KYBERNETIKA — VOLUME 18 (1982), NUMBER 6

NEW ALGORITHM FOR MINIMAL SOLUTION OF LINEAR POLYNOMIAL EQUATIONS

JAN JEŽEK

In this paper, a new efficient algorithm is described for computing the minimal-degree solution of the linear polynomial equation. This problem occurs in linear control synthesis.

0. INTRODUCTION

The polynomial equation

$$a(\lambda)\,x(\lambda) + b(\lambda)\,y(\lambda) = c(\lambda) \tag{1}$$

plays central role in the linear control theory. It occurs in optimal control synthesis, in dead-beat as well as in quadratic criteria [1, 2, 3]. In this equation, $a(\lambda)$, $b(\lambda)$, $c(\lambda)$ are given polynomials, $x(\lambda)$, $y(\lambda)$ are unknown ones. We suppose $a(\lambda)$, $b(\lambda)$ are not both identically zero. Main tool for studying this Euclidean equation is the Euclidean algorithm. It yields general solution and can serve as a computing algorithm for it. In control theory applications, some particular solutions are of interest: those with degree of x or y (or both) minimal. Algorithms are known for selecting the minimal solution from the general one. In this paper, a new algorithm is described which computes the minimal solution directly. This way is more efficient than the general solution computation. The method is also based on the Euclidean algorithm.

The indeterminate λ can represent either the derivative operator s in the continuous control theory or the delay operator ζ in the discrete case. We shall also write a instead of $a(\lambda)$ for brevity. By ∂a we mean the degree of polynomial $a(\lambda)$, $\partial a = -\infty$ means zero polynomial $a(\lambda) \equiv 0$. The leading coefficient of $a(\lambda)$ is denoted by $a_{\partial a}$. By gcd (a, b) we mean the greatest common divisor of polynomials a, b.

1. PRELIMINARIES

Let us summarize the theory of polynomial equation briefly. Let us denote

$$d(\lambda) = \gcd\,(a(\lambda), b(\lambda)),\ a_0(\lambda) = \frac{a(\lambda)}{d(\lambda)},\quad b_0(\lambda) = \frac{b(\lambda)}{d(\lambda)}$$

Theorem 1. The general solution of the homogeneous equation

$$a(\lambda)\, x(\lambda) + b(\lambda)\, y(\lambda) = 0 \tag{2}$$

is

$$x(\lambda) = b_0(\lambda)\, t(\lambda)\,, \quad y(\lambda) = -a_0(\lambda)\, t(\lambda) \tag{3}$$

where $t(\lambda)$ is an arbitrary polynomial.

Proof – see [2, 3].

Theorem 2. The equation (1) is solvable iff $d(\lambda)$ divides $c(\lambda)$.

Proof – see [2, 3].

Theorem 3. The general solution of (1) is

$$x(\lambda) = x_p(\lambda) + b_0(\lambda)\, t(\lambda)\,, \quad y(\lambda) = y_p(\lambda) - a_0(\lambda)\, t(\lambda) \tag{4}$$

where $x_p(\lambda)$, $y_p(\lambda)$ is any particular solution, $t(\lambda)$ is an arbitrary polynomial.

Proof – follows from linearity of (1) and from Theorem 1.

Theorem 4. The equation (1) with $a(\lambda) \not\equiv 0$, if solvable, has the unique "y-minimal" solution $x(\lambda)$, $y(\lambda)$ defined by

$$\partial y < \partial a - \partial d \tag{5}$$

For this solution,

$$\partial x < \partial b - \partial d \quad \text{for} \quad \partial c < \partial a + \partial b - \partial d \tag{6a}$$

$$\partial x = \partial c - \partial a \quad \text{for} \quad \partial c \geqq \partial a + \partial b - \partial d \tag{6b}$$

Proof. Let $x_p(\lambda)$, $y_p(\lambda)$ be a particular solution of (1). As $a_0(\lambda) \not\equiv 0$, let us take the quotient $q(\lambda)$ and the remainder $y(\lambda)$:

$$y_p(\lambda) = a_0(\lambda)\, q(\lambda) + y(\lambda)\,, \quad \partial y < \partial a_0 = \partial a - \partial d$$

According to Theorem 3, the couple

$$y(\lambda) = y_p(\lambda) - a_0(\lambda)\, q(\lambda)\,, \quad x(\lambda) = x_p(\lambda) + b_0(\lambda)\, q(\lambda)$$

satisfies (1). Existence of y-minimal solution is proved.

