Pergamon

Mech. Mach. Theory Vol. 33, No. 1/2, pp. 163–174, 1998

Printed in Great Britain
PII: S0094-114X(97)00012-8
0094-114X/98 $19.00 + 0.00

STATICAL BALANCING OF A ROBOT MECHANISM WITH THE AID OF A GENETIC ALGORITHM

S. SEGLA
Department of Mechanical Engineering, Technical University of Kosice, 041 87 Kosice, Slovakia

C. M. KALKER-KALKMAN and A. L. SCHWAB
Department of Mechanical Engineering and Marine Technology, Delft University of Technology, Delft, The Netherlands

(Received 7 August 1995; in revised form 28 January 1996; received for publication 1997)

Abstract—In designing robots statical balancing is of great importance. It enables essential reduction of the required motor power. Elimination of significant reduction of the gravity load at a powered joint can also lead to a simpler and more effective control system. It is shown that this problem can be formulated as an optimization problem. As objective function the average force on the gripper in the working area is used. It is shown that those forces can easily be derived with the aid of a Computer Algebra System. The lengths of the links and angles between them as well as the stiffness of springs can be considered as design variables. It is shown that a general design program based on Monte Carlo methods and genetic algorithms, that has a graphical interface that allows the user to modify search intervals, is very appropriate to these kinds of problems. As an example an industrial robot with 6-DOF is treated. The robot has a spring balancing system that has to be optimized. The robot with the parameters of its balancing system found this way is evaluated with an existing multibody dynamics program SPACAR that simulates the movements of the robot and calculates the forces at its powered joints. With the aid of MATLAB the forces in the working area are plotted in a 3D figure. The method described in this article is general and very appropriate to solve problems in practice without simplifications. © 1998 Elsevier Science Ltd

1. INTRODUCTION

To keep a robot in static equilibrium, forces are needed at the powered joints. These static gravity forces are caused by the masses of the robot links and by the mass of the payload at the gripper. The mass of the robot links is often 10–20 times as large as the mass at the payload. Balancing of the robot links (the robot mechanism) is often sufficient. The balancing system should produce such forces that should be able to eliminate, or at least essentially reduce, the static gravity forces at the powered joints.

In practice two basic ways of balancing are mostly used: balancing by springs and balancing by masses (using added counter-weights or by links mass redistribution). The advantage of the balancing by springs compared to the balancing by counter-weights is connected with the fact that the weights of the built-in springs or cylinders compared to the link weights can be neglected and therefore the gravity loads are eliminated or reduced without changing the mass and inertia parameters of the robot mechanism. In some cases a combination of spring and mass balancing (especially mass redistribution of the links) may result in better solutions.

When designing a robot, the lengths of the links, angles between them and spring-stiffnesses can be considered as (independent) design variables. The average force on the gripper in the working area of the robot can be considered as objective function to be minimized. In the ideal situation all forces are zero in this working area.

This means that we do not search a local minimum, we want the global minimum to be zero. For this reason a genetic algorithm is very appropriate, as this finds a global optimum. In the following sections the spring balancing system of an industrial robot APR 20 (see Fig. 1) will be optimized. The reader has to bear in mind that the method described can be applied to any robot or mechanism to be designed to produce a certain kinematic, static or dynamic behaviour [1]. In

Section 2 the problem is formulated as an optimization problem, and the objective function is derived with the aid of Maple [2]. In Section 3 the optimization method is discussed and in Section 4 the results for the robot APR 20 are given. In Section 5 these results are evaluated with a general multi-body program SPACAR [3] that is able to simulate the movements of any input-given mechanism, and that is able to compute the forces in any position. It is also shown that with the aid of a simple interface those forces can be plotted with the aid of MATLAB [4].

2. FORMULATION OF THE PROBLEM

It is easy to see that of the 6-DOF of the mechanism only two have to be balanced, namely $q2$ and $q3$, see Fig. 1. When we assume a mass of zero at the gripper, $q4$, $q5$ and $q6$ have no influence. The rotation about the vertical axis ($q1$) is already balanced when the robot is in a correctly vertical position. This simplifies the balancing considerably as the spatial robot can now be reduced to a planar mechanism with 2-DOF. It can be proved [5] that in the case of the massless links 6 and 8 the static gravity forces at the degrees of freedom $q2$ and $q3$ caused by gravitation forces of links 5 and 7 can be completely eliminated. The problem of the statical balancing of the robot taking into account masses of the links 6 and 8 will be formulated as an optimization problem. Spring mechanisms are used to balance the robot APR 20 (see Figs 3 and 4). Figure 3 shows statical balancing of link 7. The balancing moment is transmitted to link 7 by a mechanical belt and pulley transmission. The first pulley wheel of the transmission placed on the rotating base of the robot can rotate independently of the rotation of link 5. The second one (of the same diameter) is attached to link 7. One end of the spring rod of this balancing mechanism is connected to the lower pulley by a revolute joint at W and a balancing spring 2 of constant stiffness $k2$ is placed between the other end of the spring rod and a joint at $V2$ (placed on the rotating base of the robot) which allows rotation and translation of the spring rod. Figure 4 shows the balancing mechanism of link 5 which is similar to the previous one. The spring rod is connected to link 5 by a revolute joint at T.

