ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XII, 2 (1971)

J. KUCHARCZYK (Wrocław) and J. WĘGIERSKI (Katowice)

CONGESTION STUDIES IN RAILWAY STATIONS AND YARDS BY DIGITAL SIMULATION (II)

0. Summary. The paper is a continuation of a previous one [1] and presents the model and program designed for simulation of operations in the departure part of a classification yard. Simulation results are given as an example.

1. Simulation model. We assume that the reader is familiar with paper [1], where we presented simulation models for operational studies of passenger stations and reception parts of classification yards. Our belief was that the passenger station model with appropriate interarrival and service distributions would be also adequate for the departure part of classification yards. We knew from statistics that car groups arrived at the departure tracks in intervals distributed according to the Erlang distribution with $k = 2$. Their service time was normally distributed, as showed observations carried out for that purpose especially. Trains, being ready for departure, do not, however, leave the station immediately because the main line may be occupied by other (priority) trains; therefore the outgoing trains have to wait in the departure tracks some time for the possibility of departure. This waiting time depends upon many factors, such as e.g. the traffic intensity of priority trains in every of the departure directions. No theoretical argumentation was available to describe the distribution of this waiting time. Operations statistics showed that this distribution was different for various stations, which is intuitively clear and obvious. We decided therefore to extend our passenger station simulation model to enclose sufficient details which would describe the relevant aspects of this situation.

The developed model is schematically shown in Fig. 1 of [1]. Incoming car groups (called further trains, too) occupy free departure tracks (service channels of type 1) or wait for free ones, i.e. form a queue. Since trains will leave the station in various directions, a departure direction is assigned to every of them at entry in accordance with a known frequency distri-

bution. Their service begins immediately and its duration is independent of channel number. Afterwards they are ready to leave the station. Every train waiting to leave the station must occupy the main line for some time, provided there is no movement of priority trains on that line. This fact is represented in our model by service channels of type 2. They are occupied by both priority trains using the main line and trains waiting to leave the classification yard. Service time is constant but different for both kinds of trains and is equal to the time necessary for a train to pass the relevant track block interval on the line. Trains using the main line passing by the classification yard (called priority trains) are treated as customers with pre-emptive priority. The reason for doing so is that freight trains leaving the classification yard should not occupy the main line when their movement would hinder that of priority trains; they may leave the departure track only when a sufficient time lag between them and the next priority train exists. In the model there are as many service channels of type 2 as there are departure directions. Every of those service channels has its own characteristics, i.e. service time, priority train intensity, pre-emptive time, etc. Among them there may be such ones which have no priority trains at all, which means that there may exist such departure directions where only trains leaving the classification yard exist.

This model appeared — by simulation results — to be in agreement with reality.

2. Simulation program. The program which realizes the operations of the model described in section 1 is given as an Algol procedure in the Appendix. We shall describe now the relevant notation and the main aspects of its working.

The procedure uses several standard and declared variables and procedures which will be described first. The **integer** type standard variable *time* is used as a measure of the time of performance of the simulation on the computer, the computer being Odra 1204. The value of this variable is increased by one every second internally in the computer; therefore it is set to zero at the very beginning of the program. Used are also the standard output procedures *outinteger* (k, p), *outreal* (k, p), *outstring* (k, p), *outline* (k, n) and *outspace* (k, n), where k denotes the output channel ($k = 0$ denotes the teleprinter coupled with the operating system of the computer, $k = 1$ denotes the punch or printer), p — the relevant parameter to be output, and n — a non-negative integer. The first two of the procedures output numbers in integer and floating-point form, respectively, the third one outputs strings, and the last two procedures output newlines and spaces, respectively. The program uses also four

non-standard functions *unif*, *erlang* (*k*, *lam*), *entry*, and *obsl* which must be declared as **real** procedures in the master program. Procedure *unif* provides uniformly distributed random numbers from the interval (0, 1), procedure *erlang* (*k*, *lam*) supplies random numbers with the Erlang distribution and parameters *k* and *lam*; procedure *entry* calculates interarrival intervals with the Erlang distribution having the parameters *kw* and *parw* plus some constant minimum time lag *przw*, whereas procedure *obsl* gives normally distributed service (of type 1) times with mean *ex* and standard deviation *sigma*. The relevant random number generators have been described in [1].

The master program which incorporates the simulation procedure *sygrod* has to fix (or read in) all parameters of the simulation and obtains through the integer *bbb* information about the state of simulation when the computer leaves *sygrod*. All parameters of *sygrod*, 18 in number, will be described now.