Let us prove its uniqueness. Let x_1, y_1 and x_2, y_2 be two y-minimal solutions, i.e. $\partial y_1 < \partial a_0$, $\partial y_2 < \partial a_0$. The couple $x = x_1 - x_2$, $y = y_1 - y_2$ satisfies the homogeneous equation (2), $\partial y < \partial a_0$. But according to (3), for $y \not\equiv 0$ it must be $\partial y \geqq \partial a_0$. Therefore $y = 0$, $y_1 = y_2$. From (2), $x = 0$, $x_1 = x_2$.

Now we shall prove the property (6). The equation (1) yields by cancellation of $d(\lambda)$ an equivalent equation

$$a_0(\lambda)\, x(\lambda) + b_0(\lambda)\, y(\lambda) = c_0(\lambda) \tag{7}$$

where $c_0(\lambda) = c(\lambda)/d(\lambda)$. Let us consider the degrees of all three terms:

a) $c(\lambda) \equiv 0$, the y-minimal solution is $x(\lambda) = 0$, $y(\lambda) = 0$. For $b(\lambda) \not\equiv 0$ (6a) is satisfied, for $b(\lambda) \equiv 0$ (6b) is.

b) $\partial c < \partial a + \partial b - \partial d \Leftrightarrow \partial c_0 < \partial a_0 + \partial b_0$
$\partial(b_0 y) < \partial a_0 + \partial b_0 \Rightarrow \partial(a_0 x) < \partial a_0 + \partial b_0 \Rightarrow \partial x < \partial b_0 = \partial b - \partial d$, hence (6a) is satisfied.

c) $\partial c \geqq \partial a + \partial b - \partial d \Leftrightarrow \partial c_0 \geqq \partial a_0 + \partial b_0$
$\partial(b_0 y) < \partial a_0 + \partial b_0 \leqq \partial c_0 \Rightarrow \partial(a_0 x) = \partial c_0 \Rightarrow \partial x = \partial c_0 - \partial a_0 = \partial c - \partial a$, hence (6b) is satisfied. □

Theorem 5. The equation (1) with $b(\lambda) \not\equiv 0$, if solvable, has the unique "x-minimal" solution $x(\lambda)$, $y(\lambda)$ defined by

$$\partial x < \partial b - \partial d$$

For this solution,

$$\partial y < \partial a - \partial d \quad \text{for} \quad \partial c < \partial a + \partial b - \partial d$$

$$\partial y = \partial c - \partial b \quad \text{for} \quad \partial c \geqq \partial a + \partial b - \partial d$$

Proof. If we interchange in (1) a with b and x with y the equation remains invariant. Therefore, Theorem 5 follows from Theorem 4. □

Theorem 6. The equation (1) with $a(\lambda) \not\equiv 0$, $b(\lambda) \not\equiv 0$, $\partial c < \partial a + \partial b - \partial d$, if solvable, has the unique "minimal" solution x, y defined by one of the two equivalent conditions:

$$\partial x < \partial b - \partial d \quad \text{or} \quad \partial y < \partial a - \partial d$$

Proof. According to Theorem 4, the y-minimal solution exists uniquely. It satisfies (6a), i.e. it is at the same time x-minimal. □

2. THE ALGORITHM

We shall construct the algorithm as a recurrent process with finite number of steps. In every step, the equation is replaced by an equivalent one by a substitution. The final equation can be solved in a trivial way. By backward substitution chain, we shall obtain the solution of the original equation.