We have to find such parameters of both balancing mechanisms which minimize the forces Fx and Fy acting on point C (see Fig. 2). These forces have to ensure statical balance of the robot in any position of the robot gripper (M) in the working area of the robot $EFGH$. It should be noted that thanks to a parallelogram $ACBD$ there is a simple relation between the position of the robot gripper (M) in the rectangle $EFGH$ and the position of the joint C in the smaller rectangle $E'F'G'H'$.

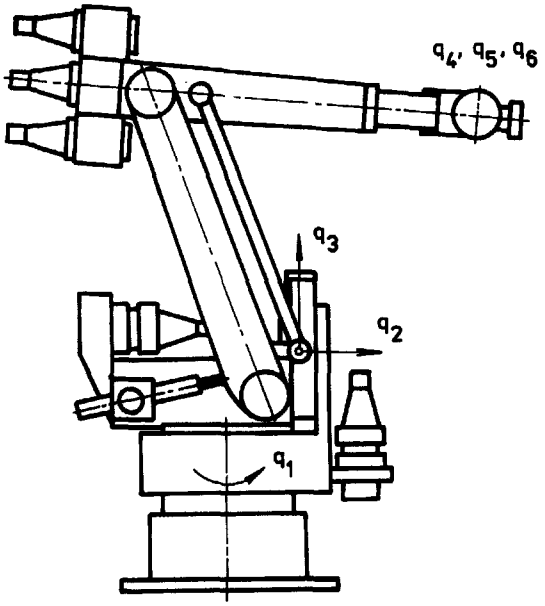

Fig. 1. Robot APR 20.

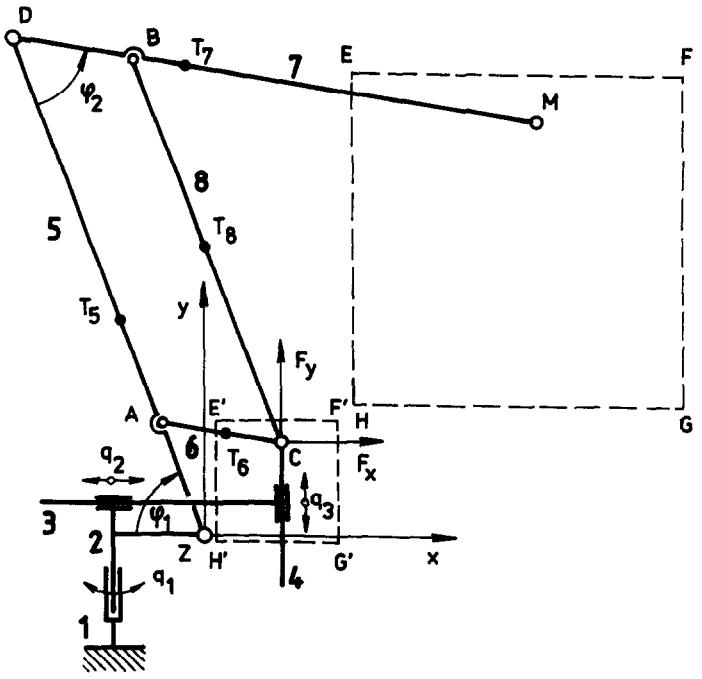

Fig. 2. Scheme of the robot APR 20.

2.1. *Variables*

In order to formulate the optimization problem mathematically we need to introduce a number of variables (see Figs 2, 3, and 4). In Section 2.5 we will see that many of the variables given below can be regarded as design variables of both balancing mechanisms of the robot.

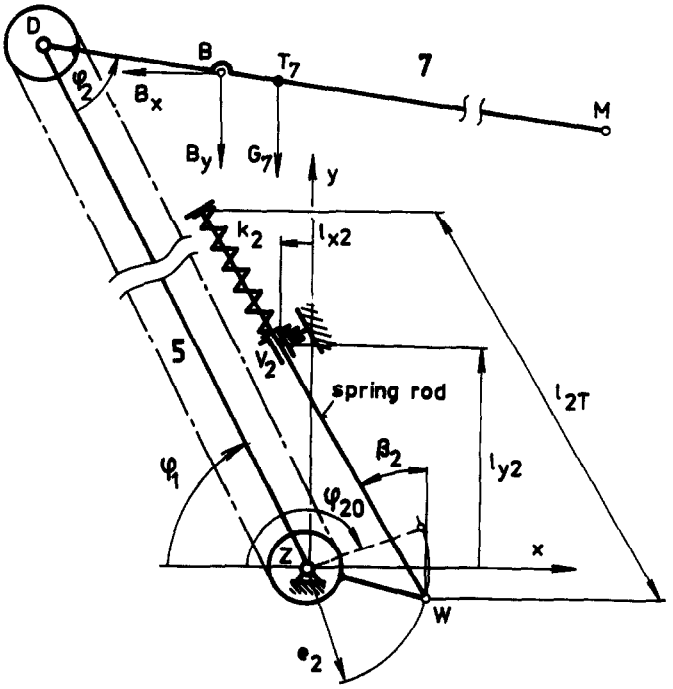

Fig. 3. Static equilibrium of the link 7.

