Modelling of technological machines motion

Yu B Chechulin[1], S K Buynachev[1]
[1]620002, 19 Mira street, Ekaterinburg, Russia Ural Federal University, Russia
email: uchechulin@mail.ru

Abstract: The software for modelling dynamic processes of technological machines drive mechanisms using algorithmic language PYTHON.

Introduction

Dynamic analysis of transition and stable power processes in drive mechanisms of technological machines is based on solving motion equations. The final task of defining optimal characteristics of variable factors delivering minimal of power impacts at the design stage suggests mathematic modeling. Motion modeling for a large complex system is always related to speed of action, memory amounts and universal code. The use of ready packages in mathematic modeling is limited by many factors.

We believe Python algorithmic language and the following approach offer a solution to that. For kinematic scheme figure 1 see graph figure 2.

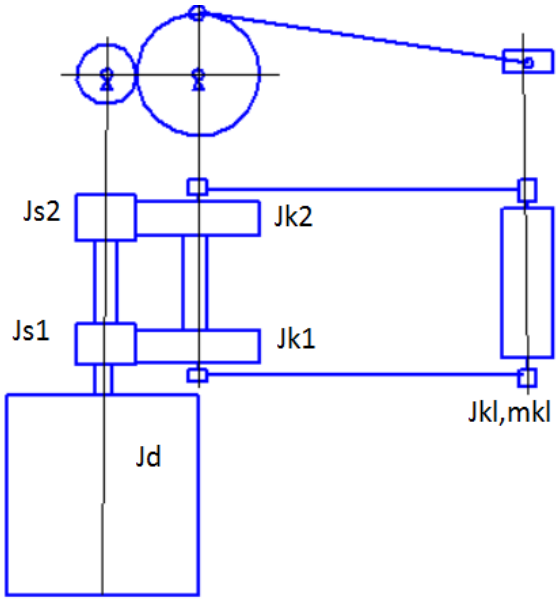

Figure 1. Kinematic scheme of CPR mill.

The graph describing the structure of this machine at figure 2. Peaks are masses and connections are elastic connections. We define

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

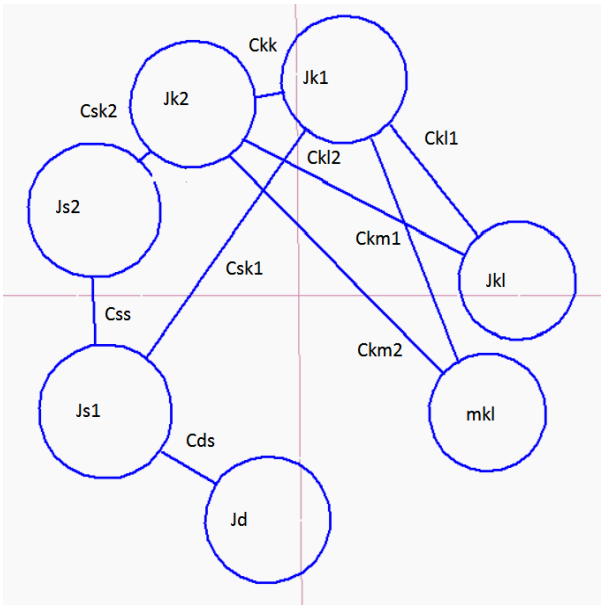

Figure 2. Graph of machine structure (ϑ_I-mass coordinates (peaks), η_j-mass coordinates difference.