First, we shall prove four lemmas for the recurrent process.

Lemma 1. Let (1) be solvable, $a(\lambda) \not\equiv 0$, $\partial c \geqq \partial a$. Then the equation

$$a(\lambda)\, x'(\lambda) + b(\lambda)\, y(\lambda) = c'(\lambda) \tag{8}$$

where

$$c'(\lambda) = c(\lambda) - \frac{c_{\partial c}}{a_{\partial a}} \lambda^{\partial c - \partial a} a(\lambda)\,, \quad \partial c' < \partial c \tag{9}$$

is also solvable, has the y-minimal solution x', y and the y-minimal solution x, y of (1) is given by

$$x(\lambda) = x'(\lambda) + \frac{c_{\partial c}}{a_{\partial a}} \lambda^{\partial c - \partial a} \tag{10}$$

Proof. Let $d = \gcd(a, b)$. From (9), d divides c' if it divides c, i.e. (8) is solvable iff (1) is. As $a(\lambda) \not\equiv 0$, the x', y exists. By substituting (10) into (1) we can see that x, y is a solution. As x', y is y-minimal for (8), $\partial y < \partial a - \partial d$. Therefore x, y is y-minimal for (1). □

Lemma 2. Let (1) be solvable, $b(\lambda) \not\equiv 0$, $\partial c \geqq \partial b$. Then the equation

$$a(\lambda)\, x(\lambda) + b(\lambda)\, y'(\lambda) = c'(\lambda) \tag{11}$$

where

$$c'(\lambda) = c(\lambda) - \frac{c_{\partial c}}{b_{\partial b}} \lambda^{\partial c - \partial b}\, b(\lambda)\,, \quad \partial c' < \partial c \tag{12}$$

is also solvable, has the x-minimal solution x, y' and the x-minimal solution x, y of (1) is given by

$$y(\lambda) = y'(\lambda) + \frac{c_{\partial c}}{b_{\partial b}} \lambda^{\partial c - \partial b} \tag{13}$$

Proof. By interchanging a with b and x with y in Lemma 1. □

Lemma 3. Let (1) be solvable, $a(\lambda) \not\equiv 0$, $\partial c < \partial a + \partial b - \partial d$, $\partial b \geqq \partial a$. Then the equation

$$a(\lambda)\, x'(\lambda) + b'(\lambda)\, y(\lambda) = c(\lambda) \tag{14}$$

where

$$b'(\lambda) = b(\lambda) - \frac{b_{\partial b}}{a_{\partial a}} \lambda^{\partial b - \partial a}\, a(\lambda)\,, \quad \partial b' < \partial b \tag{15}$$

is also solvable, has the y-minimal solution x', y and the minimal solution x, y of (1) is given by

$$x(\lambda) = x'(\lambda) - \frac{b_{\partial b}}{a_{\partial a}} \lambda^{\partial b - \partial a}\, y(\lambda) \tag{16}$$

Proof. Let $d = \gcd(a, b)$, $d' = \gcd(a, b')$. From (15), d divides $b' \Rightarrow d$ divides d'. By the same argument, d' divides $b \Rightarrow d'$ divides d. Therefore $d' = d$, i.e. (14) is solvable iff (1) is. As $a(\lambda) \not\equiv 0$, the x', y exists. By substituting (16) into (1) we can see that x, y is a solution. As x', y is y-minimal for (16), $\partial y < \partial a - \partial d$. Therefore x, y is y-minimal for (1). As $\partial c < \partial a + \partial b - \partial d$, it is minimal. $\square$

