1}}^2 m_1\, \ddot{q}_1 \\
& + S_{23}\, I_{yz3}\, \dot{q}_3^2 \\
& + I_{yy2}\, {C_2}^2 \ddot{q}_1 \\
& + C_2\, I_{yz2}\, \ddot{q}_2 \\
& + \cdots
\end{aligned}$$

and continues on for over 16,000 terms. By multiplying together various symbolic expressions the inherent factorization is lost, but this form is nevertheless very useful for the analyses described in the next section.

[1] Common manipulators have link twists of 0°, 90° or −90° leading to zero or unity trigonometric results.

RNE variable	MAPLE variable
θ_i	`q.i`
$\dot{\theta}_i$	`qd.i`
$\ddot{\theta}_i$	`qdd.i`
Q_i	`tau.i`
R_i	`R.i`
p_i^*	`p.i`
s_i	`s.i`
ω_i	`w.i`
$\dot{\omega}_i$	`wd.i`
$\dot{v}_i$	`acc.i`
f_i	`f.i`
F_i	`F.i`
n_i	`n.i`
N_i	`N.i`
$_i$	`acc.i`
$\mathbf{J}_i$	`II.i`

Table 4: Newton-Euler variables and corresponding program variable names.

3.1 Implementation details

MAPLE allows arrays and vectors, and can perform arithmetic on these data types. However for readibility, and convenience in manipulating the resulting equations many vector variables are represented in the programs by scalars named with the index as a suffix. For example, the joint angles are represented by `q1`, `q2`, `q3` etc. Such variable names are easily created in MAPLE using the concatention, or dot, operator `q.i` where `i` is a variable evaluating to an integer. Table 4 shows the relationship between the manipulator states and parameters, and the program variables.

The programs are implemented as a suite of short script files which may be invoked by MAPLE's `read` command. The script files read and return global variables which can be displayed interactively or further manipulated. For these reasons script files rather than functions were implemented, since they are more convenient for interactive work.

Two versions of the RNE program have been written to support the different kinematic conventions discussed in Section 2.2 The script `rne` uses the standard Denavit-Hartenberg notation as given by Paul[12], and `mrne` using modified Denavit-Hartenberg form as per Craig[13].

Firstly the kinematic parameters need to be established. A script is provided for the Unimate Puma 560 robot case

```
> read `dh-puma560`;
```

which makes the definitions

```
Naxis := 6:

A :=     [ 0,      A2,  A3,    0,    0,     0]:  # Ai
D :=     [ 0,      0,   D3,    D4,   0,     0]:  # Di
alpha :=[ Pi/2,  0,  -Pi/2,  Pi/2, -Pi/2,  0]:  # Alpha_i
```

The equations of motion can now be computed by

```
> read rne;
```

which results in a set of torque expressions, `tau1, tau2 ...` that are in sum of product form

$$\tau_i = \alpha_{i1}\Theta_{i1} + \alpha_{i2}\Theta_{i2} + \ldots + \alpha_{in}\Theta_{in} \tag{23}$$

where α_{ij} is a numeric coefficient, and Θ_{ij} is a polynomial in manipulator state variables and trigonometric expressions. Typically there are many thousands of terms in each expression, as shown in Table 2. The issues of simplification and factorization will be discussed next.

3.2 Trigonometric representation

MAPLE provides the trigonometric functions required such as sine and cosine. However it has been found that manipulation of expressions containing these functions is slow, and consumes substantial memory. For these reasons it has been found necessary to use symbolic names rather than the inbuilt trigonometric functions. The names chosen are similar to the conventional robotic shorthand, where $S_i = \sin q_i$ and $C_i = \cos q_i$. In MAPLE, using the dot operator, we use `S.i` for $\sin q_i$ and `C.i` for $\cos q_i$.

The native trigonometric functions can be substituted for the shorthand by the script `tribsub` which is essentially

```
for i from 1 to Naxis do
    S.i := sin(q.i):
    C.i := cos(q.i):
od:
```

3.3 Trigonometric simplification

The torque expressions from the symbolic RNE are not in their simplest possible form, but since we use the standard robotics shorthand notation to represent trigonometric terms, MAPLE's inbuilt trigonometric simplification rules cannot work. As mentioned those inbuilt rules are very general and thus slow. However they also make undesirable trigonometric substitutions for example creating terms such as $\sin(2\theta_2+\theta_3)$ which has little physical significance for the Puma 560. We implement our own limited simplification rules recognizing the identities:

$$
\begin{aligned}
S_i^2 + C_i^2 &= 1 && (24)\\
S_iC_j \pm C_iS_j &= S_{i\pm j} && (25)\\
S_iS_j \pm C_iC_j &= C_{i\mp j} && (26)
\end{aligned}
$$

An example of the simplification procedure will be given for the identity (24). Consider the torque expression;

$$\tau = AS_6^2 + BS_6^2 + CS_6^2 + DS_6^2 + AC_6^2 + CC_6^2 + EC_6^2$$

The simplification algorithm is:

1. Find the coefficients of S_i^2 and C_i^2;

```
s_coef := coeff(tau,S6^2);
 A + B + C + D
c_coef := coeff(tau,C6^2);
 A + C + E
```

2. Convert the coefficients to sets;

```
s_set := convert(s_coef, set);
 {D, A, B, C}
c_set := convert(c_coef, set);
 {E, A, C}
```

   Note that the ordering of elements within a MAPLE set is of no significance.

3. The intersection of the two sets, is a set of all terms which were multiplied by both C_i^2 and S_i^2 in the original equation

```
inter_set := s_set intersect c_set;
 {A, C}
```

4. The intersection set is converted back to an expression,

```
inter_expr := convert(inter_set, `+`);
 A + C
```

5. A zero valued correction term `1 - (S6^2 + C6^2)` is added to the original expression

```
tau := tau + inter_expr * (1 - (S6^2 + C6^2));
```

to effect the substitution Resulting in

$$\tau = BS_6^2 + DS_6^2 + EC_6^2$$

The other identities (25) and (26) may be implemented in a similar manner, making extensive use of the `coeff()` function and set manipulation. These functions are defined in the script file `trig_ident` as the functions `trig_sumsq()` and `trig_sumang()`. For the case of the Puma 560, apriori knowledge that $\theta_2 + \theta_3$ is a useful sum of angles substitution allows a more efficient simplification pass to be implemented, and this is implemented by the function `trig_sumang23()`. A more general solution would be to apply the sum of angle simplification only to those axes which are parallel, that is joints i and $i+1$ where $\alpha_i = 0$.

The script `simple` implements the specific simplifications for a Puma robot on all joints. For example

```
> read simple560;
```

3.4 Computational performance issues

Table **??** compares the CPU and memory usage for computation and simplification of the torque expressions. It can be seen that using MAPLE's native trigonometric representation is considerably more costly.

3.5 Verification of symbolic results

As discussed previously, the dynamic equations are extremely complex, and thus difficult to verify. This section discusses some of the approaches taken to verify the accuracy of the symbolic RNE and simplification code.

The dynamics of the simple two-link example given by Fu et al.[5] were computed and simplified. The computer algebra results agreed with those originally given. However for increased number of axes it is not practical to check the results against manual calculation.

For the Puma 560 the gravity term is readily extracted, section 4, and is simple enough to verify manually. The manipulator inertia matrix is positive definite, a characteristic that can also be checked using MAPLE.

Numerical comparisons were made against a numerical implementation of the recursive Newton-Euler formulation[15] and the results of a LISP code generator package[16]. For a number of manipulator poses, the numeric values for $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{G}$ were identical.