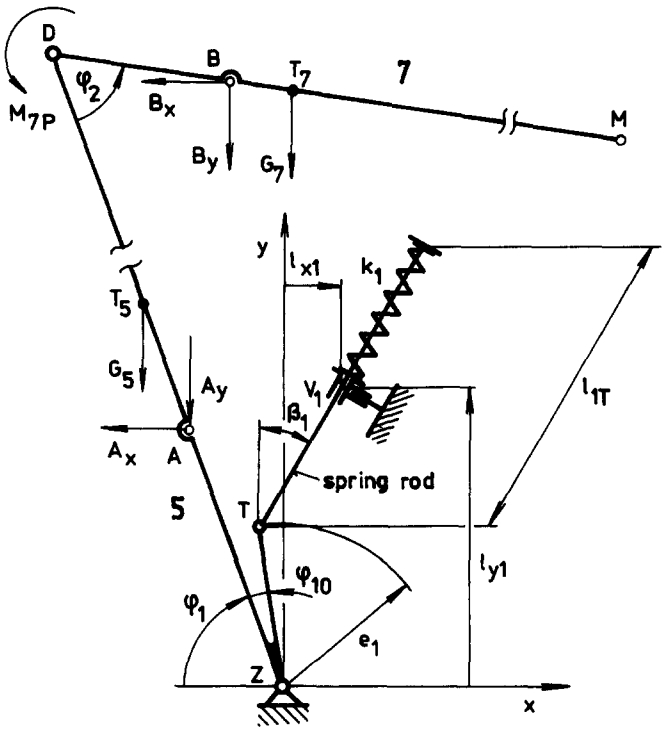

Fig. 4. Static equilibrium of the links 5 and 7 together.

Coordinates of the robot mechanism: $\phi 1, \phi 2$
Masses of robot links: $m5, m6, m7$ (includes mass of the robot gripper and nominal payload of 10 kg), $m8$
Lengths of links: 15, 16, 17, 18, $AZ = a$, $DB = b$
Lengths determining centres of gravity: $CT8 = c$, $DT7 = d$, $ZT5 = p$, $AT6 = q$
Forces acting on point C in horizontal and vertical directions: Fx, Fy
Reaction forces: Ax, Ay, Bx, By
Stiffness of springs 1 and 2: $k1, k2$
Lengths of unloaded springs 1 and 2: 101, 102
Lengths of spring rods: $l1T$, $l2T$
Distances determining position of point $V1$: $lx1, ly1$
Distances determining position of point $V2$: $lx2, ly2$
Distances between points ZT, ZW: $e1, e2$
Permissible spring deflections: $tm1, tm2$
Angle determining position of point T: $\phi 10$
Angle determining position of point W (when $\phi 1 = \phi 2$): $\phi 20$

2.2. *Equilibrium conditions*

It is clear that the forces Fx and Fy have to be minimized for all occurring positions of the robot. These forces are functions of the coordinates $\phi 1$ and $\phi 2$ and, of course, of the design variables of the robot. To compute the forces, we write down the following static equilibrium conditions. For static equilibrium of the links 6 and 8 together (see Fig. 5) in x and y directions we write:

$$Ax + Bx + Fx = 0, \tag{1}$$

$$Ay + By + Fy - G6 - G8 = 0 \tag{2}$$

and equilibrium condition of moments about point A

$$- By[16\cos(\phi 1 - \phi 2) - 18\cos\phi 1] + Bx[18\sin\phi 1 - 16\sin(\phi 1 - \phi 2)] + G6q\cos(\phi 1 - \phi 2)$$
$$+ G8[16\cos(\phi 1 - \phi 2) - c\cos\phi 1] - Fx16\sin(\phi 1 - \phi 2) - Fy16\cos(\phi 1 - \phi 2) = 0 \quad (3)$$

We can also use the equilibrium of moments of link 8 about point C

$$Bx18\sin\phi 1 + By18\cos\phi 1 - G8c\cos\phi 1 = 0 \quad (4)$$

For gravitational forces we have the relations: $Gi = mi.g$ (i = 5–8), where g is the acceleration due to gravity. The equilibrium condition of the link 7, that expresses equilibrium of all moments about point D, can be written in the form:

$$G7d\cos(\phi 1 - \phi 2) + Byb\cos(\phi 1 - \phi 2) + Bxb\sin(\phi 1 - \phi 2) - M7P = 0 \quad (5)$$

where $M7P$ is the spring balancing moment

$$M7P = k2\{102 - [12T - \sqrt{(a1^2 + a2^2)}]\}e2.\sin(3\pi/2 - \beta 2 - \phi 20 - \phi 1 + \phi 2), \quad (5a)$$

with

$$a1 = e2\cos(3\pi/2 - \phi 20 - \phi 1 + \phi 2) + ly2, \quad (5b)$$

$$a2 = e2\sin(3\pi/2 - \phi 20 - \phi 1 + \phi 2) + lx2, \quad (5c)$$

and

$$\beta 2 = \text{arctg}(a2/a1) \quad (5d)$$

Finally we write down the static equilibrium of links 5 and 7 together (equilibrium of all moments about point Z, see Fig. 4)

$$- G7[15\cos\phi 1 - d\cos(\phi 1 - \phi 2)] - Bx[15\sin\phi 1 - b\sin(\phi 1 - \phi 2)]$$
$$- By[15\cos\phi 1 - b\cos(\phi 1 - \phi 2)] - M7P - G5p\cos\phi 1$$
$$- Ax\,a\sin\phi 1 - Ay\,a\cos\phi 1 - M5P = 0 \quad (6)$$

where $M5P$ is the spring balancing moment

$$M5P = -k\{101 - [l1T - \sqrt{(b1^2 + b2^2)}]\}e1.\sin(\beta 1 + \pi/2 - \phi 1 - \phi 10), \quad (6a)$$