Lemma 4. Let (1) be solvable, $b(\lambda) \not\equiv 0$, $\partial c < \partial a + \partial b - \partial d$, $\partial a \geqq \partial b$. Then the equation

$$a'(\lambda)\, x(\lambda) + b(\lambda)\, y'(\lambda) = c(\lambda) \tag{17}$$

where

$$a'(\lambda) = a(\lambda) - \frac{a_{\partial a}}{b_{\partial b}} \lambda^{\partial a - \partial b} b(\lambda)\,, \quad \partial a' < \partial a \tag{18}$$

is also solvable, has the x-minimal solution x, y' and the minimal solution x, y of (1) is given by

$$y(\lambda) = y'(\lambda) - \frac{a_{\partial a}}{b_{\partial b}} \lambda^{\partial a - \partial b} x(\lambda) \tag{19}$$

Proof. By interchanging a with b and x with y in Lemma 3. $\square$

Now we are ready to construct the algorithm.

Theorem 7. Let a, b, c be given polynomials in (1). Let p be a two-valued variable with values p_x and p_y; here p_x means "the x-minimal solution is to be found", analogously p_y. We suppose the equation is solvable and $a \not\equiv 0$ for $p = p_x$, $b \not\equiv 0$ for $p = p_y$.

Let us construct a sequence $a^{(i)}, b^{(i)}, c^{(i)}, p^{(i)}$, $i = 0, 1, 2 \dots$ with initial values $a^{(0)} = a$, $b^{(0)} = b$, $c^{(0)} = c$, $p^{(0)} = p$ by the following recurrent rule:

If

$$a^{(i)} \equiv 0\,, \quad b^{(i)} \not\equiv 0\,, \quad c^{(i)} \equiv 0\,, \quad p^{(i)} = p_x \tag{20a}$$

or

$$a^{(i)} \not\equiv 0\,, \quad b^{(i)} \equiv 0\,, \quad c^{(i)} \equiv 0\,, \quad p^{(i)} = p_y \tag{20b}$$

then stop, it is the last element of the sequence. Otherwise, if $p^{(i)} = p_x$ we distinguish two cases:

1) $\partial c^{(i)} \geqq \partial b^{(i)}$. it is always $b^{(i)} \not\equiv 0$. We use Lemma 2 and define $c^{(i+1)}$ by (12), $p^{(i+1)} = p^{(i)}$.

2) $\partial c^{(i)} < \partial b^{(i)} \Rightarrow \partial c^{(i)} < \partial a^{(i)} + \partial b^{(i)} - \partial d^{(i)}$, we distinguish two subcases:

   2a) $\partial b^{(i)} > \partial a^{(i)}$, for $c^{(i)} \not\equiv 0$ it is always $a^{(i)} \not\equiv 0$ (case $c^{(i)} \equiv 0$, $a^{(i)} \equiv 0$ see (20)). We use Lemma 3 and define $b^{(i+1)}$ by (15), $p^{(i+1)} = p_y$.

   2b) $\partial b^{(i)} \leqq \partial a^{(i)}$, it is always $b^{(i)} \not\equiv 0$. We use Lemma 4 and define $a^{(i+1)}$ by (18), $p^{(i+1)} = p_x$.

If $p^{(i)} = p_y$ we analogously have two cases 3) and 4) which are literally the same as 1) and 2) up to interchanging a with b and x with y.

Then for every i up to some final value i_F the rule is properly defined and $a^{(i)}$, $b^{(i)}$, $c^{(i)}$, $p^{(i)}$ represent the problem of finding $x^{(i)}$-minimal or $y^{(i)}$-minimal solution of equation

$$(21) \qquad a^{(i)}(\lambda)\, x^{(i)}(\lambda) + b^{(i)}(\lambda)\, y^{(i)}(\lambda) = c^{(i)}(\lambda)$$

equivalent to (1). For the final value i_F the state (20) is reached whose solution is $x = 0$, $y = 0$.