The integers *lpr* and *lsym* denote, respectively, the numbers of trial and registered simulations of train departures from the system. *dp* is the length of the interval in which the relevant operational statistics will be memorized. *lpkmax* denotes the maximum permissible number of waiting trains before the entry; if the queue length exceeds that number, the procedure is left with *bbb* = 3 indicating overcongestion. The integer *m* equals the number of departure tracks in the system, *kw*, *parw* and *przw* denote the parameters of the interarrival distribution (see the earlier description of the procedure *entry*), whereas *ex* and *sigma* form the parameters of the service-time distribution (see procedure *obsl*). The number of departure directions is denoted by *likier*. The array *p*[*1*: *likier*] contains the cumulative probability distribution of train departures in the directions 1 to *likier*. Every departure direction uses also 5 numbers from the arrays *czpt*, *pup*, *czod*, *socz*, and *odb*, the notation being the following. The freight train service time for the channel of type 2 in direction *i* is given by *czpt*[*i*]; it is the time a leaving freight train occupies the track-block interval being adjacent to the classification yard. *pup*[*i*] is a Boolean variable with the value **true** if there are priority trains on direction *i*; if so, the variables *czod*[*i*], *socz*[*i*], and *odb*[*i*] denote the priority train service time for channel of type 2 (i.e. the track-block occupation time of a priority train), the mean of the priority train interarrival distribution, and the pre-emptive time of a freight train leaving the station (i.e. the minimum time a freight train must precede a priority train) in direction *i*, respectively.

The label *pocz* denotes that part of the procedure which initializes the simulation. At label *zmst* begins the decision part where one of the main program parts, labelled *przp*, *wyj* and *w9* is selected for entry. The

program part following *przp* simulates the input of a new customer, i.e. the arrival of a new train, that following *wyj* finds the really possible departure moment of a train whose service (of type 1) has just been finished, and that following *w9* registers the fact of an actual train departure.

As an Algol program constitutes a clear scheme of the computation, we will refer the reader wanting to study the mode of the simulation to the Appendix and give here only the relevant notation used there. The arrays *licz* and *czas* incorporate the number of times and the sum of time intervals the system had been in a given state. The arrays *ccw*, *cob*, *czt*, and *coo* of dimension 0 to 30 each contain the distribution (registered in intervals of length *dp*) of the entry waiting times, the service duration times, the departure track occupancy times, and the departure waiting times, respectively. These are needed to illustrate the relevant distributions approximately. For every departure direction *i*, *lok*[*i*] counts the number of train departures, *zt*[*i*] — the number of immediate train departures (without any waiting for departure), and *sk*[*i*], *s2k*[*i*] and *s3k*[*i*] — the sums of the interdeparture times, their squares and cubes, respectively; they characterize the interdeparture time distribution approximately. For any train leaving the station, without distinction of departure direction, the variables *lokr*, *skr*, *s2kr* and *s3kr* serve the same purpose. The number of trains which have waited before entry in the system is counted by *lzp*. The state of the system and the number of train entries at the end of the trial period of simulation are remembered in *stanp* and *npp*, respectively, whereas *tuniv* denotes the simulation time moment the trial ended. *zeg* denotes the actual simulation time moment, *stan* — the state of the system (i.e. the number of customers waiting and in the departure tracks), *np* — the number of train entries up to that moment. All other notation may be looked up in the Appendix; the description of it would be boring.

The simulation has come to a "good end" if the procedure *sygrod* is left with *bbb* = 0. All other values of *bbb* (*bbb* = 1, 2, 3) are alarm indications. In any case a printout of the simulation results, beginning at label *exit*, takes place; thus, no information is lost, and the cause of exit from *sygrod* is indicated through *bbb*.

3. Examples of simulation results. The simulation procedure *sygrod* has been used several times for different purposes. Examples will be given now.

From a certain classification yard with 9 departure tracks freight trains may depart in two directions: 90 % of the trains exit into a main line where the mean time interval between priority trains equals 23.7 min.;

the remaining trains use a freight branch line where no priority traffic exists. The remaining traffic parameters of the yard are (in min.): *parw* = 38, *ex* = 115, *sigma* = 23. The operation of the yard has been simulated for the present traffic conditions. Thereafter the traffic intensity of the priority trains on the main line was increased to see to what extend this would not significantly influence the yard operations. Table 1 summarizes the simulation results.