We use system of coordinates one of which describes position of masses,
$\vartheta=(\vartheta_1,..,\vartheta_n)^T$,
And the other
$\eta=(\eta_1,..,\eta_n)^T$,
– connections between them. These coordinates are expressed via functions with generalized coordinates
$\tau(\vartheta,q)=0$
$\pi(\eta,q)=0$
Free oscillation equation is

$$.\tau_q{}^T[\tau_\vartheta{}^T]^{-1}\Theta\vartheta''+\pi_q{}^T[\pi_\eta{}^T]^{-1}F=0.$$

This equation requires Jacobi matrixes, reverse matrixes, then we formulate second derivatives of quasi-inertial coordinates and calculate the strength of elastic connections. To simplify the system we use graph theory [1]. However as a result of linearization and simplification system adequacy is lost. It must be mentioned that nonlinearity of lever systems and gaps. This equation provides direction to several simplifications. First of all each connection equation can be transformed as an expression of one coordinate
$\vartheta_i+\tau_i(q)=0$, $i=1,n$
$\eta_i+\pi_i(q)=0$, $i=1,m$.
Secondly we take quasi-inertial coordinates as generalized coordinates
$\vartheta+q=0$.
As a result the equation of free oscillation takes a simpler form,
Expression F we will analyze later
$\Theta\vartheta''+\eta_\vartheta{}^TF=0$
In which matrix Θ is diagonal and elastic forces result in coordinates ϑ. Lagrange equations of the Ist type also simplify and look as
$\Theta\vartheta''-\Lambda_T=0$,
$F-\Lambda_I=0$,
$\Lambda_T+\eta_\vartheta{}^T\Lambda_I=0$.
Value of undetermined multipliers Λ_I depends on coordinates η, which are expressed from masses via generalized coordinates. To simplify programming connections between peaks of the graph and

coordinates η we can use splitting procedure [3]. Into each connection we put the peak of new type and bind with the peaks of coordinates ϑ. Number of new peaks equals the number of old connection equations. We get a two-segment graph figure 2. Each segment of the graph has one class that can be developed into the necessary direction.

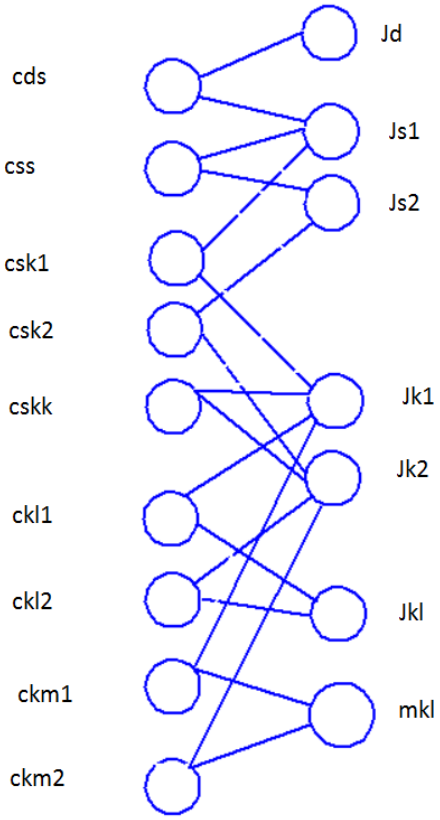

Figure 3. Two-segment graph

Right segment class can be described as [4].

```
class M:
def __init__(self,m):
self.m,self.f,self.x,=m,0.0,[0.0,0.0,0.0]
def step(self,dt):
if self.m==0: return
self.x[2]=self.f/self.m
self.x[0]+=0.5*dt*self.x[1]
self.x[1]+=dt*self.x[2]
self.x[0]+=0.5*dt*self.x[1]
```

Left segment class looks as follows.

```
class C:
def __init__(self,a,b,c,r=0.0):
self.a,self.b=a,b
self.c,self.r=c,r
self.f,self.ds,self.dv=0.0,0.0,0.0
def F(self):
ds,dv=self.a.x[0]-self.b.x[0],self.a.x[1]-self.b.x[1]
self.f = ds*self.c+dv*self.r
```