Proof. We must prove that the rule is properly defined, i.e. the cases

$$1) \qquad p^{(i)} = p_x\,, \quad \partial c^{(i)} \geqq \partial b^{(i)}\,, \quad b^{(i)} \equiv 0$$

$$2a) \qquad p^{(i)} = p_x\,, \quad \partial c^{(i)} < \partial b^{(i)}\,, \quad c^{(i)} \not\equiv 0\,, \quad a^{(i)} \equiv 0$$

$$2b) \qquad p^{(i)} = p_x\,, \quad \partial c^{(i)} < \partial b^{(i)}\,, \quad b^{(i)} \equiv 0$$

3), 4a), 4b) similarly

can never happen and that $a^{(i)}$, $b^{(i)}$, $c^{(i)}$, $p^{(i)}$ really represent the problem (21), equivalent to (1). We shall do it by induction. For $i = 0$, cases 1) and 2b) are excluded by assumptions of the theorem. In case 2a) the equation is $b(\lambda)\, y(\lambda) = c(\lambda)$. To be solvable, it must be either $\partial b \leqq \partial c$ or $c \equiv 0$, so 2a) is also excluded.

Let for some i the rule be properly defined and (21) equivalent to (1). Then $a^{(i+1)}$, $b^{(i+1)}$, $c^{(i+1)}$, $p^{(i+1)}$ as defined by the rule, always lead to an equivalent equation, solvable and never violating the condition $a^{(i)} \not\equiv 0$ for $p^{(i)} = p_y$ or $b^{(i)} \not\equiv 0$ for $p^{(i)} = p_x$. So the next step is also properly defined and the induction is completed.

Now we shall prove that for some finite i we must reach (20a) or (20b). Let us suppose that (20) is not satisfied for $i = 0$, the case $a \equiv 0$, $b \equiv 0$ is excluded. We shall prove that for some i one of polynomials $a^{(i)}$, $b^{(i)}$, $c^{(i)}$ is identically zero. Let us suppose it is not true for $i = 0$. In every step, just one of three polynomials $a^{(i)}$, $b^{(i)}$, $c^{(i)}$ is changed, either $\partial a^{(i+1)} < \partial a^{(i)}$ or $\partial b^{(i+1)} < \partial b^{(i)}$ or $\partial c^{(i+1)} < \partial c^{(i)}$ is valid. Therefore $[(\partial a + 1)(\partial b + 1)(\partial c + 1)]^{(i+1)} < [(\partial a + 1)(\partial b + 1)(\partial c + 1)]^{(i)}$ is also valid. Having started from a finite positive value, this integer function must reach zero for some i, i.e. some of the $a^{(i)}$, $b^{(i)}$, $c^{(i)}$ is zero.

If $c^{(i)}$ is the case then the sequence continues but only steps 2a) or 2b) are possible. Now $[(\partial a + 1)(\partial b + 1)]^{(i+1)} < [(\partial a + 1)(\partial b + 1)]^{(i)}$ and for some i, $a^{(i)} \equiv 0$, $p^{(i)} = p_x$ (the last step was 2b) or $b^{(i)} = 0$, $p^{(i)} = p_y$ (2a).

If $a^{(i)}$ was the case of zero polynomial then $p^{(i)} = p_x$, the equation is $b^{(i)}(\lambda)\, y^{(i)}(\lambda) = c^{(i)}(\lambda)$, solvable as proved, i.e. $\partial c^{(i)} \geqq \partial b^{(i)}$ or $c^{(i)} \equiv 0$. The sequence continues with steps 1) only possible. For some i we must reach $\partial c^{(i)} < \partial b^{(i)}$ i.e. $c^{(i)} \equiv 0$. Similarly in the case of zero $b^{(i)}$. □

3. COMPUTATIONAL ASPECTS

The construction just described can serve as an efficient algorithm for a computer or for a paper-and-pencil work. It has the property that neither $d(\lambda)$ nor ∂d must be known in advance. They are obtained as a result: in (20a), there is $b^{(i)}(\lambda) = d(\lambda)$, similarly in (20b) $a^{(i)}(\lambda) = d(\lambda)$. The case of unsolvable equation ($d(\lambda)$ does not divide $c(\lambda)$) is recognized by reaching the state $a^{(i)} \equiv 0$, $b^{(i)} \not\equiv 0$, $c^{(i)} \not\equiv 0$, $\partial c^{(i)} < \partial b^{(i)}$, $p^{(i)} = p_x$ or $a^{(i)} \not\equiv 0$, $b^{(i)} \equiv 0$, $c^{(i)} \not\equiv 0$, $\partial c^{(i)} < \partial a^{(i)}$, $p^{(i)} = p_y$.