A colleague independantly implemented the dynamic equations in MAPLE, but used the Lagrangian formulation rather than RNE. Using MAPLE we were then able to confirm that the sets of torque expressions were identical. Silver[17] has previously shown the equivalence of the Lagrangian and recursive Newton-Euler forms.

4 Further symbolic manipulation of the equations of motion

The torque expressions by themselves are only of limited value due to their enormous complexity. Once again we can use the symbolic manipulation software to help us analyze and interpret the equations.

4.1 Extraction of inertial, centrifgual, Coriolis and gravity terms

The symbolic elements of the **M**, **C** and **G** terms can be readily extracted from the sum of product form, This overcomes what is frequently cited as an advantage of the Lagrangian formulation, in that the individual terms are computed directly. Equation (1) may be written in tensor form as

$$Q_i = M_{ij}\ddot{q}_j + C_{ijk}\dot{q}_j\dot{q}_k + G_i$$

We use our knowledge of this equation to isolate the tensor elements from the torque expressions given by (1). M_{ij} is the coefficient of $\ddot{q}_j$ in τ_i, C_{ijk} the coefficient of $\dot{q}_j\dot{q}_k$ in τ_i, and G_i is the coefficient of g in τ_i. MAPLE's `coeff()` function computes the coefficient of a specified indeterminate within an expression. The elements of the inertia tensor are

```
for i from 1 to Naxis do
     for j from i to Naxis do
            M.i.j := coeff(tau.i, qdd.j);
            M.j.i := M.i.j:
     od;
od;
```

Note that we use our knowledge of the symmetric nature of the inertia matrix to reduce the number of calls to the `coeff()` function. Similary we can find the centripal and Coriolis tensor

```
for i from 1 to Naxis do
     for k from 1 to Naxis do
          for m from k to Naxis do
               C.i.k.m := coeff( coeff(tau.i, qd.k), qd.m);
          od;
     od;
od;
```

Finally, gravity terms are those which have the symbol `grav`, g, as a product.

```
for i from 1 to Naxis do
    G.i := coeff(tau.i, grav) * grav;
od;
```

For the Puma the gravity load is

$$
\begin{aligned}
\frac{\tau_{g3}}{g} &= -\left(\left(m6 + m5 + m4\right) D4 + sz3\; m3 + m4\; sy4\right) S23 \\
&\quad + \left(\left(m3 + m4 + m5 + m6\right) A3 + m3\; sx3\right) C23 \\
&\quad + \left(sz4\; m4 - sy5\; m5 - sy6\; m6\; C6\right) C23\; S4 \\
&\quad + S23\; S5\; sy6\; m6\; S6 \\
&\quad - \left(sz6\; m6 + sz5\; m5\right)\left(C23\; S5\; C4 + S23\; C5\right) \\
&\quad - S6\; sy6\; m6\; C23\; C4\; C5 \\
\frac{\tau_{g2}}{g} &= \left(m2\; sx2 + \left(m2 + m3 + m4 + m5 + m6\right) A2\right) C2 \\
&\quad - sy2\; m2\; S2 + \frac{\tau_{g3}}{g} \qquad (27)
\end{aligned}
$$

where A_i and D_i are kinematic constants, $C_i \equiv \cos\theta_i$ and $S_i \equiv \sin\theta_i$.

4.2 Other applications

MAPLE is capable of generating LaTeX style equations as well as 'C' language arithmetic statements corresponding to any expression. Example applications include preparation of documents, as well as automatically generating code for computed torque control of a specific manipulator.

5 Substitution of numeric values

The script `params-puma560` assigns numeric values appropriate to a Puma560 manipulator to all kinematic and inertial parameters. This data is as given by Corke and Armstrong[18, 19].

```
> read `params-puma560`;
```

Since many of the numeric parameters are zero, for instance the products of inertia for the links, the torque expressions are considerably simplified, and is reflected in Table 2 as a reduced number of terms.

5.1 Significance of terms

The terms in the expanded sum of product form (23) are not of equal significance and this is reflected in the magnitude of the coefficient α_{ij}. For example, the acceleration of joint 6 exerts a very small torque on joint 2 compared to the gravity load on that axis. The significance of a term is related to the magnitude of its coefficient, α_{ij}, and Figure **??** shows a histogram of the distribution of these coefficient magnitudes for the Puma 560's joint 1. The median coefficient magnitude is nearly four orders of magnitude below that of the greatest coefficient. In all examples the inertia contribution of the motor's armature is included

Ranking	Axis		
	1	2	3
1	$-3.207\, C_2\, S_2\, \dot{q}_1\, \dot{q}_2$	$37.28\, C_2$	$-8.786\, S_{23}$
2	$1.734\, {C_2}^2\, \ddot{q}_1$	$-8.786\, S_{23}$	$1.208\, C_{23}$
3	$1.547\, C_2\, S_2\, S_3\, \dot{q}_1\, \dot{q}_2$	$4.435\, \ddot{q}_2$	$0.938\, \ddot{q}_3$
4	$0.968\, \ddot{q}_1$	$1.604\, \dot{q}_1^2\, C_2\, S_2$	$-0.387\, S_3\, \ddot{q}_2$
5	$-0.773\, {C_2}^2\, C_3\, \dot{q}_1\, \dot{q}_3$	$1.208\, C_{23}$	$0.387\, \dot{q}_2^2\, C_3$

Ranking	Axis		
	4	5	6
1	$0.192\, \ddot{q}_4$	$0.171\, \ddot{q}_5$	$0.194\, \ddot{q}_6$
2	$0.0283\, S_{23}\, S_4\, S_5$	$-0.0283\, C_{23}\, S_5$	$-0.00004\, C_4\, S_{23}\, S_5\, \ddot{q}_1$
3	$0.00249\, C_{23}\, C_4\, S_5\, \dot{q}_1\, \dot{q}_3$	$-0.0283\, C_4\, C_5\, S_{23}$	$0.00004\, C_{23}\, C_5\, \ddot{q}_1$
4	$0.00249\, C_{23}\, C_4\, S_5\, \dot{q}_1\, \dot{q}_2$	$0.00249\, C_{23}\, C_5\, S_4\, \dot{q}_1\, \dot{q}_3$	$-0.00004\, C_4\, C_5\, S_{23}\, \dot{q}_1\, \dot{q}_5$
5	$0.00249\, C_4\, S_2\, S_5\, \dot{q}_1\, \dot{q}_2$	$0.00249\, C_{23}\, C_5\, S_4\, \dot{q}_1\, \dot{q}_2$	$-0.00004\, C_5\, S_{23}\, \dot{q}_1\, \dot{q}_3$

```
> read addmotor;
```

Corke[20] describes various approaches to the task of ranking the terms. The simplest is to rank based only on coefficient magnitude

```
> read top5simple;
```

and the results are shown in Table 5.1, but such a simple measure is somewhat limited. For example, velocity terms contain either a velocity squared or a product of velocities and so may be significant despite a small coefficient. Joint accelerations and velocities may also differ significantly in magnitude — for the Puma 560 peak joint acceleration is approximately ten times peak velocity. Thus it may be appropriate to substitute also for 'typical' values of joint velocity and acceleration, or to evaluate the contribution of the terms over a wide range of joint velocities and accelerations. The scripts `top5nominal` and `top5random` implement the other ranking strategies described in [20] but are increasingly expensive to execute.

