ENGINEERING MECHANICS 2009
National Conference with International Participation
Svratka, Czech Republic, May 11 – 14, 2009

pp. 197–206
Paper **#108**

CDCSIS – SOFTWARE FOR DISCRETE CONTINUOUS SIMULATION

J. Čulík*

Summary: *Tasks from dynamics are very often solved on computer with help simulation. The software for continuous simulation is used for technical problems (for example MATLAB with SIMULINK) and discrete simulation for economic problems (GPSS). The article author has composed software CDCSIS (Combined Discrete Continuous SImulation System), which has advantages of bough software types. The CDCSIS system enable observe continuous course in time if differential equations are solved and enable changes of the structure (for example change of the number and form of differential equations, number of freedom degrees). The article presents solving of time events (structure change in given time), state events (change at critical state of signals) and a brief information about CDCSIS system with an example of system using.*

Keywords: simulation, combined discrete continuous simulation, modelling

1. Introduction

The science work has follow steps:

1. The studied system is earmarked from objective reality.
2. The system is divided to elements and their behaviors are defined by formulas, differential equations or verbal - mathematical model.
3. Verification of mathematic model and reality, searching of values of parameters and sensibility on its values, validity of a model etc.

If the system is dynamic (variable in time) the 3rd part can be provided with computer simulation. The simulation model is built on computer to be its behavior in time similar as reality. If the simulation model has different behavior then the mathematic model, model details and parameters have to be changed – model verification.

If the simulation is used for design of some equipment then the simulation model is build

* Prof. Ing. Jan Čulík, DrSc.: Faculty of Biomedical Engineering, Czech Technical University in Prague; Sítná 3105; 272 01 Kladno; tel.: +420 312 608 208, fax: +420 312 608 204; e-mail: culik@fbmi.cvut.cz

on a computer. The simulation model composing has these steps:

1. An **objective realit**y which will be studied is defined – for example engine.
2. The concrete **dynamic system** which will be studied is specified - for example vibration of the engine.
3. The problem is studied from mathematic view and the **mathematic model** is composed – for example differential equations of vibration and turning speed conditions.
4. The **simulation model** according to the mathematic model is built on a computer.
5. **The experiments with the simulation model** are provided on computer. The real cases are modeling on computer and the results are compared with reality.
6. **Verification** – the mathematic and simulation models and its parameters are changed to be their behavior preferable same as reality behavior.
7. **Sensitivity analysis** – the influence of parameter values to system behavior is studied.
8. **Using the simulation model** in praxis for design of new equipments.

The simulation software can be divided to:

1. **Continuous systems** - the system which changes its signals continuous in time (MATLAB, SIMULING).
2. **Discrete system** – the system which changes its state and/or structure in finite time points (GPSS).

The **Combined Discrete Continuous SImulation System CDCSIS** is new type of software for composing of simulation model on computer. The continuous simulation can solve invariable system structure only and the discrete simulation enable system structure changes but their signals can be changed in finite time points only. The article author has composed software CDCSIS (Combined Discrete Continuous SImulation System), which has advantages of bough software types. The CDCSIS system enable observe continuous course in time if differential equations are solved and enable changes of the structure (for example change of the number and form of differential equations, number of freedom degrees).

The problem is solved with help CDCSIS in follow way. The system has to be divided to elements. The each element has attributes which define state of element. Some attributes can be used as output signals of element and input signals to the other element. The CDCSIS software is implemented as the library of procedures at languages FORTRAN, PASCAL and C++ (‘basic language’). The simulation model is a program at ‘basic language’ and system statements are substituted by the library procedure callings. The user compiles blocks; the blocks are subroutines for element behaviors in some system state, for example right sides of differential equation or program for any changes of structure. The blocks are located at the procedure *blocks.* The main program besides arbitrary statements includes calling of library procedures which realize system statements, for example requirement on scheduling of the calculation according to block subroutine and/or integration of vector, graph output, animation, canceling of calculation according to block subroutine and/or integration. The user need not learn some special language syntax but it suffices if he acquaints with library of system procedures and if he is able to write programs at ‘basic language’.

The European society EUROSIS (Gent / Belgium, http://biomath.ugent.be/~eurosim/) coordinates the research at simulation area and organizes congresses on simulation. Czech national part of this society has name CSSS (Czech and Slovak Society of Simulation).