The algorithm consists of a forward run and backward one. In the latter one, all "planned" substitutions are performed in backward order, from $x^{(i_F)}$, $y^{(i_F)}$ to $x^{(0)}$, $y^{(0)}$.

In every step of the forward run, some of the polynomials a, b, c are "reduced" by some other one. It is useful to have a subroutine for such a reduction, say, of $r(\lambda)$ by $s(\lambda)$:

$$\partial r \geqq \partial s\,, \quad r'(\lambda) = r(\lambda) - \frac{r_{\partial r}}{s_{\partial s}}\,\lambda^{\partial r - \partial s}\, s(\lambda)$$

i.e. from $r(\lambda)$ subtract $s(\lambda)$ multiplied by a number q and by λ^k such that the leading term gets cancelled. The real number $q = r_{\partial r}/s_{\partial s}$ and the integer number $k = \partial r - \partial s$ are also results of the operation. The reduction can be done "in place" with no new memory for r' needed.

In the backward run where (16) or (19) is applied, it is useful to have a subroutine for "forced reduction" of $r(\lambda)$ by $s(\lambda)$. It means: for given polynomials $r(\lambda)$, $s(\lambda)$, given real q and integer $k \geqq 0$, compute

$$r'(\lambda) = r(\lambda) - q\lambda^k\, s(\lambda)$$

i.e. from $r(\lambda)$ subtract $s(\lambda)$ multiplied by a prescribed number and a prescribed power of λ. It can also be done in place.

The operation (10) and (13) means: given a polynomial $r(\lambda)$, given number q and integer k, add $q\lambda^k$ to $r(\lambda)$. It can also be done in place and it is useful to have a subroutine or a standard way for doing it.

During the forward run, we must keep track of three things:

a) the information on which of the four reduction types (9), (12), (15), (18) was used,

b) the number q by which the polynomial was multiplied,

c) the integer k by which the polynomial was shifted.

These data will be used in the backward run for selecting the proper substitution and for doing it. The data are stored and fetched by the stack discipline (first in, last out). If your computer and programming language support recursive procedures then they can be utilized for that, otherwise you must program the stack explicitly in arrays.

Flowcharts of the forward and backward runs are in Fig. 1, 2. The four-valued variable t stores the information on the substitution type. Its values are mnemonic names for substitutions. Stack manipulation is done by operations "store" and "fetch".

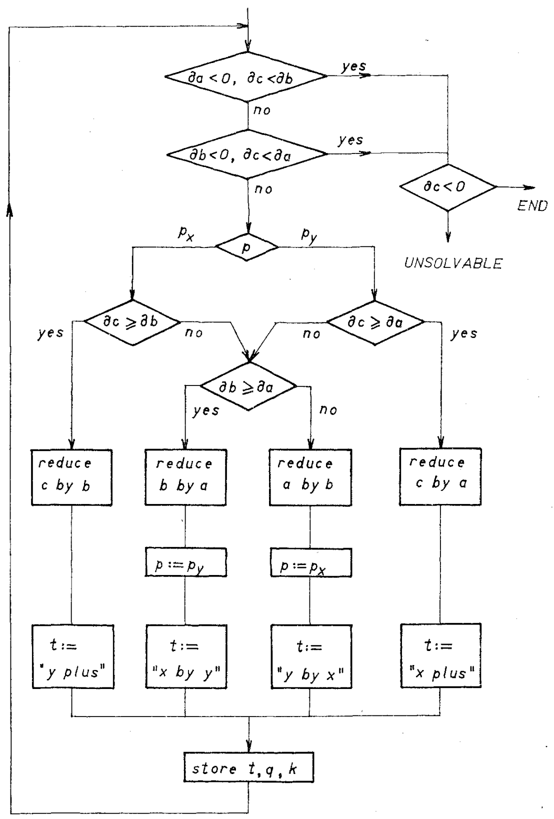

Fig. 1. Flowchart of the forward run.