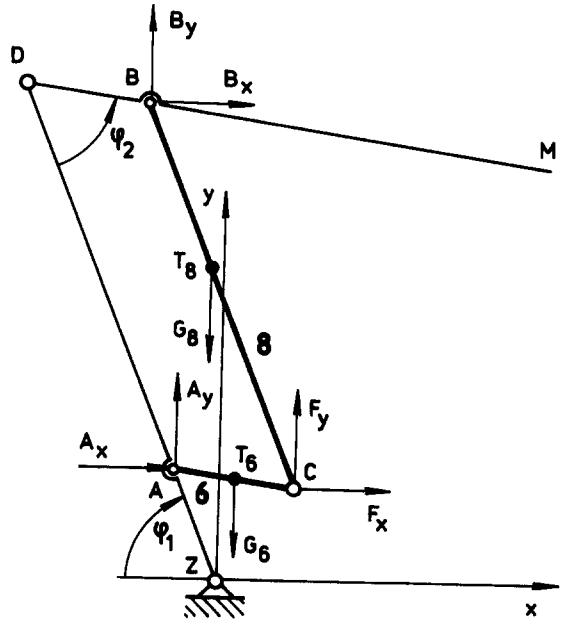

Fig. 5. Determination of the forces Fx and Fy.

with

$$b1 = e1\cos(\phi 1 + \phi 10) - lx1, \tag{6b}$$

$$b2 = -e1\sin(\phi 1 + \phi 10) + ly1 \tag{6c}$$

and

$$\beta 1 = \text{arctg}(b1/b2). \tag{6d}$$

It is clear that Fx and Fy as well as Ax, Ay, Bx, By can be solved as functions of the coordinates $\phi 1$ and $\phi 2$ from the equations (1)–(6). In the next section this will be done with the aid of a Computer Algebra Software System.

2.3. Solution of the forces from the equilibrium equations

Equations (1)–(6) can be regarded as a set of six linear equations in Fx, Fy, Ax, Ay, Bx and By. We are only interested in the values of Fx and Fy. As we want to compute Fx and Fy for various values of $\phi 1$ and $\phi 2$, and as we want to compute all those values for various values of the design variables of the robot we must be able to compute Fx and Fy in a very fast way. The fastest way is to solve equations (1)–(6) symbolically and we will do that with Maple [2].
First, we write

$$cv = \cos(\phi 1 - \phi 2) \quad sf1 = \sin\phi 1 \quad sf2 = \sin\phi 2$$

$$sv = \sin(\phi 1 - \phi 2) \quad cf1 = \cos\phi 1.$$

It is clear that we have

$$cf1.sv - cv.sf1 = -sf2 \tag{7}$$

It is easy to derive that equations (1)–(6) can be written as:

$$\begin{vmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & sv & cv \\ 0 & 0 & sf1 & cf1 & 0 & 0 \\ 0 & 0 & sv & cv & 0 & 0 \\ sf1 & cf1 & 0 & 0 & 0 & 0 \end{vmatrix} \begin{vmatrix} Ax \\ Ay \\ Bx \\ By \\ Fx \\ Fy \end{vmatrix} = \begin{vmatrix} 0 \\ A \\ B \\ C \\ D \\ E \end{vmatrix} \tag{8}$$

with

$$A = G6 + G8$$

$$B = (16.cv.G8 + q.cv.G6)/16 - (M7P - d.cv.G7)/b$$

$$C = c.cf1.G8/18$$

$$D = (M7P - d.cv.G7)/b$$

$$E = (-15.cf1.G7 - M5P - p.cf1.G5 - 15.c.cf1.G8/18)/a \tag{9}$$

Using Maple [2] we find the following solutions for Fx and Fy:

$$Fx = cf1.cv(G5.p/a + G6 + M5P/(cf1.a) + 15.G7/a - G6.q/16 - G8.c/18 - G7.d/b + M7P/(cv.b) + 15.G8.c/(a.18))/sf2 \tag{10}$$

and

$$Fy = cf1.sv(-15.c.G8/(a.18) + G8.c/18 - 15.G7/a - G5.p/a - M5P/(a.cf1) - G6)/sf2 + sf1.cv(-M7P/(b.cv) + G7.d/b + G6.q/16)sf2 + G8 \tag{11}$$

These expressions for the forces Fx and Fy will be used in the computation of the objective function in Section 2.4.

2.4. Objective function

When the variables of the robot and its balancing mechanisms given in Section 2.1. are known, the forces Fx and Fy can be computed from equations (10) and (11) for all possible values of the coordinates $\phi 1$ and $\phi 2$. In the ideal situation the forces would be zero. Let us define a coordinate system $Z(x, y)$, see Fig. 2. Then the coordinates $\phi 1$ and $\phi 2$ can be computed from the coordinates x and y of point C using equations

$$\phi 2 = 2 \arcsin(\sqrt{(x^2 + y^2)}/2a \tag{12}$$

and

$$\phi 1 = \pi/2 + \phi 2/2 - \arcsin y/\sqrt{(x^2 + y^2)} \tag{13}$$

As mentioned in Section 2, the position of point M in the rectangle $EFGH$ corresponds to the position of point C in the rectangle $E'F'G'H'$ which is determined by intervals of coordinates x and y: $0.115 \text{ m} \leqslant x \leqslant 0.295 \text{ m}$ and $-0.025 \text{ m} \leqslant y \leqslant 0.155 \text{ m}$. We are now able to compute Fx and Fy for any position of point C in the rectangular $E'F'G'H'$.