The culling process involves eliminating all terms j for which $|\alpha_{ij}| < |max(\alpha_i)|$. With MAPLE it is simple to cull all terms whose coefficient is less than 1% of the maximum by

```
for i from 1 to Naxis do
    alpha_max := maxnorm(theta.i):
    tau.i := map(   proc(a, am)
                        if abs(coeffs(a)) < am then
                                0
                        else
                                a
                        fi
                    end,
```

```
                        tau.i,
                        alpha_max/100
                );
        od:
```

`maxnorm()` returns the largest coefficient of a polynomial. `map()` is analagous to LISP's `mapcar`, and like a LISP lambda expression a function definition may be passsed as an argument to a function.

After culling all terms below 1% significance, the number of terms remaining are given in Table 2 — a substantial reduction has been achieved. Armstrong et al.[21] describe a manual approach to this task which took around 3 weeks to complete. Corke[20] uses a Monte-Carlo simulation to investigate the errors introduced by such culling procedures.

5.2 Visualizing parameter dependancies

The expressions for components of the manipulator torque may be plotted as functions of the state variables to gain insight into the dynamics.

There are two components of inertia 'seen' by the motor. One is due to the rotating armature, and the other due to the rigid-body dynamics of the link reflected through the gear system. The total inertia sets the upper bound on acceleration, and also affects the dynamics of the axis control loop. It is insightful to plot total inertia normalized with respect to the armature inertia, J_m, since this clearly shows the inertia compared with the unloaded ($\eta = 1$) case. The normalized inertia is defined to be

$$\eta_{ii}(\underline{q}) = 1 + \frac{\mathbf{M}_{ii}(\underline{q})}{G_i^2 J_{m_i}} \tag{28}$$

where $\mathbf{M}$ is the manipulator inertia matrix from (1), and J_{m_i} and G_i are the motor armature inertia and the reduction gear ratio respectively for joint i. The normalized inertial coupling is defined to be

$$\eta_{ij}(\underline{q}) = \frac{\mathbf{M}_{ij}(\underline{q})}{G_i^2 J_{m_i}} \tag{29}$$

The normalized diagonal and off-diagonal inertia elements for Puma joints 1 and 2 are shown in Figures 1 and 2 as a function of configuration. The off-diagonal terms are relatively small in magnitude compared to the diagonal values. Change in diagonal link inertia with configuration, as shown in Figure 1, has a significant effect on the dynamics of the axis control loop. The mechanical pole of the motor and link is

$$p_m = -\frac{B_m}{\eta(\underline{q}) J_m} \tag{30}$$

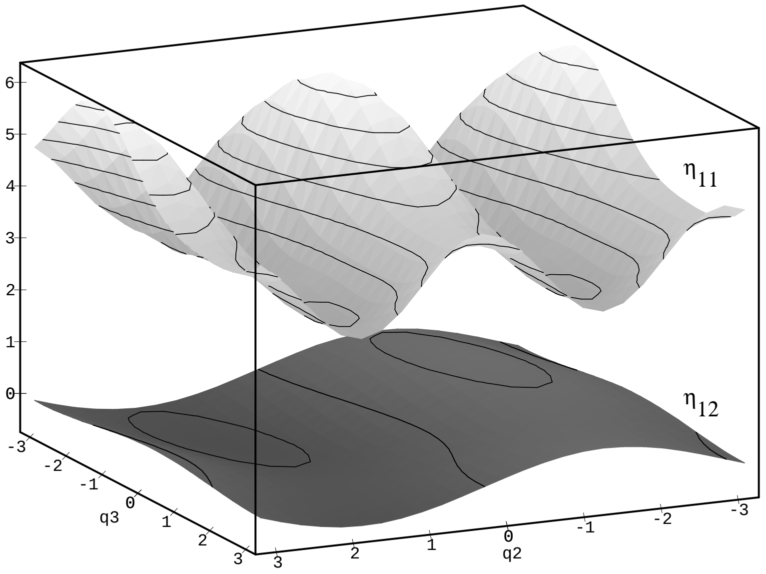

Figure 1: Plot of normalized inertia for joint 1 as a function of shoulder and elbow configuration. Shown is the diagonal term η_{11} and the inertial coupling term η_{12}. Angles in rads.

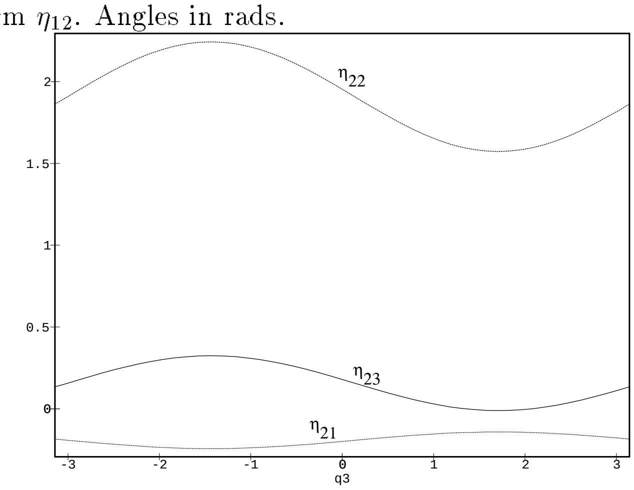

Figure 2: Plot of normalized inertia for joint 2 as a function of elbow configuration. Shown is the diagonal term η_{22} and the inertial coupling terms η_{21} and η_{23}. Angles in rads.

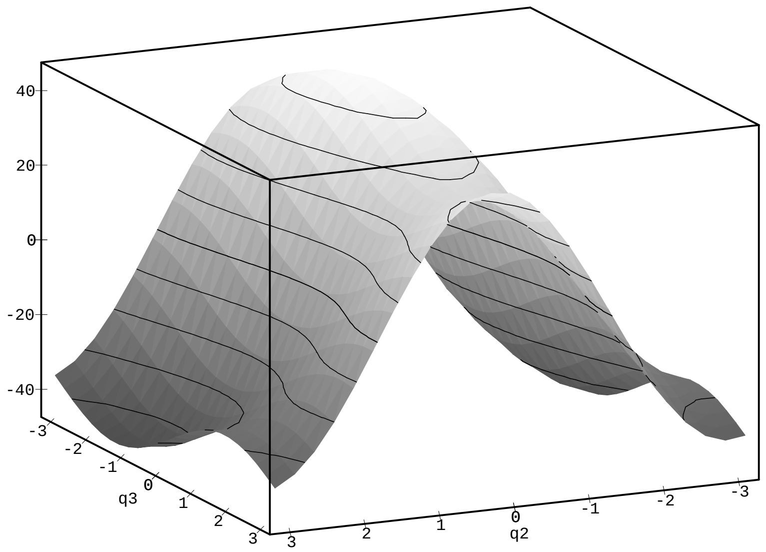

Figure 3: Gravity load (N.m) on joint 2, $\tau_{g_2}(q_2, q_3)$. Angles in rads. Gravity load varies between ± 46 N.m or $\pm 48\,\%$ of the fuse limited torque.

The variation of the mechanical pole, due to configuration change, represents a significant challenge for control design if it is to achieve stability and performance over the entire workspace. It is clear from (28) that without gearing this effect would be far more significant, making independent joint control generally infeasible for direct-drive robots.

Gravity load, plotted in Figure 3 for joint 2, shows that gravity torque is significant compared to the torque limit of the actuator.

6 Regressor form

Accurate dynamic control of a manipulator requires accurate knowledge of its dynamic parameters. A manipulator has ten inertial parameters per link; link mass, m_i, three first moments (or link center of gravity), and three moments, and three products, of inertia, and these parameters are difficult to determine. Corke and Armstrong-Hélouvry[22] have shown the considerable variation in previously published data sets for the Puma 560 robot. An alternative approach is to identify the inertial parameters for a particular robot using measured input/output data [23, 24, 25, 26, 27, 28].