TABLE 1

Main line traffic intensity characterized by *socz* [1]	23.7	15.0	10.0	5.0
Probability of departure track occupancy:				
at most 3	0.622	0.610	0.549	0.192
at most 6	0.991	0.989	0.974	0.533
at most 9	1.000	1.000	1.000	0.816
more than 9	0.000	0.000	0.000	0.184
Departure waiting time (in min.):				
mean	1.4	2.9	9.5	101.4
standard deviation	4.4	7.4	18.1	91.8
Probability of departure without delay	0.671	0.584	0.417	0.039

In another classification yard simulated, the traffic conditions were as follows. Two departure directions with 90 % of the trains leaving into a freight line with no priority trains and 10 % of the trains departing into a line with priority trains whose interarrival time distribution has *socz*[*2*] = 15 min. The other parameters: *m* = 8, *parw* = 30 min., *ex* = 120 min., *sigma* = 25 min. Table 2 summarizes the simulation results obtained by increasing the priority train traffic intensity.

TABLE 2

Priority train traffic intensity (*socz* [*2*])	15.0	10.0	7.5
Probability of departure track occupancy:			
at most 4	0.655	0.616	0.615
at most 8	0.997	0.995	0.993
more than 8	0.003	0.005	0.007
Departure waiting time (in min.):			
mean	1.3	1.7	4.5
standard deviation	6.9	10.1	24.0
Probability of departure without delay	0.523	0.439	0.205

From a given classification yard trains are departing in three directions, as follows:

direction i	per cent of trains	priority train traffic *socz*[i] (in min.)
1	40	30
2	50	no priority trains
3	10	45

In the present situation, however, all trains depart really in one direction where immediately after the classification yard a group of transit tracks may accumulate departing trains, if necessary, before they enter the appropriate railway line. Since the whole yard is working under light traffic conditions, the question was raised whether cancelation of the transit group would bring traffic perturbances. The operational characteristics of the yard are as follows: $m = 8$, *parw* = 37.5 min., *ex* = 62.5 min., *sigma* = 22.0 min. Table 3 gives a comparison of simulation results in both situations.

TABLE 3

Transit group	present	not present
No. of departure directions	1	3
Probability of having more than 4 departure tracks occupied	0.009	0.000
Departure waiting time:		
mean	5.7	5.5
standard deviation	0.7	0.6
Probability of departure without delay	0.759	dir. 1: 0.850 dir. 2: 0.926 dir. 3: 0.941 mean: 0.897

It follows that abandoning the use of the transit group reservoir and a direct connection of the departure tracks with the departure lines would even better the traffic situation in the yard in question.

Now, still another, more general, simulation model is being developed. It will incorporate reliability considerations. A report will follow.

Reference

[1] J. Kucharczyk, Danuta Machulec and J. Węgierski, *Congestion studies in railway stations and yards by digital simulation (I)*, Zastosow. Matem. 12 (1971), p. 79-90.

DEPT. OF STATISTICS AND OPERATIONS RESEARCH, UNIVERSITY OF WROCŁAW,
AND
RESEARCH INSTITUTE OF THE POLISH STATE RAILWAYS
DEPT. OF RAILROAD DEVELOPMENT IN MINING INDUSTRIAL DISTRICTS, KATOWICE