An appropriate objective function can be defined in the form:

$$fav = \sum_{i=1}^{N} \frac{\sqrt{Fxi^2 + Fyi^2}}{N} \tag{14}$$

where N is the number of points in the rectangle $E'F'G'H'$ at which the forces Fx and Fy are computed. For this purpose a rectangular grid will be used. It is obvious that the objective function defined by equation (14) represents an average resulting force acting on point C which is needed to balance the robot. This average force will be minimized.

2.5. Design variables

Not all variables in equations (10) and (11) are design variables of the robot balancing system. The balancing moments $M7P$ and $M5P$ are determined by equations (5a) and (6a) and for $l1T$ and $l2T$ (computed variables) we have the following relations (see Figs 3 and 4)

$$l1T = \sqrt{(lx1^2 + ly1^2)} + tm1 \tag{15}$$

$$l2T = \sqrt{(e2 + ly2)^2 + lx2^2} + tm2 \tag{16}$$

which determine appropriate lengths of the spring rods. In the example treated in Section 4 the following variables will be specified:

$$m5, m6, m7, m8, 15, 16, 17, 18, a, b, c, d, p, q, tm1, tm2. \tag{17}$$

For their values see Section 4.1.

This means, for the following design variables a minimum of (14) has to be found:

$$e1, l01, k1, lx1, ly1, \phi 10, e2, l02, k2, lx2, ly2, \phi 20 \tag{18}$$

For these variables search intervals have to be given in order to find an optimal design, see Section 4.2.

Remark. The variables $e1$ and $e2$ occur in the expressions for the forces Fx and Fy only in the combinations $e1.k1$, $e2.k2$ or $e1$/length and $e2$/length. This means that we can give $e1$ and $e2$ arbitrary values. If $e1$ or $e2$ is a factor f larger, then $k1$ and $k2$ will be a factor f smaller and this will give the same solution for the forces Fx and Fy. It means that we can specify the variables $e1$ and $e2$ and that in this case we have two independent design variables less and that will speed up the optimization program.

3. METHOD OF SOLUTION

The design of the robot in Fig. 1 has now been formulated as an optimization problem, we want a minimum of (14) and we want to know the values of the independent design variables (18). To find this minimum, we use existing optimization programs described in [6–8].

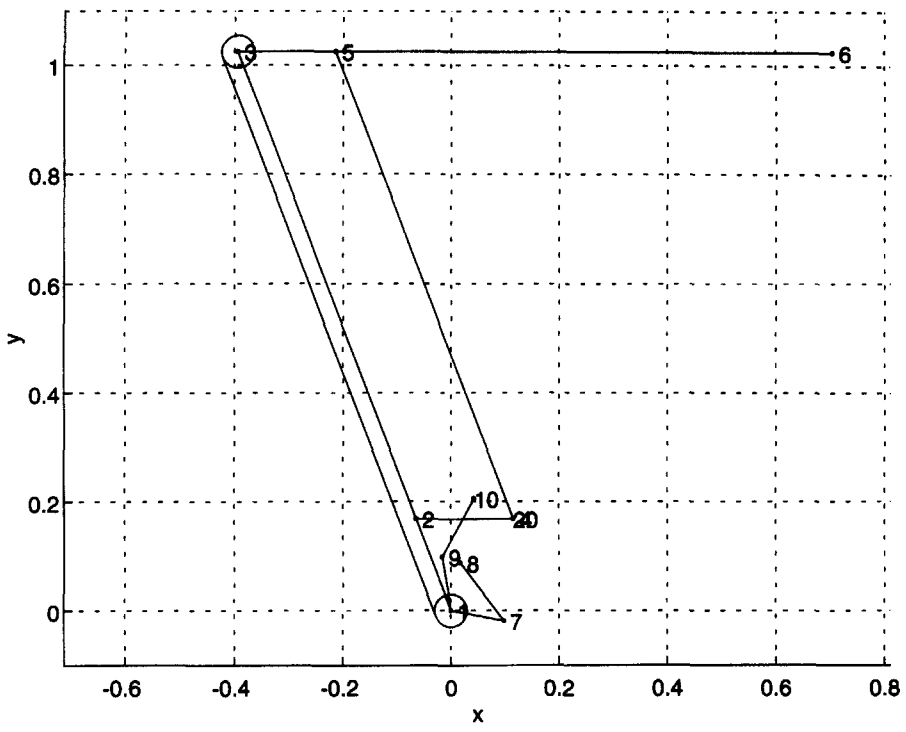

Fig. 6. Initial position of robot for case 1 as simulated by SPACAR.

3.1. Optimization program GOOD

It is clear that we have to find a global minimum for the function (14) for the variables (18). As we are not interested in a local extreme, all gradient methods are rejected. For a small number of variables a Monte Carlo method [6] with interactive interval reduction is very appropriate.