If the equations of motion are linear in the inertial parameters then the parameters can be identified using least squares techniques based on measured motor torque, position, velocity and acceleration. In fact the torque expressions resulting from the Newton-Euler procedure, (23), are non-linear in the inertial parameters s_x, s_y and s_z, and the kinematic parameters a_i, $d_i \sin\alpha_i$ and $d_i \cos\alpha_i$[29]. However since the kinematic parameters are known apriori this latter non-linearity presents no difficulty. The Lagrange formulation, which utilizes inertia about the link coordinate frame, has been shown to be linear in the dynamic parameters[29]. If the NE model is re-formulated such that the inertia tensors are expressed in link coordinate frames rather than link center-of-mass coordinate frames, the modified NE formulation will be linear in the center of mass vector $\underline{s}_i$.

If $\mathbf{J}_i$ is the classical link inertia about the center-of-mass of link i, the corresponding inertia tensor $\mathbf{J}'_i$ about the link i coordinate frame is given by the parallel axis theorem

$$\mathbf{J}'_i = \mathbf{J}_i + m_i(\underline{s}_i^T \underline{s}_i \mathbf{I}_3 - \underline{s}_i \underline{s}_i^T) \tag{31}$$

where $\mathbf{I}_3$ is a 3×3 identity matrix. In MAPLE this is implemented by

```
> read Jsubstit;
```

which subsitutes the symbols `Ixx.i` etc. for `Jxx.i` etc. Now that the torque equations are linear in the dynamic parameters they may be written in a form suitable for linear identification techniques to be applied

$$\underline{Q} = \mathbf{D}(\underline{q}, \underline{\dot{q}}, \underline{\ddot{q}})\, \underline{\Theta} \tag{32}$$

where $\underline{\Theta}$ is a $10n \times 1$ vector of inertial parameters

$$\underline{\Phi} = [m_1\ m_2\ \cdots I_{xx_1}\ I_{xy_1} \cdots s_{x_1} \cdots]$$

and $\mathbf{D}$ is an $n \times 10n$ matrix of state dependent dynamic terms. The i^{th} column will be referred to as $\mathbf{d}_i$.

$$\mathbf{D} = [\underline{d}_1\ \underline{d}_2 ... \underline{d}_m] \tag{33}$$

Once again making use of the `coeff()` function we can build the coefficients of the regressor matrix, $\mathbf{D}$. A vector containing the symbolic names of all inertial parameters is first established

```
params := [ m.(1..Naxis) Jxx.(1..Naxis) Jyy....
```

Generating the matrix $\mathbf{D}$ of (32) is now straightforward

```
Nparams := 10 * Naxis:
for j from 1 to Nparams do
    for i from 1 to Naxis do
        D[i,j] := coeff(tau.i, params[j]):
    od:
od:
```

that is, $\mathbf{D}_{ij}$ is the coefficient of Φ_j in the expression for τ_i.

Symbolic expansion of the dynamic equations reveals that for a specific manipulator, many of the inertial parameters do not appear, and many others appear only in linear combinations. There are three categories of inertial parameter[30]

Method	3-axis			6-axis		
	Mul	Add	Factors	Mul	Add	Factors
General numeric RNE	402	345		852	738	
Symbolically simplifed by MAPLE	238	108	45	538	254	106
Symbolic simplified by MAPLE after parameter value substitution	203	59	39	358	125	93
ARM closed form				461	352	
ARM recursive form				224	174	

- Uniquely identifiable.
- Identifiable in linear combinations. These are parameters that appear in linear combinations with other dynamic parameters. For the Puma 560 for example, the link masses m_4, m_5 and m_6 always appear in the torque expressions as the factor $(m_4+m_5+m_6)$. The sum $m_4+m_5+m_6$ may be identified, but not the individual link masses. Parameters in this category are those for which the corresponding $\underline{d}_i$ are linearly dependant, that is

$$\underline{d}_k = \alpha_1 \underline{d}_1 + \alpha_2 \underline{d}_2 + \cdots \alpha_{k-1} \underline{d}_{k-1}$$

- Unidentifiable. These are inertial parameters that have no effect on the joint torque and thus cannot be estimated. The unidentifiable parameters are those $\underline{\Theta_i}$ for which the corresponding $\underline{d}_i = \underline{0}$.

We can find the unidentifiable parameters by locating all columns of $\mathbf{D}$ whose norm is zero.

```
for i from 1 to Nparams do
    if norm(submatrix(D, 1..Naxis, i..i)) = 0 then
        lprint(`parameter`, params[i], ` is not identifiable`);
    fi;
od:
```

and results shown in Figure **??** agree with the results of Khalil when the different kinematic conventions are taken into account.

7 Online torque computation

The two major forms of control incorporating manipulator dynamics are computed torque control and feedforward control. Both require the evaluation of the equations of motion

$$\tau = rne(\underline{q},\ \underline{\dot{q}},\ \underline{\ddot{q}}) \tag{34}$$

The straightforward symbolic RNE formulation in Section 3 leads to very long sum of product expressions, which are extremely difficult to factorize

```
parameter    Jxx1    is not identifiable
parameter    Jzz1    is not identifiable
parameter    Jxy1    is not identifiable
parameter    Jyz1    is not identifiable
parameter    Jxz1    is not identifiable
parameter    SX1    is not identifiable
parameter    SY1    is not identifiable
parameter    SZ1    is not identifiable
parameter    m1    is not identifiable
parameter    Jm1    is not identifiable
parameter    Jm2    is not identifiable
parameter    Jm3    is not identifiable
parameter    Jm4    is not identifiable
parameter    Jm5    is not identifiable
parameter    Jm6    is not identifiable
15   null columns
```

Figure 4: List of unidentifiable (non-base) parameters for the Puma 560. Computed by the script `lindep`.

to an efficient run-time form — a better approach is to retain the factors during symbolic evaluation. A simple modification to the RNE program saves all non-zero symbolic factors and is enabled by the logical variable `keep_factors` before the `rne` script is run. After each intermediate expression in the RNE procedure is evaluated the function `keep()` is invoked, for example

```
if keep_factors then
        f.i := keep(f.i);
fi;
```

If the argument has more than one product terms then that expression is replaced by a single new variable, which is equated to the previous expression. Table 7 shows the operation count for this factored symbolic representation, as well as the number of factors generated, of the dynamic equations for the Puma 560 manipulator. The operation count is considerably less than the general form given in Table 1.

Table 7 shows the operation count for this factored symbolic representation of the dynamic equations for the Puma 560. The operation count is considerably less than the general form. Substituting numeric values[22] for inertial parameters, many of which are zero, reduces the operation count still further. The operation counts are compared in Table 7.

The MAPLE script `genCcode` takes the factorized symbolic expressions and produces three intermediate files: `TEMP-DECL`, `TEMPS`, and `TORQUES`.

Processor	Approach	Axes	Year	Time (μs)
PDP-11/45[11]	assembler, RNE	6	1982	4500
Sun3/60†	'C' general RNE	6	1988	13400
Sun4/75†	'C' general RNE	6	1990	500
μPD77230 DSP[31]	assembler, general RNE	6	1989	550
68030+68882 25MHz†	'C' + symbolic simplification	6	1989	2300
68030+68882 25MHz†	'C' + symbolic simplification	3	1989	1200
Sun4/75†	"	6	1990	144
Sun4/75†	"	3	1990	62

These are `#include`d into a wrapper function `rne.c` which defines the function

```
rne(double *Tau, double *q, double *qd, double *qdd)
```

Some typical execution times are summarized in Figure 7. Those marked with a † were benchmarked by the author, and compiler optimizations were enabled. It could be concluded that offline symbolic manipulation, high-level languages and state-of-the-art computers, provide a simple and powerful means of computing manipulator dynamics at a sufficiently high rate without the need to resort to parallel architectures or assembly language.