2 System CDCSIS – Combined Discrete Continuous SImulation System

2.1. CDCSIS software

The CDCSIS software is the library of subroutines for interpretation of system statements. The system statements are realized as calling of library subroutines at the same programming language. The CDCSIS library is completed at FORTRAN, PASCAL and C++. The simulation program can be a part of every program at this language. It means that the program has a declaration part, input of parameters and initial values, estimate calculation, simulation part, assessment of results etc. The graphic output of results and animation can be realized by system subroutines. The C++ version will be presented in this article.

2.2. Elemental notions

If the simulation is used the follow elemental notions are used:

- **The dynamical system** is object of our interest.
- **The element** is part of dynamical system. The behavior of the system is described by behavior of elements and their interconnection.
- **The attributes and/or signals** define state of element. The attributes are inner object parameters and signals are output from one element and/or input to the other element. The attributes can be programmed as structure (type *struct*).
- **The activity** is element which is part of system in all observed time.
- **The transaction** is the element which goes to the observed system changes its position in system and after some time goes out. The statements *new* and *delete* are suitable for its programming.
- **The block** is a subroutine for calculation of element outputs (activities or transactions).
- **The discrete block** is calculated at one time point only.
- **The event** is putting or canceling the block or integrator to and/or off simulation calculation.
- **The time event** is event at the given time point.
- **The state event** is event at some critical values of attributes.
- **The continuous block** is calculated at each calculation step (The Runge – Kutta's integration method has 4 calculation steps in one time step).
- **The check block** is calculated at the end of each time step and it checks event formation.

2.3. Solving of differential equations

The system of differential equations must to be transformed to system of differential equations the first order, for example the equation

$$2\ddot{x} - 4\dot{x} + 10tx - 3t^2 = 0$$

is transformed to

$$\dot{x} = x1$$
$$\dot{x}1 = 2x1 - 5tx + 1{,}5t^2$$

The right sides of differential equations calculation has to be programmed as block and the numerical integration of $\dot{x}, \dot{x}1$ and its storage to $x1$, x is demand only. The initial conditions of $x1$, x are their values before the integration.

The CDCSIS software uses numerical integration Runge – Kutta's fourth order. This method need calculate the right sides of differential equations four times. The initial time step (see function WAIT) is used as integration step and it is changed to be calculated at event times.

2.4. Time step choosing

The user chooses the time step according to integration occurrence and simulation software changes the time step to be calculated exactly in all event times.

The ordering of state event is programmed in follow way: the state event comes if some variable (for example x) leave off positive value. The correct time of state event software CDCSIS searches with help lineal interpolation. The last positive signal x is stored and if the signal x in new time step is negative then the correct time step with zero's signal is estimated by linear interpolation. The calculation is given back to the time when signal x was equal zero (see fig. 1).

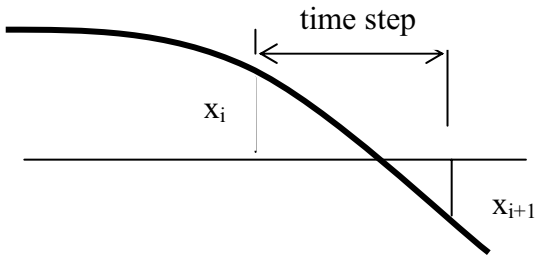

Fig. 1. Searching of time of state event.

2.5. Ordering the blocks to calculation process

The simulation software must order the blocks to calculation process according to signal flow direction, if an output of block *A* is used as an input in block *B* then the block *A* must be calculated in the same time before the block *B.* The software CDCSIS orders blocks according to his priority. The user assigns each block by the priority. If the block *A* has higher priority then the block *B* than the block *B* will be calculated after block *A*.

The priority determines the block type too, the continuous block has the positive priority, discrete block priority equal to zero and check block has negative priority. It means that the continuous blocks are calculated at first, then the discrete blocks and the check block at last.

3. Composing of simulation model at CDCSIS system

The simulation program consists from system declaration (declaration of system variables and library procedures), user declarations, procedure *blocks* with block programs and main program. The values which exist through whole simulation calculation can be declared as variables, vectors or matrixes. The transactions - elements which enter to simulation, change position and go out can be declared as groups of structures using statement *new* and *delete.* The simulation program calls the transactions with help pointers.