This method has been implemented by a Fortran program GOOD (Generator Of Optimal Designs). With this program that works in an interactive way the user is able to modify the search intervals for the independent variables manually during run time. With the aid of graphic functions it is shown on the screen where the subsequent suboptima are found, and the user can reduce the intervals accordingly. This way the convergence to an optimum can be speeded up. Recently the program has been extended to a genetic algorithm [8]. Using the same graphic interface, the user is able to start complete new populations in intervals where optima can be expected. It is found that "premature convergence" can be avoided this way. It appeared that for the number of independent variables (10) and for the time-consuming objective function we have here the genetic algorithm gave the fastest results.

3.2. Genetic algorithm

Genetic algorithms are extensively described in [9–11]. In order to apply a genetic algorithm, we have to "code" the designs. We will here only mention how the necessary coding for the robot designs takes place in the program used. As shown in the preceding section, a design for the robot is determined by 10 independent variables (18). The function to be optimized is written as $f = -fav$ (see 14), where f is called the fitness function, that has to be maximized. The independent variables are coded in binary strings. When we divide a search domain of a variable *f.i.* in 1024 intervals, we can code a value in a certain interval as a binary number between 0000000000 and 1111111111. For two different designs (selected with a preference for "better" designs) we can apply cross-over between the two bit strings for this variable. When we do this for all variables, we have created two "child designs". Mutation is applied with a certain mutation probability (can be modified

interactively during run time). In general, when we divide a search interval *si* into a power of two intervals, and the number of bits is *nb*, we can define a tolerance *tol* by:

$$tol = si/2^{nb} \tag{19}$$

The tolerance *tol* can be specified by the user for all variables involved. *nb* is then computed from (19). A so-called start population (of *f.i.* 100 robots) is randomly generated, and from this start population new populations are created in the manner just described. In GOOD the algorithm has been implemented in such a way that population size, mutation probability and search intervals can be modified during run time. The child designs tend to have better values of the fitness function (14), as explained in [9–11]. This way many suboptima are generated with increasingly better values of the fitness function (14). With the aid of the graphic interface of GOOD it is possible to find out in what regions of the search intervals the better designs are found, and to start a complete new population in those regions, see [8]. In the next sections results are given.

```
PLBEAM   1  1 11  2 12
PLBEAM   2  2 12  3 13
PLBEAM   3  3 14  5 15
PLBEAM   4  5 15  6 16
PLTRUSS  5  2  4
PLTRUSS  6  4  5
PLBEAM   7  1 17  7 18
PLTRUSS  8  7  8
PLBEAM   9  1 11  9 19
PLTRUSS 10  9 10
PLBELT  11  1 17  3 14  0.03 0.03
PLTERN  12  4 21 20     0.0
X 1  0          0
X 2 -0.0645024  0.168046
X 3 -0.393465   1.02508
X 4  0.115498   0.168046
X 5 -0.213465   1.02508
X 6  0.704535   1.02508
X 7  0.0982066 -0.0188539
X 8  0.0176586  0.0874153
X 9 -0.0161054  0.0986946
X 10 0.0425081  0.204759
X 20 0.115498   0.168046
FIX  1 1
FIX  1 2
FIX  8 1
FIX  8 2
FIX 10 1
FIX 10 2
FIX 21 1
RLSE  8 1
LENGTH0  8 -0.0116236
RLSE 10 1
LENGTH0 10 -0.0004555
INPUTX 20 1
INPUTX 20 2
END
ESTIFF  8 23158
ESTIFF 10 64957.7
XF 2 0    -4.0875
XF 5 0   -25.2547
XF 4 0   -31.0874
XF 3 0 -1016.74
XF 1 0  -304.539
XF 6 0  -193.682
END
```

Fig. 7. Inputfile for SPACAR with definition of the robot mechanism, case 1.

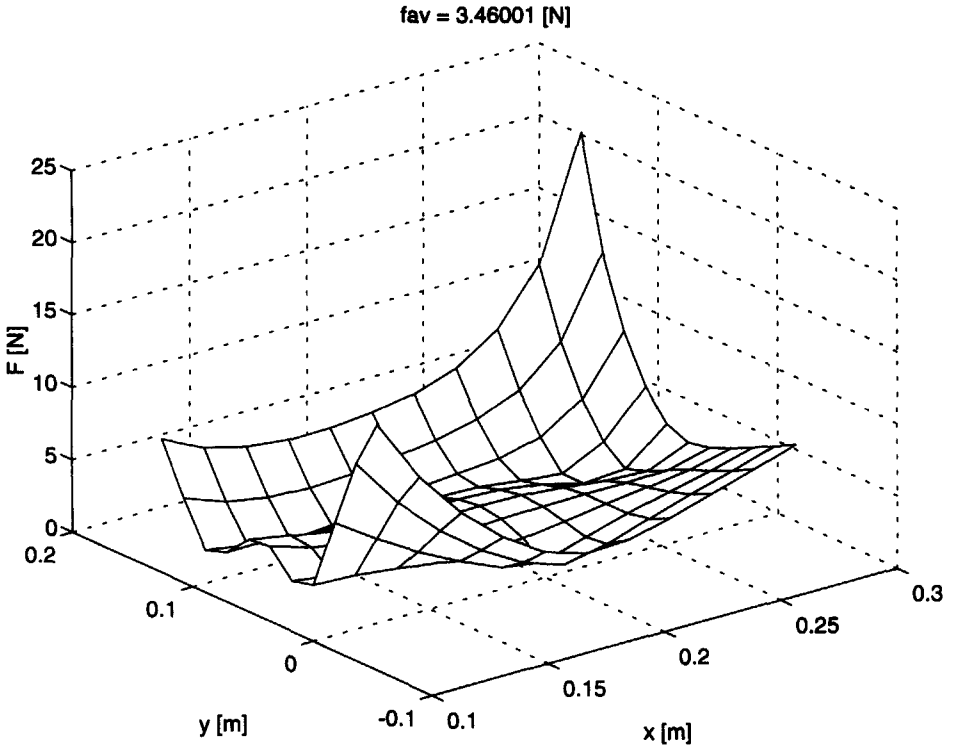

Fig. 8. Absolute value of driving forces, $F = \sqrt{Fx_2 + Fy_2}$, as function of position in input working area, case 1.