ARM has also been used[1] to generate customized computations for both direct and inverse dynamics. The latter can be expressed in either closed or recursive form, with the recursive form becoming advantageous with increasing numbers of joints. The results of ARM are compared with the results obtained using our simple MAPLE programs in Table 7.

8 Conclusion

References

[1] J. J. Murray, *Computational Robot Dynamics.* PhD thesis, Carnegie-Mellon University, 1984.

[2] G. Cesareo, F. Nicolo, and S. Nicosia, "DYMIR: a code for generating dynamic model of robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 115–120, 1984.

[3] J. Burdick, "An algorithm for generation of efficient manipulation dynamic equations," in *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 212–218, 1986.

[4] B. W. Char and K. O. Geddes, *Maple Reference Manual.* Symbolic Computation Group, University of Waterloo, 1988.

[5] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, *Robotics. Control, Sensing, Vision and Intelligence.* McGraw-Hill, 1987.

[6] C. S. G. Lee, B. Lee, and R. Nigham, "Development of the generalized D'Alembert equations of motion for mechanical manipulators," in *Proc. 22nd CDC*, (San Antonio, Texas), pp. 1205–1210, 1983.

[7] T. Kane and D. Levinson, "The use of Kane's dynamical equations in robotics," *Int. J. Robot. Res.*, vol. 2, pp. 3–21, Fall 1983.

[8] J. Uicker, *On the Dynamic Analysis of Spatial Linkages Using 4 by 4 Matrices.* PhD thesis, Dept. Mechanical Engineering and Astronautical Sciences, NorthWestern University, 1965.

[9] M. Kahn, "The near-minimum time control of open-loop articulated kinematic linkages," Tech. Rep. AIM-106, Stanford University, 1969.

[10] J. Hollerbach, "A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-10, pp. 730–736, Nov. 1980.

[11] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "On-line computational scheme for mechanical manipulators," *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 102, pp. 69–76, 1980.

[12] R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control.* Cambridge, Massachusetts: MIT Press, 1981.

[13] J. J. Craig, *Introduction to Robotics.* Addison Wesley, 1986.

[14] B. Char *et al.*, *Maple V language reference manual.* Springer-Verlag, 1991.

[15] P. Corke, "A robotics toolbox for MATLAB," *IEEE Robotics and Automation Magazine*, vol. 3, pp. 24–32, Mar. 1996.

[16] A. Izaguirre and R. Paul, "Automatic generation of the dynamic equations of the robot manipulators using a LISP program," in *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 220–226, 1986.

[17] W. M. Silver, "On the equivalance of Lagrangian and Newton-Euler dynamics for manipulators," *Int. J. Robot. Res.*, vol. 1, pp. 60–70, Summer 1982.

[18] P. Corke and B. Armstrong-Hélouvry, "A search for consensus among model parameters reported for the PUMA 560 robot," in *Proc. IEEE Int. Conf. Robotics and Automation*, (San Diego), pp. 1608–1613, May 1994.

[19] P. Corke, "A computer tool for simulation and analysis: the Robotics Toolbox for MATLAB," in *Proc. National Conf. Australian Robot Association*, (Melbourne), pp. 319–330, July 1995.

[20] P. I. Corke, "An automated symbolic and numeric procedure for manipulator rigid-body dynamic significance analysis and simplification," in *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 1018–1023, 1996.

[21] B. Armstrong, O. Khatib, and J. Burdick, "The explicit dynamic model and inertial parameters of the Puma 560 arm," in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 1, (Washington, USA), pp. 510–18, 1986.

[22] P. Corke and B. Armstrong-Hélouvry, "A meta-study of PUMA 560 dynamics: A critical appraisal of literature data," *Robotica*, vol. 13, no. 3, pp. 253–258, 1995.

[23] H. Mayeda, M. Maruyama, K. Yoshida, N. Ikeda, and O. Kuwaki, "Experimental examination of the identification methods for an industrial robot manipulator," in *Proc. 2nd International Symposium on Experimental Robotics*, (Toulouse, France), June 1991.

[24] B. Armstrong, *Dynamics for Robot Control: Friction Modelling and Ensuring Excitation During Parameter Identification.* PhD thesis, Stanford University, 1988.

[25] C. G. Atkeson, C. H. An, and J. M. Hollerbach, "Rigid body load identification for manipulators," *Proc. IEEE Conf. Decision and Control*, pp. 996–1002, 1985.

[26] C. H. An, "Trajectory and force control of a direct drive arm," Tech. Rep. 912, MIT Artificial Intelligence Laboratory, 1986.

[27] P. Khosla, *Real-Time Control and Identification of Direct-Drive Manipulators.* PhD thesis, Carnegie-Mellon University, 1986.

[28] A. Izaguirre, M. Hashimoto, R. Paul, and V. Hayward, "A new computational structure for real time dynamics," Tech. Rep. MS-CIS-87-107, University of Pennsylvania, Dec. 1987.

[29] P. Khosla and T. Kanade, "Parameter identification of robot dynamics," *Proc. IEEE Conf. Decision and Control*, pp. 1754–1760, 1985.

[30] P. Khosla, "Categorization of parameters in the dynamic robot model," *IEEE Trans. Robot. Autom.*, vol. 5, pp. 261–268, June 1989.

[31] M. Kabuka and R. Escoto, "Real-time implementation of the Newton-Euler equations of motion on the NEC μPD77230 DSP," *Micro Magazine*, vol. 9, pp. 66–76, Feb. 1990.

9 Listing of recursive NE script

A Summary of functions and scripts

Manipulator models	
`dh-puma560`	Symbolic kinematic parameters for Puma 560
`dh-2link`	Symbolic kinematic parameters for 2-link example from Fu, Gonzalez and Lee (p. ??)
`params-puma560`	Numerical inertial and kinematic parameters for Puma 560 (for standard D&H conventions)
`params-puma560-mdh`	Numerical inertial and kinematic parameters for Puma 560 (for modified D&H conventions)

Dynamics	
`rne`	Recursive Newton-Euler (standard D&H conventions)
`mrne`	Recursive Newton-Euler (modified D&H conventions)
`addmotor`	Add armature inertia to each torque expression, $G_i^2 J_{m_i}$
`puma6.m`	Precomputed torque expressions for Puma 560 (standard D&H conventions).