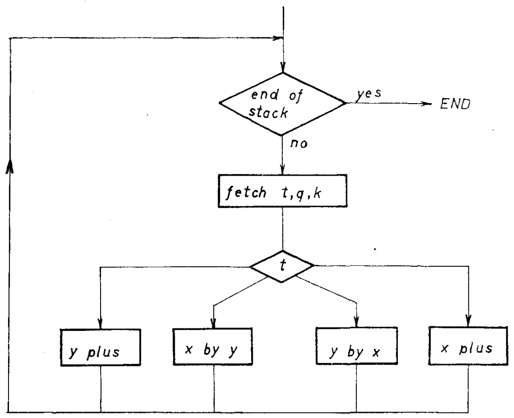

Fig. 2. Flowchart of the backward run.

4. EXAMPLES

The algorithm was implemented on IBM 370 computer in FORTRAN language. Here presented are four extremely simple examples which can be easily computed with paper and pencil. In Tables 1, 2 the case $a(\lambda) = 1 + 3\lambda + 2\lambda^2$, $b(\lambda) = 1 + 2\lambda$, $c(\lambda) = \lambda + 2\lambda^2$ is solved for the y-minimal solution $x(\lambda) = 1$, $y(\lambda) = -1$ and for the x-minimal solution $x(\lambda) = 0$, $y(\lambda) = \lambda$. The polynomials a, b, c have the common factor $d(\lambda) = 1 + 2\lambda$. After cancelling it, we have a new equation $a(\lambda) = 1 + \lambda$, $b(\lambda) = 1$, $c(\lambda) = \lambda$ with the same solution. This is solved in Tables 3, 4.

Table 1. Equation with a common factor, y-minimal solution

a	b	c	p	t	$q\lambda^k$	t	x	y
$1 + 3\lambda + 2\lambda^2$	$1 + 2\lambda$	$\lambda + 2\lambda^2$	p_y				1	−1
				c by a	1	x plus		
$1 + 3\lambda + 2\lambda^2$	$1 + 2\lambda$	$-1 - 2\lambda$	p_y				0	−1
				a by b	λ	y by x		
$1 + 2\lambda$	$1 + 2\lambda$	$-1 - 2\lambda$	p_x				0	−1
				c by b	−1	y plus		
$1 + 2\lambda$	$1 + 2\lambda$	0	p_x				0	0
				b by a	1	x by y		
$1 + 2\lambda$	0	0	p_y				0	0

Table 2. Equation with a common factor, x-minimal solution

a	b	c	p	t	$q\lambda^k$	t	x	y
$1+3\lambda+2\lambda^2$	$1+2\lambda$	$\lambda+2\lambda^2$	p_x				0	λ
				c by b	λ	y plus		
$1+3\lambda+2\lambda^2$	$1+2\lambda$	0	p_x				0	0
				a by b	λ	y by x		
$1+2\lambda$	$1+2\lambda$	0	p_x				0	0
				b by a	1	x by y		
$1+2\lambda$	0	0	p_y				0	0

Table 3. Equation without common factor, y-minimal solution

a	b	c	p	t	$q\lambda^k$	t	x	y
$1+\lambda$	1	λ	p_y				1	-1
				c by a	1	x plus		
$1+\lambda$	1	-1	p_y				0	-1
				a by b	λ	y by x		
1	1	-1	p_x				0	-1
				c by b	-1	y plus		
1	1	0	p_x				0	0
				b by a	1	x by y		
1	0	0	p_y				0	0