4. RESULTS

4.1. Specified variables

In the example treated the following variables have been specified:

$e1 = 0.1$ m, $e2 = 0.1$ m, $m5 = 57.0$ kg, $m6 = 1.0$ kg, $m7 = 97.43$ kg, $m8 = 5.16$ kg, $15 = 1.098$ m,

$l6 = 0.18$ m, $l7 = 1.098$ m, $l8 = 0.918$ m, $tm1 = 0.15$ m, $tm2 = 0.1$ m, $a = 0.18$ m,

$b = 0.18$ m, $c = 0.458$ m, $d = 0.2225$ m, $p = 0.5$ m, $q = 0.105$ m.

It is to be understood that these variables could just as well be considered as design variables. In this case we would have to specify intervals for them and they would be treated like the variables in Section 4.2.

4.2. Search intervals for design variables

In Table 1 the search intervals for all independent design variables are given.
This table has been used in the optimization program. A minimum for (14) is found in the given search intervals for the design variables.

Table 1. Search intervals

Variable	Units	Lower bound	Upper bound
$k1$	N/m	0	4,000,000
$l01$	m	0.15	0.4
$lx1$	m	−0.08	0.08
$ly1$	m	0.035	0.234
$\phi 10$	rad	−0.3491	0.3491
$k2$	N/m	0	600,000
102	m	0.1	0.45
$lx2$	m	−0.04	0.04
$ly2$	m	0.024	0.18
$\phi 20$	rad	2.7925	3.4906

Table 2. Output of optimization programs

Variable	Units	Case 1	Case 2	Case 3	Case 4
$k1$	N/m	64957.72	77470.14	60332.34	58951.69
$l01$	m	0.3595802	0.325919	0.3582281	0.3801751
$lx1$	m	−0.04250812	0.01328951	0.07142384	0.03219239
$ly1$	m	0.2047589	0.1753409	0.1961656	0.2276097
$\phi 10$	rad	0.2047383	−0.07568362	−0.3489188	−0.1374789
$k2$	N/m	23157.98	6885.527	9764.061	—
$l02$	m	0.299869	0.1778149	0.331819	—
$lx2$	m	−0.01765858	0.001941306	0.004234375	—
$ly2$	m	0.08741526	0.02669327	0.09313436	—
$\phi 20$	rad	3.331267	3.143471	3.061444	—
$l1T$	m	0.3591247	0.3258438	0.3587638	0.379875
$l2T$	m	0.2882454	0.2267082	0.2931808	—
fav	N	3.15	0.8545785	2.290451	46.09412

4.3. *Output of optimization programs*

The values found for the design and computed variables of the robot balancing system for a minimum of the object function (14) are given in Table 2. The specified variables are given in Section 4.1. Four cases have been optimized. The results of the first case characterize optimal balancing mechanisms of the existing robot APR 20 with $d = 0.2225$ m. For the second case we specified $d = 0$. It means that the centre of gravity $T7$ is at joint D (see Fig. 2). For the third case we specified the value of $m7 = 87.43$ kg (without the payload of 10 kg) and therefore we have $d = 0.12236$ m. We can conclude that the best result was gained for $d = 0.12236$ m. Comparing average forces fav for the three cases we can conclude that the best result was gained for $d = 0$. Of course, we could consider this specified variable as a design variable. The fourth case is characterized by the value of $d = 0$ however only the balancing mechanism for the link 5 was used. The result of optimization ($fav = 46.09412$ N) is much worse than in the previous three cases. It is caused by the fact that this balancing mechanism is not able to balance link 5 and also links 6 and 8 well.