Substitutions and simplifications	
`trig_ident`	trigonometric simplifications
`simple560`	Puma560 specific simplifications
`count`	count number of product terms in a torque expression
`trigsub`	substitue robotics shorthand for native MAPLE trig functions
`trigunsub`	substitue native MAPLE trig functions for robotics shorthand
`Jsubstit`	substitute link-referred inertia (I) for COM-referred inertia (J)

Ranking and culling	
`tauhist`	function to produce a histogram of coefficient magnitudes
`top5simple`	list 5 most significant terms based on coefficient magnitude only
`top5nominal`	list 5 most significant terms based on coefficient magnitude and nominal velocity and acceleration
`top5random`	list 5 most significant terms based on coefficient magnitude evaluated over many random points in manipulator state space

Symbolic manipulations	
`components`	extract inertia, centripetal, Coriolis and gravity terms
`regressor`	compute the regressor matrix, D.
`unident`	compute the unidentifiable inertial parameters

Run-time code	
`genCcode`	create tempory files for inclusion in `rne.c`
`rne.c`	wrapper C language function that includes code modules created by `genCcode`

B Example session

The following session was perfomed using MAPLE V.3 and the worksheet `demo.ms`.

```
Warning: new definition for   norm
Warning: new definition for   trace
```

Load the kinematic parameters for the Puma 560. It displays the default number of axes (you can override this by subsequently setting N :=3 for instance), the source of the parameters and the conventions used, the direction of the gravity vector (this robot is mounted conventionally), and whether or not intermediate expressions such as link acceleration and velocity should be kept. Useful to keep them for debugging purposes, used to need to discard them due to memory limits but this is now a thing of the past.

```
> read `dh-puma560`;
```

$$N := 6$$

$$dh\_ident := 6 - axis\ Puma\ 560\ as\ per\ Paul\ and\ Zhang\ (1986)$$

$$gvec := [\,0\ 0\ grav\,]$$

$$keep\_intermediate := false$$

Now do the recursive Newton-Euler procedure symbolicaly.

```
> read rne;
```

$$keep\_factors := false$$

$$TempCounter := 0$$

$$< Recursive\ NE\ using\ notation\ of\ Paul >$$

$$6 - axis\ Puma\ 560\ as\ per\ Paul\ and\ Zhang\ (1986)$$

$$init\ vars$$

$$1$$

$$2$$

$$3$$

$$4$$

$$5$$

$$6$$

$$w0 := [\,0\;0\;0\,]$$

$$wd0 := [\,0\;0\;0\,]$$

$$\mathit{forward\ recursion}$$

$$1$$

$$2$$

$$3$$

$$4$$

$$5$$

$$6$$

$$\mathit{backward\ recursion}$$

$$6$$

```
tau   6   has    1331    terms
```

$$5$$

```
tau   5   has    2421    terms
```

$$4$$

```
tau   4   has    3552    terms
```

$$3$$

```
tau   3   has    8919    terms
```

$$2$$

```
tau   2   has    11046    terms
```

$$1$$

```
tau   1   has    25728    terms
```

We now have a number of variables in the workspace such as tau1, tau2 etc. which are the torque expressions (equations of motion really) for the various joints. These are in the form of a sum of terms, and each term is a product of a numeric coefficient (derived from manipulator inertial and kinematic parameters) and various trigonometric terms.These are not in the simplest possible form, so we use the simple script to apply some trigonmetric simplifications. The inbuilt MAPLE rules can't be used since we are still using the roboticist's shorthand of S1 and C1 to represent $\sin\theta_1$ and $\cos\theta_1$.

```
> read simple560;
```

$$simplifying\ tau, 1$$

```
S6^2+C6^2   2244   substitutions
S5^2+C5^2   1378   substitutions
S4^2+C4^2   802   substitutions
S3^2+C3^2   224   substitutions
S2^2+C2^2   7   substitutions
4655    terms eliminated
S23   2134   substitutions
C23   1288   substitutions
3422   terms eliminated
```

$$simplifying\ tau, 2$$

```
S6^2+C6^2   976   substitutions
S5^2+C5^2   584   substitutions
S4^2+C4^2   406   substitutions
S3^2+C3^2   49   substitutions
2015    terms eliminated
S23   1111   substitutions
C23   608   substitutions
1719   terms eliminated
```

$$simplifying\ tau, 3$$

```
S6^2+C6^2   702   substitutions
S5^2+C5^2   423   substitutions
S4^2+C4^2   330   substitutions
1455    terms eliminated
S23   1111   substitutions
C23   608   substitutions
1719   terms eliminated
```

$$simplifying\ tau, 4$$

```
S6^2+C6^2   132   substitutions
S5^2+C5^2   209   substitutions
S3^2+C3^2   20   substitutions
S2^2+C2^2   10   substitutions
371    terms eliminated
S23   533   substitutions
C23   288   substitutions
821   terms eliminated
```

$$simplifying\ tau, 5$$

```
S6^2+C6^2   164   substitutions
S4^2+C4^2   30   substitutions
S3^2+C3^2   10   substitutions
S2^2+C2^2   5   substitutions
```

```
209    terms eliminated
S23   324   substitutions
C23   214   substitutions
538   terms eliminated
```

$$simplifying\ tau, 6$$

```
S4^2+C4^2   12   substitutions
S3^2+C3^2   8   substitutions
S2^2+C2^2   4   substitutions
24    terms eliminated
S23   197   substitutions
C23   128   substitutions
325   terms eliminated
```

Now check the number of product terms in each torque expression. There has been a considerable reduction.

```
> read count;
tau   1   has    16222    terms
tau   2   has    6738    terms
tau   3   has    5421    terms
tau   4   has    2329    terms
tau   5   has    1674    terms
tau   6   has    982    terms
```

Look at the first 5 elements of the waist joint torque expression

```
> op(1..5,tau1);
```

$$-2\, C3\, S6\, C5^2\, S3\, S2^2\, qd1\, Ixy6\, C6\, S4^2\, qd3,\\ -C4^3\, D3\, S5\, m6\, C5^2\, S3\, S2^3\, qd1^2\, sz6\, C3^2,\\ C4^3\, D3\, m4\, C3^2\, S2^3\, qd1^2\, sx4\, S3, 2\, C3\, qd2\, Izz3\, S3\, S2^2\, qd1,\\ 2\, sy2\, m2\, S2^2\, qd1\, qd2\, A2$$

We may wish also to express the equations of motion in the classic form

$$\tau = \mathbf{M}(q)\ddot{q} + \mathbf{C}(q, \dot{q})\dot{q} + \mathbf{g}$$

The coefficients $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{g}$ can be obtained simply by factorizing the equations of motion

```
> read components;
inertia tensor
Coriolis/centripetal tensor
gravity vector
```

So far everything has been symbolic, so lets substitute in some parameters. For the Puma 560 hese parameters have been derived from many sources (see papers by Corke & Armstrong). All units here are standard SI.

```
> read `params-puma560`;
```

Now many parameters, particularly products of inertia are zero, so we can simplify further.

```
> read simple560;
```

$$simplifying\ tau, 1$$

```
S6^2+C6^2   114   substitutions
114    terms eliminated
0   terms eliminated
```

$$simplifying\ tau, 2$$

```
S6^2+C6^2   48   substitutions
48    terms eliminated
0   terms eliminated
```

$$simplifying\ tau, 3$$

```
S6^2+C6^2   48   substitutions
48    terms eliminated
0   terms eliminated
```

$$simplifying\ tau, 4$$

```
S6^2+C6^2   35   substitutions
35    terms eliminated
0   terms eliminated
```

$$simplifying\ tau, 5$$

```
S6^2+C6^2   40   substitutions
40    terms eliminated
0   terms eliminated
```

$$simplifying\ tau, 6$$

```
0    terms eliminated
0   terms eliminated
```

```
> read count;
tau   1   has    1140    terms
tau   2   has    484    terms
tau   3   has    353    terms
tau   4   has    94    terms
tau   5   has    90    terms
tau   6   has    18    terms
```