Received on 18. 9. 1970

APPENDIX: DEPARTURE SIMULATION PROCEDURE

```
procedure sygrod (lpr, lsym, dp, lpkmax, m, kw, parw, przw, ex, sigma,
    likier, p, czpt, pup, czod, socz, odb, bbb);
 value lpr, lsym, dp, lpkmax, m, kw, parw, przw, ex, sigma, likier;
 integer lpr, lsym, lpkmax, m, kw, likier, bbb;
 real dp, parw, przw, ex, sigma;
 array p, czpt, czod, socz, odb;
 Boolean array pup;
 begin
  integer k, log, mm, np, lzt, stan, nrzak, stanp, i, poc, lodj, nrpk,
      npp, nrw, lpk, kier, ldj, lzp;
  real tuniv, zeg, twyj, mwyj, rob, mbprz, mnprz, tnp, lokr, skr, s2kr,
      s3kr, rb, rb1, rb2, tpodr, fwyj;
  integer array tor[1: m], licz[0: m+lpkmax+1], nnr[1: lpkmax+1],
      zt[1: likier];
  array tp, to, rb, tw, nkier[1: m+lpkmax+1], czas[0: m+lpkmax+1],
      ccw, czt, cob, coo[0: 30], tpod, lok, sk, s2k, s3k, tpup[1: likier];
  procedure rej(c, g);
   value c; real c; array g;
   begin
    integer kk;
    kk: = entier(c/dp);
    if kk > 30 then kk: = 30;
    g[kk]: = g[kk]+1
   end rej;
  time: = 0; log: = lpr+lsym; mm: = m+lpkmax+1;
```

```
pocz:
  licz[0]: = 0; czas[0]: = 0;
  for i: = 1 step 1 until mm do
   begin
    czas[i]: = tp[i]: = to[i]: = tb[i]: = tw[i]: = 0;
    licz[i]: = 0
   end i;
  for i: = 0 step 1 until 30 do ccw[i]: = czt[i]: = cob[i]: = coo[i]: = 0;
  for i: = 1 step 1 until likier do
   begin
    lok[i]: = sk[i]: = s2k[i]: = s3k[i]: = tpup[i]: = 0;
    zt[i]: = 0
   end i;
  for i: = 2 step 1 until m do tor[i]: = 0;
  np: = stan: = lzt: = licz[1]: = tor[1]: = nrzak: = nrp: = 1;
  tp[1]: = zeg: = tb[1]: = mbprz: = lokr: = skr: = s2kr: = s3kr: =
      tpodr: = 0;
  lpk: = lodj: = ldj: = lzp: = 0;
  to[1]: = obsl; mnprz: = entry; tw[1]: = -1;
  for i: = 1 step 1 until likier do tpod[i]: = -10↑10;
  tpodr: = -10↑10; rb: = unif;
  for i: = 1 step 1 until likier do if rb ⩽ p[i] then go to wet 1;
  bbb: = 1; go to exit;
wet 1:
  kier: = i; nkier[1]: = kier;
zmst:
  if stan = 0 then go to przp;
  mwyj: = fwyj: = 10↑99;
  for i: = 1 step 1 until m do
   begin
    rob: = tor[i];
    if rob ≠ 0 then
     begin
      rb: = to[rob]; rb1: = tw[rob];
      if rb < mwyj ∧ rb1 < 0 then
       begin mwyj: = rb; nrzak: = i end;
      if rb1 > 0 ∧ rb1 < fwyj then
       begin fwyj: = rb1; nrw: = i end;
     end tor[i] ≠ 0
   end i;
  go to if mnprz < mwyj ∧ mnprz < fwyj then przp else
        if fwyj < mnprz ∧ fwyj < mwyj then w9 else wyj;
```

```
przp:
  np: = np+1;
  for i: = 1 step 1 until mm do if tw[i] = 0 then
   begin
    nrp: = i; go to ddal
   end i;
  bbb: = 2; go to exit;
ddal:
  tp[nrp]: = mbprz: = mnprz; tw[nrp]: = -1;
  mnprz: = mbprz+entry; rob: = unif;
  for i: = 1 step 1 until likier do if rob ≤ p[i] then go to wj1;
  bbb: = 1; go to exit;
wj1:
  kier: = i; nkier[nrp]: = kier;
  czas[stan]: = czas[stan]+mbprz-zeg;
  stan: = stan+1; zeg: = mbprz;
  if lzt < m
   then begin
          poc: = nrp; go to wja
        end lzt < m
   else begin
          lpk: = lpk+1; lzp: = lzp+1; nnr[lpk]: = nrp;
          if lpk > lpkmax then begin bbb: = 3; go to exit end;
          go to zmst
        end lzt ≥ m;
wja:
  lzt: = lzt+1;
  for i: = 1 step 1 until m do if tor[i] = 0 then
   begin
    tor[i]: = poc; tb[poc]: = zeg; to[poc]: = zeg+obsl;
    go to zmst
   end i, tor[i] = 0;
  bbb: = 4; go to exit;