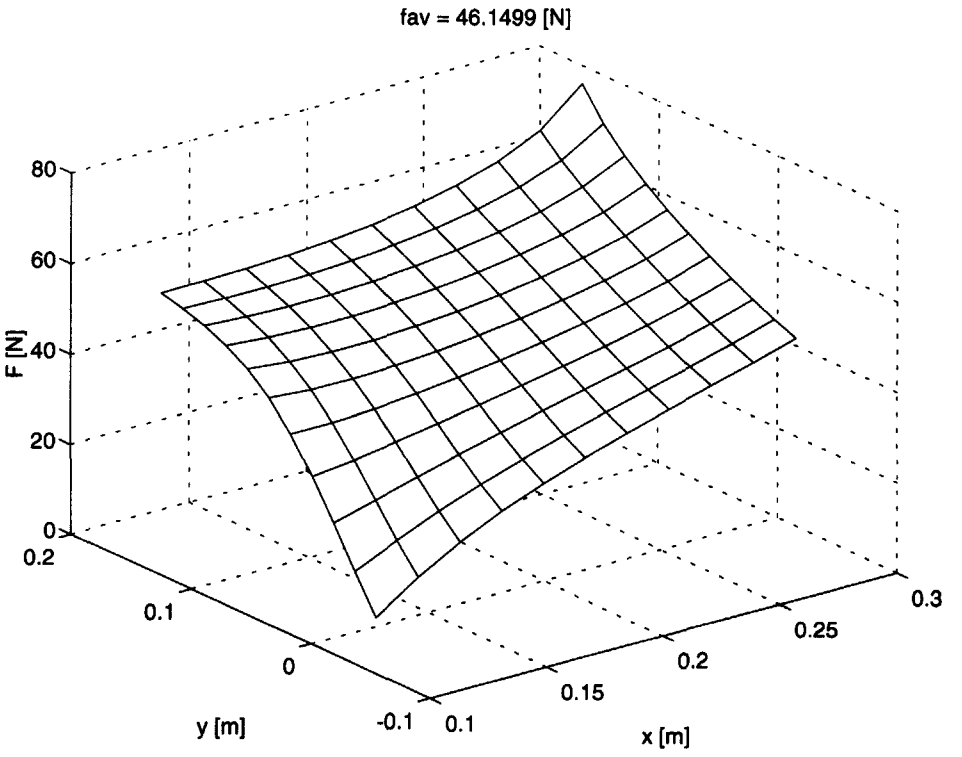

Fig. 9. Absolute value of driving forces, $F = \sqrt{Fx^2 + Fy^2}$ as function of position in input working area, case 4.

5. ANALYSIS OF RESULTS

To check the results of the optimization programs, the resulting robots are given as input to a general purpose multibody system dynamics computer program SPACAR [3]. This program needs as input the topology, geometry, mass and force distribution of an arbitrary multibody system, in this case a robot. The output of the program is the movement, position, velocity and acceleration, together with all forces for the given input movement. In the case of the robot a kinetostatic analysis was done, that is the inertia forces due to the movement were excluded. Since a number of cases with the same topology but with different geometry had to be analysed MATLAB [4] was used as a pre- and postprocessor. In fact, the SPACAR program, which is just a collection of FORTRAN subroutines, has been constructed as a subroutine to the MATLAB program with the aid of the MATLAB MEX utility. This way it was also possible to have a graphic animation of the geometry during the analysis. In Fig. 6 the initial position of the robot for case 1 is shown. For every case from Table 2 the variables are read into MATLAB. With a MATLAB script these variables are translated into a SPACAR specific input file. For case 1 this input file is shown in Fig. 7. Together with a description of the results to be stored after each integration step the SPACAR program is run from the MATLAB environment. The results, driving positions and forces, are plotted in MATLAB in a 3D graphics form. These figures give a good impression of the driving forces in the working area of the robot. Finally the average force is calculated and compared to the output *fav* (case 1 and case 4) from GOOD (Table 2).

Since the discretisation of the working area differs from GOOD, the *fav* resulting from SPACAR is not exactly the same, but an agreement of 95% is obtained. The results computed with SPACAR are shown in Figs 8 and 9.

6. CONCLUSION

It was shown that the problem of finding suitable design variables for balancing mechanisms of a robot can be formulated as an optimization problem that can easily be solved with the aid of a Monte Carlo with interactive interval reduction as described in [6] and [7]. A genetic algorithm gives even better and faster results. It is obvious that the method can be also used to optimize structures of robot mechanisms. It is probable that in the case of the so called "ASEA" robot mechanism, see for example [12], we would not need to use a mechanical transmission (which complicates the design of the robot) to balance the link 7 of Fig. 2. The advantage of the method described in this article compared to other methods, see for example [5] and [12], consists in the fact that this method is general and appropriate to solve problems in practice. It is to be expected that many other design problems of robots can be solved in the same manner.

Acknowledgement—The authors are indebted to P. J. Dekker for computing the expressions for *Fx* and *Fy* with the aid of Maple.

REFERENCES

1. van der Werff. K., Kalker-Kalkman, C. M. and Kempes, W., *Proceedings of the 9th World Congress on the Theory of Machines and Mechanisms*, Milano, Italia, 30 August–2 September, 1995.
2. Maple 5.3, University of Waterloo, Ontario, Canada.
3. Soest, J. van, Schwab, A. L. *et al.*, *Journal of Biomechanics*, 1992, **25**, 1219–1226.
4. *MatLab Users Guide*. The Mathworks, Natick, MA 1993.
5. Korendasev, A. I. (Ed.), *Manipulation Systems of Robots*. Mashinostroenie, Moscow, 1989.
6. Kalker-Kalkman, C. M., in *Engineering with Computers 7*, Springer-Verlag, New York, 1991, pp. 173–183.
7. Kalker-Kalkman, C. M., in *Advances in Computer-aided Engineering*. Delft University Press, 1994, ISBN 90-407-1017-1, pp. 61–70.
8. Kalker-Kalkman, C. M. and Offermans, M. F., *Proceedings of ASME Design Automation Conference*, Boston, September 1995.
9. Davis, L., *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.
10. Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.
11. Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution programs*. Springer-Verlag, New York, 1992.
12. Kolarski, M., Vukobratovic, M. and Borovac, B., *Mech. Mach. Theory* 1994, **29**(3), 427–454.