The observed dynamical system has to be divided to elements. The programs of behavior of elements at some situation are named blocks. The blocks are written in procedure *blocks* which starts by the statement *switch*(*sj*). The components of statement switch are blocks; each block is closed with statement *break.* The calling of procedure *schedule* is demand to putting of block program to simulation model in given time. The blocks is marked with value of switch *sj*. The block can be continuous (right sides of differential equations), discrete for the once provided calculation and/or check. The check block controls with help function *event* if a state event arrives.

The main program prepares computing data and with help calling of library subroutines orders calculation scheduling according to blocks (see *schedule*), integration of vector (*integr*), canceling of blocks and/or integrations which have be previously put the calculation process (*cancel, cancelblock, cancelint*), graphic output of some signals (*graf*) and animation (*anim*). If all requirements are finished then the simulation system is call with help statement *wait.* If the simulation is finished then the calculation continue according to statements behind calling of *wait.* The demand on calculation according to blocks, integration and/or their canceling can be given from block too. The continuous blocks realize the continuous calculation and discrete and/or check blocks realize the change of system structure.

4. CDCSIS library and system variables

The CDCSIS system uses these main variables:

simtime ...simulation time,
step ...time step,
ptrans ...pointer to service transaction in time of jump to block,
stest ...number of check prints,
sj ...switch value in time of jump to block.

The CDCSIS library in language C++ has these procedures (in alphabet ordering):

***void anim*(*double dx, double dy, double xmax, double ymax, double x, double y, double alfa, int n, double *a, double *b*)**

The procedure for an animation some object. The object is drawn at the monitor and it is moved and turned in time. The object is given as polygon line and/or circles. The procedure has the parameters:

- *dx,dy* … horizontal and vertical distance of the monitor at user coordinates.
- *xmax, ymax* ...maximal sizes of draw object.
- *a, b* ...pointers to local coordinates a_i, b_i of the drawing object. The object is drawn as a polygon (given points are connected by lines). If the coordinate $a_i >= 1000$ than the circle is drawn with radius b_i at centre with previous coordinates for positive b_i and next point is not connected with previous point for negative *b*.

- *n* ...number of point a_i, b_i.
- *x, y, alfa* ...global coordinates of object local coordinate origin and its turning angle.

***void beg*(*double starttime*)**

The procedure clears all system data and set parameter *starttime* to *simtime* value (simulation time). The procedure must be used only if the simulation calculation will be called once more.

***int test*()**

The behavior of simulation model can be checked with help check prints. The CDCSIS print information of each system procedure activity and/or of each block calculation. The control prints are provided for *stest*>0 (system variable) and *stest* is decreases. The logical function *test* can be used for check print in blocks; the function *test* has value *true* for *stest* > 0. The graphic output and/or animation are provided till *stest* will be zero. The *stest* start value is zero and can be changed anytime.

***void cancel*(*string name*)**

The procedure cancels blocks and integrators labeled with the name *name*.

***void cancelblock*(*struct rblock *p*)**

The procedure cancels the block with pointer *p*.

***void cancelint*(*struct rint *p*)**

The procedure cancels integrator with pointer *p*.

***int event*(*double x, double p*)**

The logical function *event* checks formation of state event. The function is *true* if *x* leaves off positive value and the event time is put to the *simtime* (simulation time); *p* is work variable.

***void graf*(*double *p, double ampl, string text*)**

The procedure orders the scheduling on a graphic output of one signal. The procedure *graf* can be called maximal 5 times (maximal 5 graph outputs may be ordered). The procedure has parameters:

- *p* ... pointer to drawing signal,
- *ampl* ...maximal absolute value of signal, if the signal is greater then *ampl* than the *ampl is* automatically two-ply,
- *text* ...graphic output is labeled with *text.*

***void hold*(*double timeh, int block*)**

The procedure enables programming of object history. The block with number *block* will be called after time *timeh* with the same parameters as the block where statement *hold* is situated.

***struct rint integr*(*int np,double *pvin,double *pvout,double *pvw,string ptext*)**

The procedure orders the integration of a vector and it has the parameters:

- *np* … length of integrated vector.
- *pvin, pout, pvw* ...pointers to input, output (length *np*) and work vector (length 2*np*).
- *ptext* ... alphanumeric name of integrator.