wyj:
  kier: = nkier[tor[nrzak]];
  rob: = tpod[kier]+czpt[kier];
  twyj: = if mwyj < rob then rob else mwyj;
  if pup[kier] then
   begin
et1: rob: = tpup[kier]+czod[kier];
    tnp: = rob+erlang(1, socz[kier]);
    if twyj > rob then
```

```
      begin
       tpup[kier]: = tnp; go to et1
      end;
     if twyj+odb[kier] > tpup[kier] then
      begin
et2:   if tnp-rob < odb[kier] then
       begin
        twyj: = rob; rob: = tnp+czod[kier];
        tnp: = rob+erlang(1, socz[kier]);
        go to et2
       end tnp-rob < odb[kier];
      tpup[kier]: = tnp;
     end twyj+odb[kier] > tpup[kier]
    end pup[kier];
  k: = tor[nrzak]; tw[k]: = twyj; ldj: = ldj+1;
  if ldj > lpr then
   begin
    if twyj-to[k] = 0 then zt[kier]: = zt[kier]+1;
    lokr: = lokr+1; rb: = twyj-tpodr;
    skr: = skr+rb; s2kr: = s2kr+rb×rb;
    s3kr: = s3kr+rb×rb×rb; rb: = twyj-tpod[kier];
    lok[kier]: = lok[kier]+1; sk[kier]: = sk[kier]+rb;
    s2k[kier]: = s2k[kier]+rb×rb;
    s3k[kier]: = s3k[kier]+rb×rb×rb;
    rej(tb[k]-tp[k], ccw); rej(twyj-tb[k], czt);
    rej(to[k]-tb[k], cob); rej(twyj-to[k], coo);
   end ldj > lpr;
  tpod[kier]: = tpodr: = twyj;
  go to zmst;
w9:
  czas[stan]: = czas[stan]+fwyj-zeg;
  zeg: = fwyj; stan: = stan-1;
  lodj: = lodj+1; k: = tor[nrw];
  tp[k]: = tb[k]: = to[k]: = tw[k]: = 0;
  if lodj = lpr then go to kopr else if lodj = log then go to druk;
  licz[stan]: = licz[stan]+1;
cd:
  lzt: = lzt-1; tor[nrw]: = 0;
  if lpk > 0 then
   begin
    lpk: = lpk-1; poc: = nnr[1];
    if lpk > 0 then for i: = 1 step 1 until lpk do nnr[i]: = nnr[i+1];
```

```
      go to wja
     end lpk > 0;
    go to zmst;
kopr:
    for i: = 0 step 1 until mm do
     begin czas[i]: = 0; licz[i]: = 0 end;
    stanp: = stan; licz[stan]: = licz[stan]+1;
    tuniv: = zeg; npp: = np;
    go to cd;
druk:
    bbb: = 0;
exit:
    begin
     procedure print (k, int, tyt, l, ls);
      value k, ls; integer k, ls;
      Boolean int; string tyt;
      begin
       outstring (k, tyt);
       if int then outinteger (k, l) else outreal (k, l);
       if ls > 0 then outline (k, ls)
                 else if ls < 0 then outspace (k, abs(ls))
      end print;
     procedure stat (c, tyt);
      array c; string tyt;
      begin
       integer i; real s, s2, s3, lobs, r;
       lobs: = s: = s2: = s3: = 0;
       for i: = 0 step 1 until 30 do
        begin
         r: = c[i]; lobs: = lobs+r; r: = r×i;
         s: = s+r; s2: = s2+i×r; s3: = s3+i×i×r
        end i;
       outline(1, 2); outstring(tyt); outline(1, 2);
       print (1, false, 'NO. OBS. = ', lobs, -2);
       if lobs > 0 then
        begin
         print (1, false, 'MOMENTS 1-3:', s/lobs, -2);
         print (1, false, '', s2/lobs, -2);
         print (1, false, '', s3/lobs, 1);
         print (1, false, 'MEAN = ', dp×(s/lobs+0.5), -2);
         print(1,false,'STAND. DEV. = ', dp×sqrt((s2-s×s/lobs)/lobs),0)
        end lobs > 0;
```

```
    outline (1, 2);
    print (1, true, 'FREQUENCY DISTRIBUTION: INTERVAL
       LENGTH = ', dp, 1);
    outstring (1, 'INTERVAL NO., OBSERV. NO., FRACTION');
    outline (1, 1);
    for i: = 0 step 1 until 30 do if c[i] > 0 then
     begin
      outinteger (1, i); outinteger (1, c[i]);
      outreal (1, c[i]/lobs); outline(1, 1)
     end i, c[i] > 0;