self.a.f-=self.f
self.b.f+=self.f
Creation of right segment objects.
Jd,Js1,Js2,Jk1,Jk2 =M(150),M(0.1),M(0.1),M(0.5),M(0.5)
Jkl,mkl =M(10.0),M(1450.0)
V=[Jd,Js1,Js2,Jk1,Jk2,Jkl,mkl]
Creation of left segment objects.
cds,css=C(Jd,Js1,3540.0,12.5), C(Js1,Js2,3140.0,12.5)
csk1,csk2,ckk=C(Js1,Jk1,3540.0,12.5), C(Js2,Jk2,3140.0,12.5)
ckkl1,ckkl2= C(Jk1,Jkl,3540.0,12.5), C(Jk2,Jkl,3140.0,12.5)
ckm1,ckm2= C(Jk1,mkl,3540.0,12.5), C(Jk2,mkl,3140.0,12.5)
E=[cds,css,csk1,csk2,ckk,ckkl1,ckkl2,ckm1,ckm2]
Here programming stops and two lists V and E should be scanned. As a result we present graphs at figure 4.

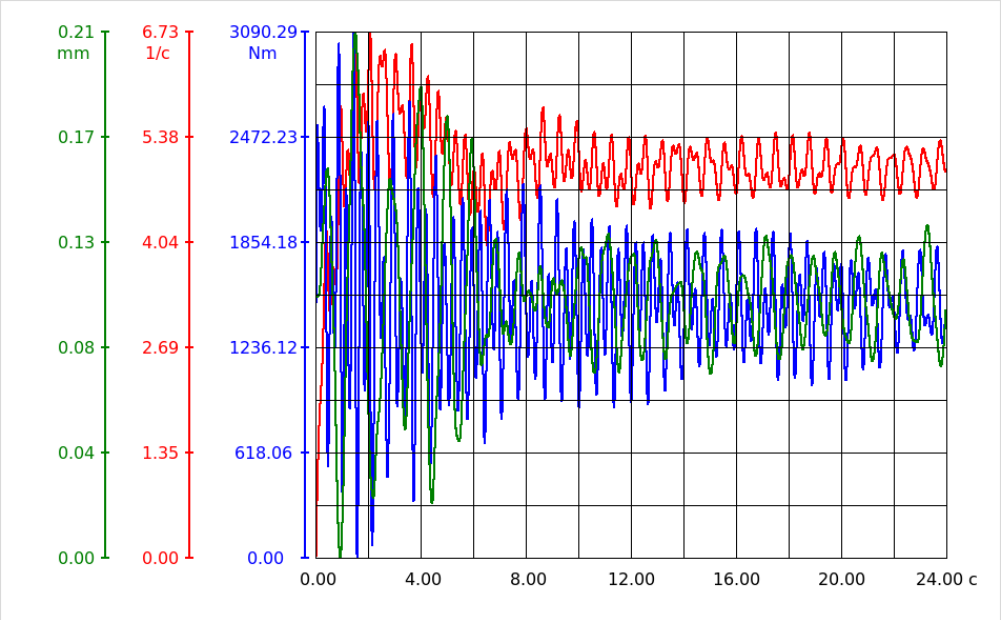

Figure 4. Motion and load at acceleration and stable motion (Red - speed of engine rotor, blue - engine rotor momentum, green - oscillation of mill crate position in a horizontal plane).

Results and Discussion
In this case connections in the graph have a meaning of information flows. Matrix transformations are not needed. As a result we get a solution reflecting mill motion. For the first time we get a description of crate in a horizontal plane so easily and precisely.

Conclusion
Suggested modeling solution has the following advantages:
1. Number of masses and hardnesses is not limited.
2. No limitation in nonlinearity description.
3. Simplicity of programming.

References
[1] Veitz V L 1969 *Dinamika mashinnykh agregatov* (Mashinostroenie)
[2] Evstigneev V A 1985 *Primenenie teorii grafov v programmirovanii* (M. Nauka)
[3] Thulasiraman M, Grafy K 1984 *Seti i algoritmy: translated from Engl* (M. Mir)

[4] Bizli D 2000 *Yazik programmirovanija Python* (K. DiaSoft)