***struct rblock schedule*(*double timep,int iblock,int priorp,double *transp,string ptext*)**

The procedure orders the scheduling the block to simulation calculation, it has the parameters:

- *timep* ...time of filling the block to calculation,
- *iblock* ...number of block (value if switch *sj* – see procedure *blocks*),
- *priorp* ...priority and type of block. The block is continuous if *priorp* >0, discrete for =0 and check block for <0,
- *transp* ...pointer to service object (transaction),
- *ptext* ...alphanumeric name of scheduled block.

void wait(double endtime, double gstepp)

The simulation calculation is started and it is provided according to the requirements which have been given by system statements. The procedure has parameters:

- *endtime* ...simulation calculation will be finished at this time,
- *gstepp* ...basic time step and time step of animation and graphic outputs. The time step is changed it simulation time to be punctually calculated in event times.

5. Example of simulation program

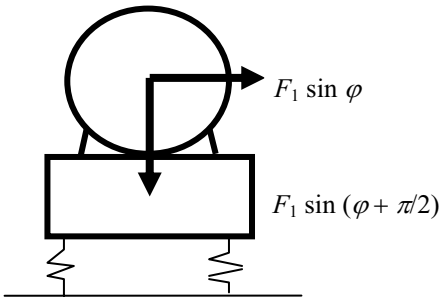

Fig. 2. Schema of engine inertia forces.

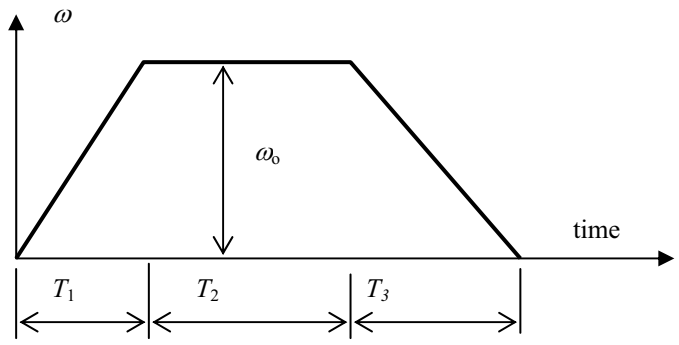

Figure 3. Changing of turn velocity in time.

Let us compose the program for simulation of engine vibration (see fig. 2) in time. The turning velocity is changed according to fig. 3. The engine has mass m = 500 kg and mass inertia moment J_p = 250 kgm^2, the spring constant of all springs is $k = 10^6$ N/m, the amplitude if harmonic force is F_1 = 1000 N, the times of increase, constant and decrease of turn velocities are T_1 = 5 s, T_2 = 1 s, T_3 = 5 s. The logarithm decrement of dumping is 0.9. The stable turning velocity is $\omega_0 = 100$ s^{-1}.

The differential equations of vibration are

$$\ddot{y}m + b_1\dot{y} + ky = F_1 \sin\varphi$$
$$\ddot{\alpha}J_p + b_2\dot{\alpha} + 1.62\alpha k/2 = F_1 \sin(\alpha + \pi/2)$$

The highest derivations are ($2\omega_{b1}=b_1/m$, $2\omega_{b2}=b_2/Jp$)

$$\ddot{y} = \frac{F_1 \sin\varphi + ky}{m} - 2\omega_{b1}\dot{y},$$

$$\ddot{\alpha} = \frac{F_1 \sin(\varphi + \pi/2)0.7}{J_p} - 2\omega_{b2}\dot{\alpha}$$

The accelerations of turn angel are

$$\ddot{\alpha} = \frac{\omega_0}{T_1} \text{ in the interval } <0,T_1>,$$

$$\ddot{\alpha} = 0 \text{ in the interval } <T_1,\ T_1+T_2>,$$

$$\ddot{\alpha} = -\frac{\omega_0}{T_3} \text{ in the interval } <T_1+T_2,\ T_1+T_2+T_3>.$$

The values $y, \dot{y}, \ddot{y}$ are stored at vector **y,** $\alpha, \dot{\alpha}, \ddot{\alpha}$ at vector **alfa** and $\varphi, \dot{\varphi}, \ddot{\varphi}$ at vector **fi**.