The inertia of the shoulder joint is given by

```
> M22;
```

$$
\begin{aligned}
&.02191385000\, C3 - .7697525880\, S3 + .001243584\, S5^2\, C5\, S4^2 \\
&\quad - .000058464\, S4^2\, S5\, C4 - .0024871680\, C4\, S5\, C3 \\
&\quad - .000058464\, C4^3\, S5 - .0024871680\, C5\, S3 - .000058464\, C4\, S5 \\
&\quad + .001243584\, C5\, S4^2 + .002487168\, C5\, C4^2 + .00015\, C6^2\, C4^2
\end{aligned}
$$

$$+ .00039216\, C5^2\, S4^2 + .00034\, S5^2\, S4^2 + .00180\, S4^2 + .00015\, C6^2\, C5^2\, S4^2 + .00229216\, C4^2 + .00015\, S6^2\, C5^2\, S4^2 + .001243584\, C5^3\, S4^2 + .00015\, S6^2\, C4^2 + 2.104964074$$

$$.00015\, C6^2\, C5^2\, S4^2 + .02191385000\, C3 - .7697525880\, S3 + .00015\, C6^2\, C4^2 + .00015\, S6^2\, C5^2\, S4^2 - .000058464\, S4^2\, S5\, C4 - .0024871680\, C4\, S5\, C3 + .001243584\, S5^2\, C5\, S4^2 + .001243584\, C5\, S4^2 + .00034\, S5^2\, S4^2 + .00180\, S4^2 + 2.104964074 - .000058464\, C4^3\, S5 - .0024871680\, C5\, S3 + .001243584\, C5^3\, S4^2 - .000058464\, C4\, S5 + .002487168\, C5\, C4^2 + .00039216\, C5^2\, S4^2 + .00015\, S6^2\, C4^2 + .00229216\, C4^2$$

and the inertial coupling between the waist and shoulder joints is

```
> M12;
```

$$-.0038189375\, S23 - .1388827650\, C23 - .023751000\, C2 - .6915669190\, S2 - .001243584\, C23\, S5\, S4 - .0012435840\, S2\, S5\, S4 - .000433440\, C23\, C5 - .000058464\, S23\, S5\, S4 + .00049216\, S23\, S4\, C4 - .00005216\, C23\, C5\, S5\, S4 - .00039216\, S23\, C4\, C5^2\, S4 + .00015\, S23\, S4\, C6^2\, C4 - .00015\, C23\, S5\, C6^2\, C5\, S4 + .001243584\, S23\, C4\, C5\, S4 + .00015\, S23\, S4\, S6^2\, C4 - .001243584\, S23\, C4\, S5^2\, C5\, S4 - .001243584\, S23\, C4\, C5^3\, S4 - .00015\, S23\, C4\, C6^2\, C5^2\, S4 - .00034\, S23\, C4\, S5^2\, S4 - .00015\, S23\, C4\, S6^2\, C5^2\, S4 + .000433440\, S23\, S4^2\, S5\, C4 - .00015\, C23\, S5\, S6^2\, C5\, S4 + .000433440\, S23\, C4^3\, S5$$

$$-.0038189375\, S23 - .000433440\, C23\, C5 - .1388827650\, C23 + .00015\, S23\, S4\, S6^2\, C4 + .00015\, S23\, S4\, C6^2\, C4 - .00005216\, C23\, C5\, S5\, S4 - .023751000\, C2 - .00039216\, S23\, C4\, C5^2\, S4 - .6915669190\, S2 - .00034\, S23\, C4\, S5^2\, S4 - .0012435840\, S2\, S5\, S4 - .001243584\, C23\, S5\, S4 - .001243584\, S23\, C4\, C5^3\, S4 + .000433440\, S23\, S4^2\, S5\, C4 + .001243584\, S23\, C4\, C5\, S4 - .00015\, S23\, C4\, S6^2\, C5^2\, S4 - .00015\, C23\, S5\, S6^2\, C5\, S4 - .001243584\, S23\, C4\, S5^2\, C5\, S4 - .00015\, S23\, C4\, C6^2\, C5^2\, S4 + .000433440\, S23\, C4^3\, S5 - .00015\, C23\, S5\, C6^2\, C5\, S4 - .000058464\, S23\, S5\, S4 + .00049216\, S23\, S4\, C4$$

Now we can plot these expressions as a function of pose. But first we need to replace the robot shorthand with proper trig expressions;

```
> read trigsub;
```

```
shorthand replaced: S.i -> sin(qi) etc.
```

```
> M22;
```

$$
\begin{aligned}
&.002487168\cos(\,q5\,)\cos(\,q4\,)^2 + .001243584\sin(\,q5\,)^2\cos(\,q5\,)\sin(\,q4\,)^2 \\
&\quad - .000058464\sin(\,q4\,)^2\sin(\,q5\,)\cos(\,q4\,) \\
&\quad - .0024871680\cos(\,q4\,)\sin(\,q5\,)\cos(\,q3\,) \\
&\quad + .00015\cos(\,q6\,)^2\cos(\,q5\,)^2\sin(\,q4\,)^2 \\
&\quad + .00015\sin(\,q6\,)^2\cos(\,q5\,)^2\sin(\,q4\,)^2 \\
&\quad - .0024871680\cos(\,q5\,)\sin(\,q3\,) + .001243584\cos(\,q5\,)\sin(\,q4\,)^2 \\
&\quad - .000058464\cos(\,q4\,)\sin(\,q5\,) + .001243584\cos(\,q5\,)^3\sin(\,q4\,)^2 \\
&\quad - .000058464\cos(\,q4\,)^3\sin(\,q5\,) + .00015\cos(\,q6\,)^2\cos(\,q4\,)^2 \\
&\quad + .00039216\cos(\,q5\,)^2\sin(\,q4\,)^2 + .02191385000\cos(\,q3\,) \\
&\quad - .7697525880\sin(\,q3\,) + .00180\sin(\,q4\,)^2 + .00229216\cos(\,q4\,)^2 \\
&\quad + 2.104964074 + .00015\sin(\,q6\,)^2\cos(\,q4\,)^2 \\
&\quad + .00034\sin(\,q5\,)^2\sin(\,q4\,)^2
\end{aligned}
$$

and we can see the substitution has occurred.

```
Warning in iris-plot: empty plot
```

Now we can plot the inertia of the waist joint as shoulder and elbow angle varies. We need first to set the waist angle to a finite value (the value doesn't matter since it doesn't effect the inertia) and also set the wrist pose (to zero angles in this case).

```
> plot3d( subs(q1=0,q4=0,q5=0,q6=0, M11),  q2=-Pi..Pi, q3=-Pi..Pi);
```

gives the plot shown in Figure 6. Before proceeding we need to put back the roboticist's trig shorthand

```
> read trigunsub;
shorthand restored: sin(qi) -> S.i etc.
```

As mentioned above the torque expressions are in the form of a sum of terms, and each term is a product of a numeric coefficient (derived from manipulator inertial and kinematic parameters) and various trigonometric terms. A histogram of the magnitude of the coefficients can be plotted by

```
> read tauhist;
> tauhist(tau1);
.5056548778
```

Now lets find the 'most significant' terms in the torque expressions. This first method is very simple minded, it just looks at the magnitude of the coefficients.

```
> read top5simple;
```

$$top\_ranked := true$$

$$tab1 :=$$
$$[1, -3.204\, S2\, C2\, qd1\, qd2, 37.23\, C2, -8.744 \sin(q2+q3)]$$
$$[2, 1.732\, C2^2\, qdd1, -8.744 \sin(q2+q3), -.3849\, S3\, qdd2]$$
$$[3, 1.409\, qdd1, 2.105\, qdd2, .3849\, C3\, qd2^2]$$
$$[4, .8448\, C2\, S3^2\, C3^2\, S2^2\, qd1^2, 1.602\, S2\, qd1^2\, C2,$$
$$-.3849\, S3\, S2\, qd1^2\, C2]$$
$$[5, .8448\, S3^2\, C3^2\, C2^3\, qd1^2, -1.024\, S2, .3849\, C3\, C2^2\, qd1^2]$$

$$tab2 :=$$
$$[1, .02825 \sin(q2+q3)\, S5\, S4, -.02825 \cos(q2+q3)\, S5, 0]$$
$$[2, .002487\, S5\, C4\, S2\, qd1\, qd2, -.02825 \sin(q2+q3)\, C5\, C4, 0]$$
$$[3, .002487 \cos(q2+q3)\, S5\, C4\, qd1\, qd3,$$
$$.002487\, S5\, C3\, S2\, qd1^2\, S3\, C2, 0]$$
$$[4, .002487 \cos(q2+q3)\, S5\, C4\, qd1\, qd2,$$
$$.002487\, C5\, S4\, S2\, qd1\, qd2, 0]$$
$$[5, .00130 \cos(q2+q3)\, qdd1,$$
$$.002487 \cos(q2+q3)\, C5\, S4\, qd1\, qd3, 0]$$

other ranking methods are implemented by the scripts top5nominal and top5random (see paper by Corke ICRA '96). Note that these more complex ranking procedures are extremely voracious for memory and CPU cycles.