Table 4. Equation without common factor, x-minimal solution

a	b	c	p	t	$q\lambda^k$	t	x	y
$1+\lambda$	1	λ	p_x				0	λ
				c by b	λ	y plus		
$1+\lambda$	1	0	p_x				0	0
				a by b	λ	y by x		
1	1	0	p_x				0	0
				b by a	1	x by y		
1	0	0	p_y				0	0

5. COMPUTATIONAL COMPLEXITY

Let us estimate the number of numerical operations and the amount of storage needed for the above method and compare them with those for other methods. A suitable unit of computational complexity is an operation of "reduction": subtracting a multiple of a number from (or adding to) another number. For simplicity, let $\partial a = \partial b = \partial c = N$, $\partial d = 0$. The complexity functions $f(N)$, $g(N)$ for number of operations and number of storage places depending on N are polynomials, e.g. $f(N) = \alpha + \beta N + \gamma N^2$. Because mainly the behavior of $f(N)$ for great N is interesting, we neglect all but the leading term γN^2. We do this systematically in all stages during complexity function calculation, e.g. we replace $N + 1$ by N; it simplifies the job significantly. Three methods are investigated:

1) The above method. In the forward run, ∂a, ∂b, ∂c are lowered from N to 0 with $3 \sum_{n=0}^{N} n = \frac{3}{2}N^2$ reductions needed. In the backward run, ∂x and ∂y grow from 0 to N with $2 \sum_{n=0}^{N} n = N^2$ operations needed. Total $\frac{5}{2}N^2$ operations. Auxiliary storage needed is N reals, N integers and N four-valued codewords.

2) The general solution method [2, 3]. It computes polynomials p, q, r, s satisfying

$$P \, . \, A = \begin{bmatrix} p & q \\ r & s \end{bmatrix} . \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The matrix P represents accumulated row operations on the vector A. The degrees ∂p, ∂q, ∂r, ∂s grow from 0 to N, the degrees ∂a, ∂b fall from N to 0 with $6 \sum_{n=0}^{N} n = 3N^2$ reductions needed. Thenafter the particular solution $x_0 = pc$, $y_0 = qc$ is recomputed to the minimal one by taking the remainder in division by b resp. by a. This multiplication and division needs $4N^2$ operations. Total $7N^2$ operations is worse than that of the above method by factor 2·8. Auxiliary storage $4N$ for p, q, r, s is also greater.

3) The probable most wide-spread method of indeterminate coefficients. By it, the polynomial equation is replaced by a set of $M = 2N$ numerical equations for coefficients. The number of operations for solving by elimination is known to be $M^3/3 = \frac{8}{3}N^3$, the storage $M^2 = 4N^2$. These formulae show that this method compared with any polynomial one is worse by an order of magnitude: N^3 instead of N^2 operations, N^2 instead of N storage.

6. CONCLUSION

The method is very simple, effective and easy to learn. Number of operations needed is less than that in any other method known.

ACKNOWLEDGEMENT

The author wishes to thank Ing. V. Kučera DrSc. and prof. Ing. V. Strejc DrSc. for valuable comments on the algorithm.

(Received March 17, 1982.)

REFERENCES

[1] V. Kučera: Algebraic theory of discrete optimal control for single-variable systems I — Preliminaries. Kybernetika *9* (1973), 2, 94—107.

[2] V. Kučera: Algebraická teorie diskrétního lineárního řízení. (Algebraic Theory of Discrete Linear Control). Academia, Praha 1978.

[3] V. Kučera: Discrete Linear Control — The Polynomial Equation Approach. Wiley, New York 1979.

Ing. Jan Ježek CSc., Ústav teorie informace a automatizace ČSAV (Institute of Information Theory and Automation — Czechoslovak Academy of Sciences), Pod vodárenskou věží 4, 182 08 Praha 8. Czechoslovakia.