    outline (1, 2)
  end stat;
print (0, true, 'DURATION OF SIMULATION (SECS.): ', time, 2);
print (1, true, 'TRIAL+REG. SIMULATION NOS.: ', lpr, 0);
print (1, true, '+', lsym, 1);
print (1, true, 'NO. OF DEPARTURE TRACKS: ', m, 1);
print (1, true, 'NO. OF DEPARTURE DIRECTIONS: ', likier, 1);
rb: = 0;
for i: = 1 step 1 until likier do
 begin
  print (1, true, 'DIRECTION NO. ', i, -2);
  print (1, false, 'DEPARTURE PROB. = ', p[i]-rb, -2);
  print (1, false, 'FR. TR. MIN. TIME LAG = ', czpt[i], 1);
  if pup[i] then
   begin
    outspace (1, 5);
    print (1, false,'PR. TR. INTERARRIVAL MEAN = ',socz[i],-2);
    print (1, false, 'PR. TR. OCCUPANCY TIME = ', czod[i], -2);
    print (1, false, 'FR. TR. ADVANCE TIME = ', odb[i], 1)
   end pup[i]
 end i;
outline (1, 1);
print (1, true, 'ENTRY: K = ', k, -2);
print (1, false, 'MEAN = ', parw, -2);
print (1, false, 'LAG = ', przw, 1);
print (1, false, 'SERVICE: MEAN = ', ex, -2);
print (1, false, 'STAND. DEV. = ', sigma, 2);
outstring (1, 'SIMULATION BEGIN:'); outline (1, 1);
print (1, true, 'STATE OF SYSTEM = ', stanp, -2);
print (1, true, 'ARRIVAL NO. = ', npp, -2);
print (1, false, 'TIME MOMENT = ', tuniv, 1);
outstring (1, 'SIMULATION END:'); outline (1, 1);
```

```
    print (1, true, 'STATE OF SYSTEM = ', stan, -2);
    print (1, true, 'ARRIVAL NO. = ', np, -2);
    print (1, false, 'TIME MOMENT = ', zeg, 1);
    rob: = zeg-tuniv;
    print (1, false, 'TIME INTERVAL = ', rob, 2);
    outstring (1, 'STATE DISTRIBUTION:
STATE, NO. OF OCCURRENCES, TIME SPENT, TIME FRACTION');
    outline (1, 1);
    for i: = 0 step 1 until mm do if licz[i] ≠ 0 then
     begin
      outinteger (1, i); outinteger (1, licz[i]);
      outreal (1, czas[i]); outreal (1, czas[i]/rob);
      outline (1, 1)
     end i, licz[i] ≠ 0;
    outline (1, 1);
    print (1, true, 'NO. OF TR. STOPPED AT ENTRY = ', lzp, 2);
    stat (ccw, 'ENTRY WAITING TIME');
    stat (cob, 'SERVICE DURATION TIME');
    stat (czt, 'TRACK OCCUPANCY TIME');
    stat (coo, 'DEPARTURE WAITING TIME');
    outstring (1, 'INTERDEPARTURE TIME DISTRIBUTIONS:');
    outline (1, 2);
    print (1, true,'ALL DIRECTIONS: DEPARTURES NO. = ', lokr, -2);
    if lokr > 0 then
     begin
      print (1, false, 'MOMENTS 1-3; ', skr/lokr, -2);
      print (1, false, '', s2kr/lokr, -2);
      print (1, false, '', s3kr/lokr, 0)
     end lokr > 0;
    outline (1, 2);
    for i: = 1 step 1 until likier do
     begin
      print (1, true, 'DIRECTION NO.', i, -2);
      rb: = lok[i];
      print (1, true, 'DEPARTURES NO. =', rb, -2);
      if rb > 0 then
       begin
        print (1, false, 'MOMENTS 1-3:', sk[i]/rb, -2);
        print  (1, false, '', s2k[i]/rb, -2);
        print (1, false, '', s3k[i]/rb, 0)
       end lok[i] > 0;
      outline (1, 1);
```

```
      print (1, true, 'NO. OF DEPARTURES WITHOUT
          DELAY = ', zt[i], 1)
    end i
  end of printing block
end sygrod
```

J. KUCHARCZYK (Wroclaw) i J. WĘGIERSKI (Katowice)

BADANIE SYMULACYJNE PRACY STACJI KOLEJOWYCH OSOBOWYCH I ROZRZĄDOWYCH (II)

STRESZCZENIE

W pracy opisano program, służący do symulacji pracy grupy odjazdowej stacji rozrządowych. Przytoczono także przykładowe wyniki symulacji. Dodatek zawiera program symulacji w języku Algol 60, przedstawiony w postaci procedury.