Program in language C++

```
double pom,k,T1,T2,T3,omb1,omb2,m,Ip,F1,omega,y[3],alfa[3],fi[3],yp[4],alfap[4],fip[4];
int i,i1,i2;
char text[100];
void blocks(){
 switch(sj){
 case 1://continuous block which calculates right sides of differential equations
    pom=F1*sin(fi[0]);
    y[2]=(pom-k*y[0])/m-2.0*omb1*y[1];
    alfa[2]=(pom*0.7-k*0.81*alfa[0])/Ip-2.0*omb2*alfa[1];
   break;
 case 2:fi[2]=0.0;// acceleration in time T1
   break;
 case 3:fi[2]=-omega/T3;// acceleration in time T1+T2
   break;
 case 4:cancel("in1");// cancel of integrator with name “in” and cancel of harmonic force
                      // F1=0.0
      }}
//main program
int main(){
T1=5.0;T2=1.0;T3=5.0;omega=100.;k=1.e6;m=500.;Ip=250;F1=1000.;// given values
omb1=0.1418*sqrt(k/m);omb2=0.1418*sqrt(k*0.81/Ip);// dumping contants
y[0]=0.0;y[1]=0.0;fi[0]=0.0;fi[1]=0.0;alfa[0]=0.0;alfa[1]=0.0;// initial conditions
fi[2]=omega/T1;// turning acceleration in time = 0
schedule(0.0,1,1,0,"y,alfa");// ordering of calculation of right sides of differential equations
schedule(T1,2,0,0,"T1");// ordering of change turning acceleration in time T1
schedule(T1+T2,3,0,0,"T2");//  - “ – in time T1+T2
```

```
schedule(T1+T2+T3,4,0,0,"T2");// ordering of stop of harmonic force in time T1+T2+T3
graf(&y[0],0.01,"vertical displacement");// ordering of graphic output
graf(&alfa[0],0.01,"turning");//  - ” -
integr(2,&fi[1],fi,fip,"in1");// ordering of integration of turning acceleration
integr(2,&y[1],y,yp,"int y");// ordering of integration y’’
integr(2,&alfa[1],alfa,alfap,"int alfa");//   - “ -  α’’
wait(T1+T2+T3+0.5,0.02);// jump to simulation system
return(0);
}
```

Comment: The program does not contain part of the system function declaration. This part can be copied to factual program only.

6. Conclusion

The simulation system CDCSIS is suitable for problems where the differential equations change their form and number of unknowns or for the systems which has any other changes. The simulation model can be a part of large-scale program. The block can be all what it is possible to program, for example measured data evaluation, random data, human behavior if a drive simulator is composed, result of iteration calculation.

Demands on changer of calculation structure can be given from blocks according to momentary situation with help time and state events.

Examples of the other problems which were solved with help simulation system CDCSIS.

1. Oscillation of vehicle wheel. The border between overpower of friction and changing of degrees of freedom were solved with help ‘state event’ and system function *event.*
2. Bridge vibration. The vehicle was considered as transaction. The vehicle going to bridge and going out were solved as time event.
3. Human fall to colliery. The man incidence with the wall was solved with help ‘state event’.
4. Moving of balls at billiard table. The balls shock and shock the ball on a barrier were solved as ‘state event’.
5. Model of hospital beds occupation. The system is discrete and has no differential equations. All bed occupation changing was solved as ‘time events’.
6. Model of influenza epidemics.
7. Spine deformation as an effect of brace. The spine was searched as partly stiff (vertebra parts) and elastic (inter-vertebrae parts). The changes of differential equations at part border were searched as time event.

7. Acknowledgement

The research was support by grant SM6840770012 “Trans-disciplinal Research at Biomedical Engineering Area”.

8. References

Čulík, J. (2008) System CDCSIS for Discrete Continuous Simulation. Proc. 'Computer mechanics 2008", Nečtiny 3.11.-5.11.2008, West Bohemia University in Plzeň, Czech Society for Mechanics, Czech Committee IFToMM. CD rom.

Čulík, J. (2008) CDCSIS PASCAL. Manual - Faculty of Biomedical Engineering. CTU Prague.

Čulík, J. (2008) CDCSIS C++. Manual - Faculty of Biomedical Engineering. CTU Prague.

Čulík, J. (1986) Solving of Combined Simulation Problems. (Řešení problémů kombinované simulace). In: Automatizace, 29, č. 4, s. 107 – 110. SNTL Prague.

Čulík, J. (1989) Experience with Software for Continuous-Discrete Simulation (Zkušenosti s programovým vybavením pro kombinovanou diskrétně spojitou simulaci). In.: Information bulletin „Computer techniques at universities" (Výpočetní technika na vysokých školách). CTU Prague.