All the techniques above are based on a symbolic expansion of the equations of motion, which as we have seen result in extremely long expressions. All the computational advantage of the recursive Newton-Euler procedure is lost if these long expressions were used to compute torque. However a simple trick allows us to create computationally efficient closed form torque expressions. This is supported in the *ne* script used above, but needs to be enabled

```
> keep_factors := true;
```

$$keep\_factors := true$$

Now run the script again.

```
> read rne;
```

$$TempCounter := 0$$

$$< \mathit{Recursive\ NE\ using\ notation\ of\ Paul} >$$

$$6 - \mathit{axis\ Puma\ 560\ as\ per\ Paul\ and\ Zhang\ (1986)}$$

$$\mathit{init\ vars}$$

$$1$$

$$2$$

$$3$$
$$4$$
$$5$$
$$6$$
$$w0 := [\,0\;0\;0\,]$$
$$wd0 := [\,0\;0\;0\,]$$
$$forward\ recursion$$
$$1$$
$$2$$
$$3$$
$$4$$
$$5$$
$$6$$
$$backward\ recursion$$
$$6$$

```
tau   6   has    2    terms
```

$$5$$

```
tau   5   has    2    terms
```

$$4$$

```
tau   4   has    1    terms
```

$$3$$

```
tau   3   has    2    terms
```

$$2$$

```
tau   2   has    1    terms
```

$$1$$

```
tau   1   has    1    terms
```

During the symbolic expansion intermediate variables (t0, t1, t2, ...) have been created and used in subsequent expressions. The number of intermediate variables used in this case can be determined from the final value of the tempory expression counter

```
> TempCounter;
```

$$93$$

The resulting torque expressions are now in terms of these intermediate expressions, for instance;

```
> tau1;
```

$$T92$$

Now the symbol T92 is just a place holder, the actual value of the expression has been saved in the related variable

```
> t92;
```

$$T86\ S2 + T87\ C2 + .35\ qdd1$$

which is defined in terms of other place holder variables. To create a form useful for run time we use the script

```
> read genCcode;

proc() ... end
```

$$0$$

```
Building substitutions
writing TEMP-DECL
writing TEMPS
writing TORQUES
** compile with gcc -c dyn.c
```

which creates a number of tempory files called TEMP-DECL, TEMPS and TORQUES which are included into the generalized function dyn() whose source is given in dyn.c. Compiling this module then provides the programmer with a computationally efficient form for the manipulator's inverse dynamics.

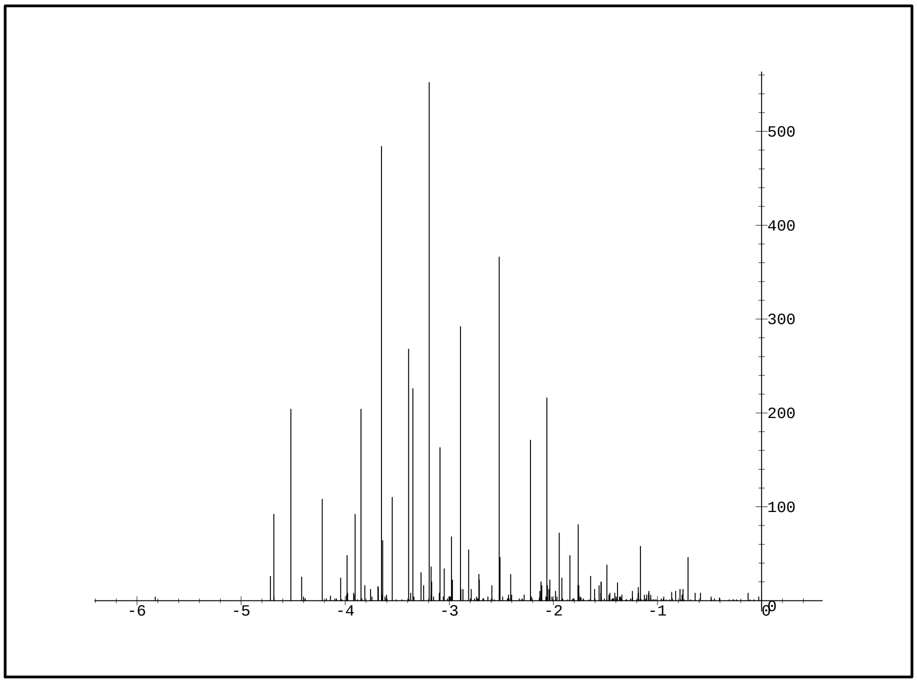

Figure 5: Histogram of product term coefficients for $\tau 1$.

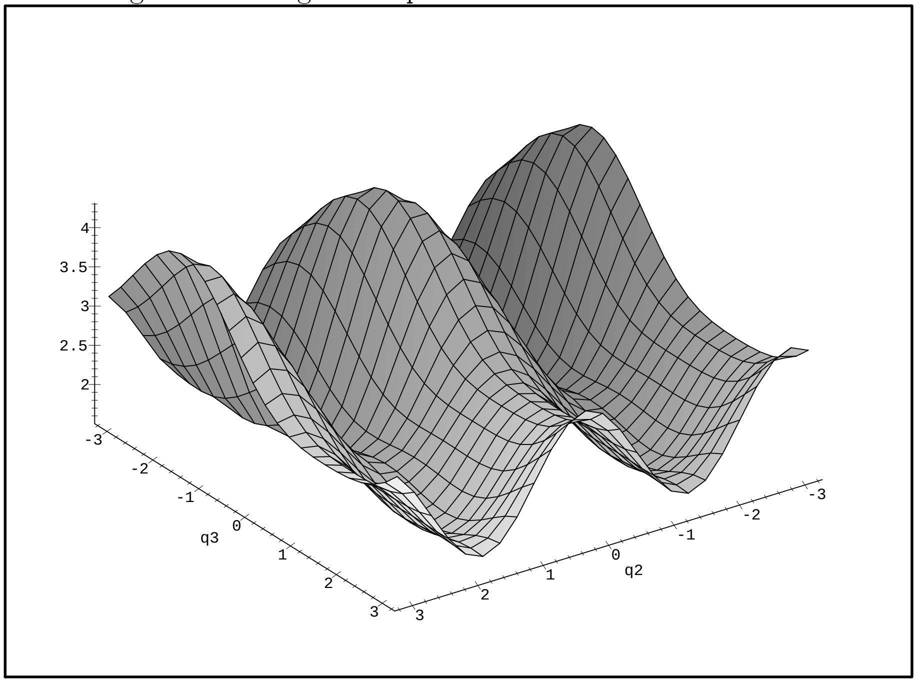

Figure 6: Plot of joint 1 inertia as a function of joint 2 and 3 angle.